

Technical University of Crete
School of Electronics
And Computer Engineering

Thesis

A Smart Home Management System on the Cloud

Panagiotis Kostopoulos

Examination Committee

Evrpidis Petrakis, Professor (Supervisor)

Konstantinos Kalaitzakis, Professor

Dr. Stelios Sotiriadis, Research Fellow

CHANIA 2016

Abstract

The objective of this thesis is to create a smart home management system on the cloud, which offers users a service from which they can manage and control their homes remotely and any given moment by making use of the concepts of Cloud computing and the Internet of Things and their capabilities. The system uses existing GEs, GEs implemented for the purpose of this thesis and computing resources provided by Cloud infrastructure and web pages to provide these services to the users through a Graphical User Interface.

The implemented system's application consists of several web services which utilize the components mentioned above to provide users with the ability to manage their smart homes remotely through a Graphical User Interface. Users can manage several aspects of their home, such as controlling the operation of its devices, managing certain aspects of their operation, utilizing the provided capability for automatic control of their home and managing the home's users and basic information. They can also receive useful information about their home, including information about several aspects of the devices' operation, notifications about important device updates and basic information about the home and its users.

For the purpose of testing and demonstrating the functionality of this system, several computer programs were implemented to simulate the home's devices and appliances, as well as the access point to which they are connected and through which communication with our system is implemented.

Thanks

I would like to thank my supervising professor Mr. Petrakis for his guidance until the completion of this thesis. All postgraduate students and members of the lab, Alexandros, Kostas, Stelios and Spyros, and all undergraduate students, for their help on this project. Finally I would like to thank my family for all their support throughout my studies.

This thesis is dedicated to my family
who have supported me during this entire process,
to my friends, and to everyone by my side
throughout this journey.

Contents

| | |
|---|----|
| Abstract..... | 2 |
| Thanks..... | 3 |
| 1. Introduction..... | 5 |
| 1.1. Smart Home..... | 5 |
| 1.1.1. Internet of Things (IoT)..... | 5 |
| 1.1.2. Cloud Computing | 5 |
| 1.2. Motivation | 8 |
| 1.3. Proposed Solution | 8 |
| 1.4. Structure | 10 |
| 2. Background and Existing Technologies | 11 |
| 2.1. Smart Home - Previous Work | 11 |
| 2.2. Internet of Things | 12 |
| 2.2.1. Definition..... | 12 |
| 2.2.2. Smart Devices..... | 12 |
| 2.2.3. Sensors..... | 13 |
| 2.2.4. Internet of Things and Cloud Computing..... | 13 |
| 2.3. Cloud Computing | 14 |
| 2.3.1 Definition..... | 14 |
| 2.3.2. Service Models | 14 |
| 2.3.3. Deployment Models | 15 |
| 2.3.4. Cloud Computing Providers | 16 |
| 2.3.5. Service-Oriented Architecture..... | 17 |
| 2.3.6. Existing Technologies | 18 |
| 2.3.7. Virtualization of Resources | 21 |
| 3. Design and Implementation | 22 |
| 3.1. Architecture..... | 22 |
| 3.1.1. Front End - Sensor Layer & Access Point..... | 22 |
| 3.1.2. Back End & Users | 23 |
| 3.1.3. Data Flow | 24 |
| 3.1.4. GEs & Application | 27 |
| 3.2 System Implementation..... | 30 |
| 3.2.1. Front End Implementation (Sensor Layer)..... | 30 |
| 3.2.2. Back End Implementation | 39 |
| 3.2.3. Use of the Application (UI Examples) | 52 |
| 4. Conclusions and Future Work | 61 |
| 4.1. Conclusions | 61 |
| 4.2. Restrictions..... | 62 |
| 4.3. Future Work | 63 |
| Bibliography | 64 |

1. Introduction

1.1. Smart Home

1.1.1. Internet of Things (IoT)

The term The Internet of Things ^[1] (IoT) refers to the network of physical objects, or “things”, embedded with electronics, software and sensors that feature an IP address for internet connectivity which enables communication between these objects and other Internet-enabled devices and systems to collect and exchange data, also allowing them to be sensed and controlled remotely.

Over the past years, a plethora of these objects has entered our lives. Environmental monitoring, medical and healthcare systems, building and home automation, manufacturing, transportation are only a few of the many aspects in which they find application.

The growth of the Internet of Things is indicated by the rapid increase in the number of connected devices over the past years. There are currently 25 billion connected devices, while in 2003 there were only 500 million, and it is projected that by 2020 the number of connected devices will be 50 billion or even as high as 75 billion.

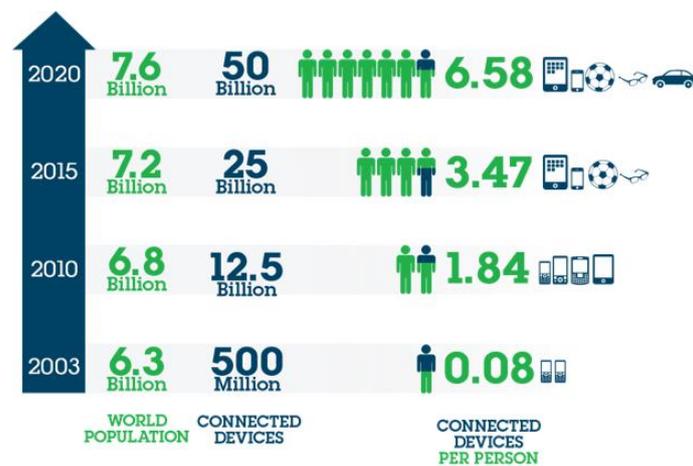


Figure 1 : Connected devices to global population ratio, over time

(Statistics sources : Cisco (CSCO), Morgan Stanley (MS))

1.1.2. Cloud Computing

Cloud Computing ^[2] enables universal, convenient, on-demand access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services), available rapidly to the user through little management effort or interaction with the service provider.

1.1.2.1. Advantages and disadvantages

This cloud model is composed of six essential characteristics which comprise its main advantages :

On-demand self-service. A user can obtain computing resources, such as server time and network storage, simply and on-demand without having to interact with each service provider.

Broad network access. Services can be accessed over the network at any time or place on a variety of platforms (smart phones, tablets, laptops and workstations).

Resource pooling. The provided resources are pooled to serve multiple users by dynamically assigning physical and virtual resources according to user demand.

Virtualization of resources. Virtualization allows multiple operating system images to run on the same physical resources and resource pooling, mentioned above.

Rapid elasticity. Users can elastically and frequently manage resources according to their needs, increasing resources when needed or releasing them when they are no longer required.

Measured service. Resource usage can be monitored and controlled, providing transparency and the ability to charge the user based on the resources used and the duration of that use, which in turn allows the user to control the cost.

These characteristics also provide the following advantages :

Reduced cost. Users, mainly businesses, can be greatly benefited by the use of provided services. Service providers have complete responsibility for upgrades and maintenance, thus reducing the need for specialized staff. Also, the equipment required by the business is greatly reduced, since resources are available through a simple internet connection.

Increased productivity. Services can be used simultaneously and in parallel by multiple users. This way, a business can make files available to all employees simultaneously through a main computer, thus keeping them up to date with all changes and ensuring maximum productivity.

As with all new technology, Cloud Computing troubles consumers as to whether it is a safe and sufficient solution for them. Despite the advantages provided by the Cloud, there are numerous questions regarding the negative effects resulting from its use. Some of these are “Is my data safe?”, “Can I access services in the case of a network failure?”. We will attempt to address these concerns.

Security and Privacy. When referring to Cloud Computing the question that arises is if the data hosted is safe. The fact is that users are very reluctant to entrust sensitive data to third parties, as their safety is not fully guaranteed by any provider and technical details, such as the location of their data, are not disclosed to consumers. According to studies carried, there is always the risk of sensitive data being exposed to the public on the internet. So we conclude that at a time when the consequences and possible costs of such failures are increasing, companies that manage confidential data should develop better ways to assess the safety and privacy practices on their Cloud Computing services, in order to gain public confidence.

Data deletion. The safe and effective deletion of data upon termination of the client-provider contract is not guaranteed. The method by which data is deleted and the possible existence of a copy, which can be used by unauthorized Cloud users, is not made known to the user.

Legal matters. Compliance with the law is a very important element between the consumer and the provider, since the legislation on the issue of cloud computing is considered problematic. Users today are not sufficiently aware of their rights against the provider and what the rights of the provider on their data are. This issue is made worse by the fact that the cloud infrastructure hosting the data is not in one place on the planet but is partitioned across the world, since there isn't one single legislative framework for the consumers' protection.

Network usage. As was mentioned, access to Cloud services can be accomplished through any device with internet access. This advantage turns into a disadvantage in the event of a network access failure, since access to services or files becomes impossible.

1.1.2.2. Overview

Cloud computing offers three service models. Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

Software as a Service (SaaS) provides the consumer with the capability to use the provider's applications running on a cloud infrastructure.

Platform as a Service (PaaS) provides the consumer with the capability to deploy consumer-created or acquired applications onto the cloud infrastructure, using tools supported by the provider in a platform interface.

Infrastructure as a Service (IaaS) provides the consumer with the capability to provision computing resources where he is able to deploy and run arbitrary software.

It is deployed in four models. Private cloud, Public cloud, Community cloud, and Hybrid cloud.

In a **Private cloud**, the cloud infrastructure is provisioned for exclusive use by a single organization.

In a **Public cloud**, the cloud infrastructure is provisioned for open use by the general public.

In a **Community cloud**, the cloud infrastructure is provisioned for exclusive use by a specific community of consumers.

In a **Hybrid cloud**, the cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together.

Service-oriented architecture is the most modern architectural approach in terms of designing and developing complex IT systems. It is based on the logic that any larger problem can be solved optimally and its complexity addressed effectively, by dividing it into smaller problems which compose it. The same applies to systems based on web services, such as ours. Such systems consist of a set of services that communicate with each other and can be used in multiple separate systems.

In cloud computing particular attention is given by each provider in the sharing of computing resources. The **virtualization of resources** allows multiple operating system images to run on the same physical resources and physical and virtual resources to be pooled to serve multiple users according to user demand. The software between physical infrastructure (CPU, memory, network) and virtual machines which is responsible for the distribution of the available resources of the Cloud called is called a Hypervisor.

Cloud computing has attracted the interest of large and recognized in field of technology companies. Each offers unique services and infrastructure based on various architectures. Some of the most important providers are Amazon (Amazon Web Services), Microsoft (Microsoft Azure), IBM (IBM SmartCloud) and FIWARE.

1.2. Motivation

People have always sought to make their lives easier by finding more and simpler ways to control the environment in which the live.

With the demand for more time in our lives constantly increasing, the need to minimize the time and energy spent on various aspects of everyday life is even greater. The ability to control our devices and appliances in the easiest possible way and to utilize them as efficiently as possible is vital to this endeavor.

Also, people always seek ways to be as comfortable as possible. To have control over factors such as room temperature, humidity, lighting etc. is to have control over the level of comfort we experience.

As humans we spend most of our time in our homes. That is why we have always strived to find ways to make our stay there as comfortable, effortless and safe as possible. Achieving this is one of our main concerns in life and that show the importance of this need.

So, over the years, a number of different technologies of home control and automation have emerged.

Currently, with a multitude of devices, home appliances and sensors available that support wireless connectivity and a wide range of control options, controlling the environment we live in has become much easier and more efficient than ever before.

1.3. Proposed Solution

The solution we are proposing to the needs mentioned above is a smart home management system on the cloud. It is a service available on the cloud, which means it can be accessed anywhere and anytime by any device with an internet connection and a web browser.

Through this service the user can create an account. After creating and logging in to his account he can register his home by registering all the wirelessly controllable devices that are connected to his home's main access point ¹. After doing so he will have control over all the registered devices through this platform and will also be able to set rules for the home automation in the case he wishes to have certain aspects of his home controlled automatically.

The functions offered by this service for each specific home are :

- View/Control Devices
- Insert/Delete/Modify Device
- View Home Users
- Insert/Delete User
- Limited/Unlimited user control rights.
- Enable/Disable Home Automation
- Modify Home Automation Settings
- View/Delete Device Control History
- View/Delete Device Notifications

The advantages of this system are :

- It is a universal control platform for all home devices, so there is no need to access different platforms in order to control each device.
- Any new device can be registered and controlled on the system after it has been connected to the access point.
- It gives users the ability to monitor device operation.
- User is notified about important device updates, such as the completion of a device's set operation, keeping the user up to date.
- It provides information about power consumption, aiding in consumption management.
- The quicker and easier control of devices in conjunction with power consumption information and home automation leads to saving money on energy bills.
- The Home Automation option reduces effort needed to control certain aspects of the home and saves the users' time.
- Limited system functionality can be set for certain users, such as children, reducing device control options and system management, which increases security.

The fact that the system is on the cloud also provides the following advantages :

- **Ease of access.** Users can access the service from any place at any time, as long as an internet connection is available.
- **Elasticity.** If a user needs to add or remove a large number of devices, he can acquire or release resources to support his increased or decreased needs.
- **Pay per use.** Users are charged based on the use of the system and the resources they are using. This way they know how much they are paying and can manage it according to their needs.

¹ https://en.wikipedia.org/wiki/Wireless_access_point

- **Reduced cost.** Users do not need to purchase equipment and aren't responsible for equipment upgrades and maintenance, which reduces cost greatly. Also, they are not left with unnecessary equipment once they decide they no longer need resources or the service entirely.

1.4. Structure

In the 1st chapter we gave the definition of the Internet of Things, the new trend on the Internet, and cloud computing, analyzed its strengths and weaknesses and mentioned some of the most important cloud computing providers.

We then analyzed the motivation behind our work and then proceed to analyze our proposed solution, its main functions and the advantages it offers.

In the 2nd chapter we mention previous work on Smart homes and Home automation and existing technologies.

We describe the Internet of Things, the new trend on the Internet, what it offers us, its direct association with the cloud and how it works and the devices and sensors that compose it.

We elaborate further on Cloud Computing technology. We give its definition and analyze the strengths and weaknesses of Cloud technology and what it offers to the world of technology. We make an analysis of the service models on the cloud, what general and specific cause services are.

Furthermore we refer to Service-Oriented Architecture and the technologies we use in this thesis (REST, HTTP, JSON, etc.).

In the 3rd chapter we present the system's architecture, its individual parts at a sensor layer (Front end), system management and user interface level (Back End). We analyze the system's data flow and the general purpose services (Generic Enablers) which were implemented as part of the system.

In the 4th chapter we present the conclusions we came to by working on this system, the limitations on our work on the implementation of the system and suggestions about future work that can be done on it.

2. Background and Existing Technologies

2.1. Smart Home - Previous Work

Smart Homes are an idea that has existed for many years now. Naturally, quite a few technologies have been developed to realize it. Below we list a few.

X10

X10 appeared in the mid 1970's. It started out as a power line-based system (meaning it was hardwired into the walls), but eventually became wireless. It has slow speeds and doesn't support for communication between devices well either.

Insteon

Insteon is a home automation protocol designed to bridge the gap between power line-based and wireless protocols, so it uses both. It's also compatible with x10 devices, making it a good option for homes already equipped with X10 devices.

Z-Wave

Z-Wave is a wireless home automation protocol that runs on the 908.42MHz frequency band. It's relatively new in terms of home automation protocols, but has grown quite rapidly in the past few years. It utilizes a mesh network, which means that one Z-Wave product will pass the signal along to another until it reaches its intended destination. This system greatly extends its range and also reduces power consumption, making it extremely low power, which is ideal for devices that rely on battery power.

Zigbee

ZigBee is an 802 wireless communication standard built by the IEEE that runs on the 2.4 GHz frequency band. It's seen significant growth in the past few years, and can be found in a relatively large number of devices. It uses a star and mesh network structure, offering extended range, fast communication between devices and consuming a very small amount of power.

Wi-Fi

Wi-Fi is the most common wireless communication technology, a reason why a broad range of manufacturers have begun making smart home devices that work with it. It runs on the 2.4 GHz frequency band. Devices connect directly to the wireless router, removing the need for extra equipment. It is power demanding though, so it's not ideal for battery-based smart devices. Also, in a home with many Wi-Fi-connected devices interference and bandwidth become issues as devices compete for bandwidth, which could make them slower to respond.

Bluetooth and BLE

Bluetooth and BLE are wireless communication technologies that run on the 2.4 GHz frequency band. BLE is short for Bluetooth Low Energy. There many devices with Bluetooth or BLE integrated into them. It is used in home automation but usually not as the main protocol. Bluetooth and BLE don't consume little power (as the name suggests, BLE consumes even less), but also have a fairly limited range and lower communication speeds compared to other networking protocols (BLE's are even smaller), so it isn't recommended for devices that are required to be constantly connected, like security systems and motion sensors.

2.2. Internet of Things

2.2.1. Definition

As mentioned in chapter 1, The term The Internet of Things (IoT) refers to the network of physical objects, or “things”, embedded with electronics, software and sensors that feature an IP address for internet connectivity which enables communication between these objects and other Internet-enabled devices and systems to collect and exchange data, also allowing them to be sensed and controlled remotely.

Over the past years, a plethora of these objects has entered our lives. Environmental monitoring, medical and healthcare systems, building and home automation, manufacturing, transportation are only a few of the many aspects in which they find application.

2.2.2. Smart Devices

A smart device ^[3] is an electronic device, generally connected to other devices or networks via different wireless protocols (such as Bluetooth, Wi-Fi, etc.) that can operate, to some extent, interactively and autonomously. In recent years, more and more of the devices around us have acquired “smart” capabilities. Even in our homes, a large percentage of devices and appliances are now offered with such capabilities. Smart Televisions, air-conditioners, even fridges and washing machines now come with Wi-Fi connectivity and support wireless and inter-device communication and control. Furthermore, for devices that do not have such capabilities but have simple functions, such as light switches, garage doors etc., devices are available which can be connected to them and offer control over their functions (on/off, dim etc.). There has never been a time when the idea of a Smart Home has been more feasible and realistic.

2.2.3. Sensors

In recent years a large variety of sensors has entered our everyday life and find wide application, as much in production as in our personal life. These sensors are used in order to gather information from the environment and human activity and then, via the internet, for specific systems to store and process the data produced. Sensors can now be found everywhere since they are integrated on a series of devices used every day, such as smartphones, cars as well as in applications found in medicine, industry, robotics etc. The measurements taken are of many kinds, from the movement of the human body and cardiac function, to humidity and acceleration.

These sensor come to set the foundations of the Internet of Things. The trend is that all of our tools, and not only our computers, but our personal devices, cars and even power grids, communicate with each other and so set a new form of information organization for the production of services and products. The idea of the Internet of Things is based on these two major technological developments, the digitization of more and more devices-machines and the production of a large amount of data.

2.2.4. Internet of Things and Cloud Computing

Cloud computing is called upon to solve the problem of management and storage of large amounts of data that are produced by these devices. As analyzed previously, the benefits offered by cloud computing are multiple, in a way that means these two concepts could be nothing other than directly linked. "Smart" applications developed within the framework of a "smart" city, should have the possibility of flexibility and elasticity advantages only cloud computing technology currently offers. Also, the pay-per-use ability offered to users makes it even more attractive for use in such applications, as a device-sensor can send data at certain times of a day. For example, if a smart home device is set to perform a specific task by, the user can be notified once the device completes the task or if a problem occurs and not every time the device generates information.

Many companies engaged in providing cloud computing services have already begun to develop special tools for storing, processing and analyzing data from IoT devices. The development of specific services in Cloud Computing environments for the management of Internet of Things devices is already in development and these services are readily available to software developers to create innovative applications. Start-ups in the IT field, in particular, have the greatest need of the technology offered by providers through SaaS (Software as a Service) and PaaS (Platform as a Service) for Internet of Things devices, because of its low cost.

2.3. Cloud Computing

2.3.1 Definition

As mentioned in chapter 1, Cloud Computing is a model which enables universal, convenient, on-demand access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services), available rapidly to the user through little management effort or interaction with the service provider.

This cloud model is composed of three service models, and four deployment models.

2.3.2. Service Models

The three service models of Cloud computing are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

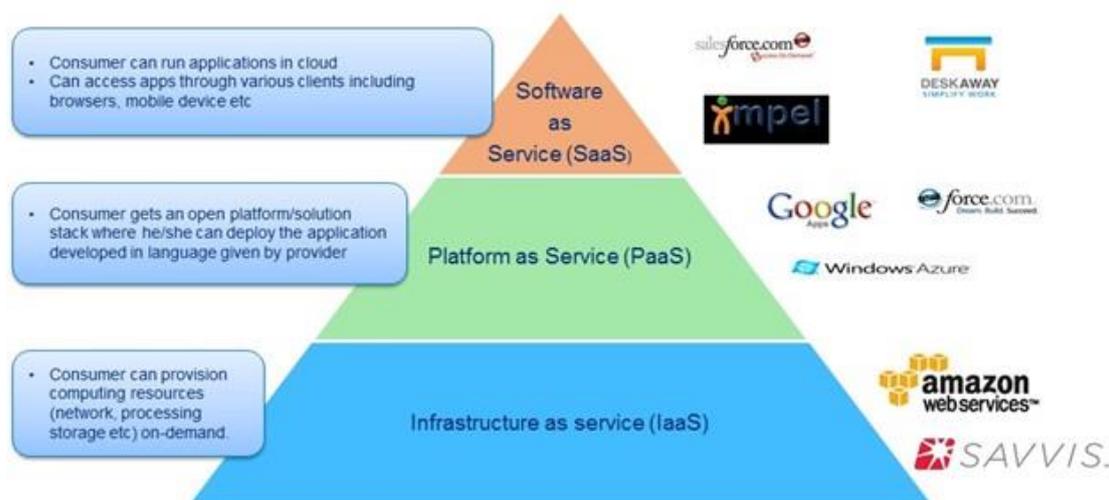


Figure 2 : Cloud computing service models

Software as a Service (SaaS). The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings. Examples of SaaS applications are Gmail, Dropbox, Yahoo Mail, etc. By acquiring such an application consumers only pay a fee once and support and updates for the application are provided by the service provider, reducing cost for consumers. There is also the ability to pay a subscription for the service, so consumers can be charged only for as long as they utilize the service. Finally there is no need for the use of the consumer's computing resources as access through a web browser doesn't require the installation of any additional software.

Platform as a Service (PaaS). The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications, created using programming languages, libraries, services, and tools supported by the provider, in a platform interface characterized by the elasticity and flexibility it offers. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and, possibly, configuration settings for the application-hosting environment. Consumers don't need to worry about maintenance of the software and the platform either, as they are the provider's responsibility. Major companies that offer PaaS are Microsoft (Azure) and Google.

Infrastructure as a Service (IaaS). The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications, on virtual machines. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls). Also, users can use the infrastructure according to their needs at that time, allowing them to pay only for as long as it is needed, eliminating the need to purchase equipment (computing, networking, etc.) and reducing cost.

2.3.3. Deployment Models

The four deployment models of the cloud computing model are Private cloud, Public cloud, Community cloud and Hybrid cloud.

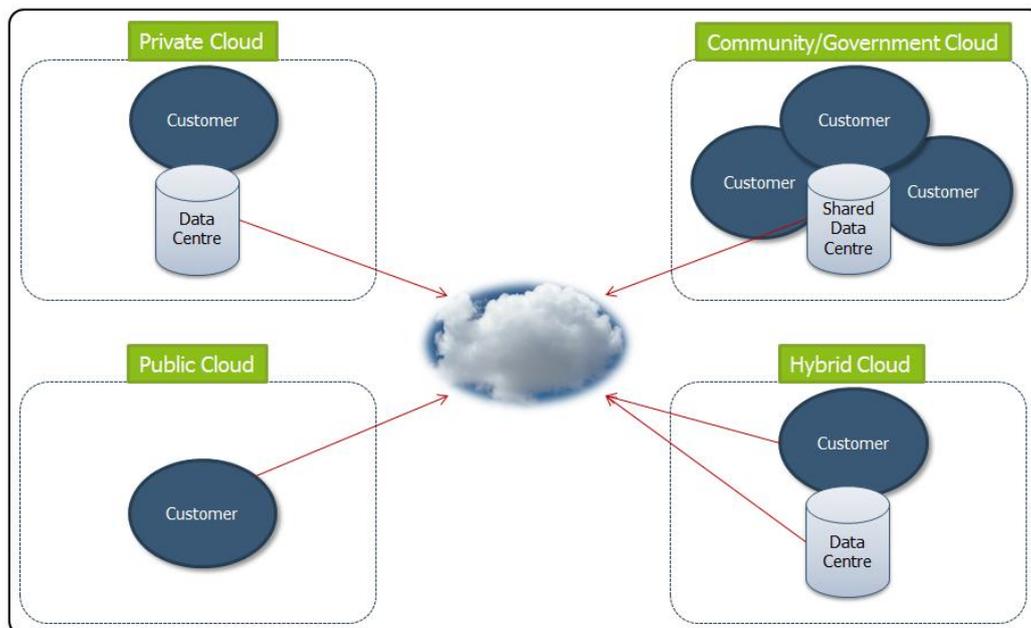


Figure 3 : Cloud computing deployment models

Private cloud. The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises. Private cloud solutions are adopted by companies that desire full control over services, which follow a common legal framework to ensure their security. Large implementation and running costs are characteristic of this model.

Public cloud. The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider. Charging via a pay-per-use model, greater flexibility due to direct provision of services and scalability, in a larger or smaller scale, according to user needs are the main advantages of this model.

Community cloud. The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations) and needs. It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

Hybrid cloud. The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds). The use of this model provides, in certain cases, greater flexibility to businesses due to the variety of resource combinations possible. These combinations allow for a more effective management of workload and large computing resources demand.

2.3.4. Cloud Computing Providers

Cloud computing has attracted the interest of large and recognized in field of technology companies. Each offers unique services and infrastructure based on various architectures. Some of the most important providers are mentioned below.

Amazon

Amazon Web Services ^[4] (AWS), is a collection of remote computing services, also called web services, that make up a cloud-computing platform offered by Amazon.com. It provides IaaS (Infrastructure as a Service) services. The most central and well-known of these services are Amazon Elastic Compute Cloud, which allows users to rent virtual computers on which to run their own computer applications, and Amazon Simple Storage Service, which is an online file storage web service that provides storage through web services interfaces.

IBM SmartCloud

IBM SmartCloud ^[5] offers SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) services through public, private and hybrid cloud deployment models, in addition to the components that make up those clouds. Its infrastructure's architecture is based on Openstack.

Microsoft Azure

Microsoft Azure ^[6] is a cloud computing platform and infrastructure created by Microsoft, for building, deploying and managing applications and services through a global network of Microsoft-managed and Microsoft partner hosted datacenters. It provides both PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) services and supports many different programming languages, tools and frameworks.

FIWARE Lab

FIWARE is a non-commercial platform that offers general purpose (GEs) and special purpose services (SEs) which come with simple application programming interfaces (APIs) to enable the implementation of "smart" applications. General purpose services (Generic Enablers) are provided by cloud computing infrastructure as PaaS (Platform as a Service) for application development. In addition, it enables the user to acquire the required expertise to become familiar with the services provided as well as experiments and results of these services by users who have already used them. These services are distributed free of charge and are public so that everyone can have access to them. This makes the platform attractive to users who want to create their own applications. FiWare also allows developers to obtain services as infrastructure (IaaS), so that they can create virtual machines and allocate computing resources (memory, processing power, storage space).

2.3.5. Service-Oriented Architecture

Service-oriented architecture ^[7], upon which our system was based, is based on the logic that any larger problem can be solved optimally and its complexity addressed effectively by dividing it into smaller problems which compose it. The same applies to systems based on web services, such as ours. This architecture is a flexible set of design principles and technology used in the development stage of systems. Such systems consist of a set of services that communicate with each other and can be used in multiple separate systems by different business sectors.

Some of the advantages that have made it particularly well-known and attractive are :

- Services are reusable and can be made available on a large scale.
- Faster and more efficient debugging.
- Shorter distribution time for new products and applications.
- Services are not bound by the system, but can be replaced.
- In an existing system the inclusion and integration of a new service does not require changes to the operating mechanism of the service.

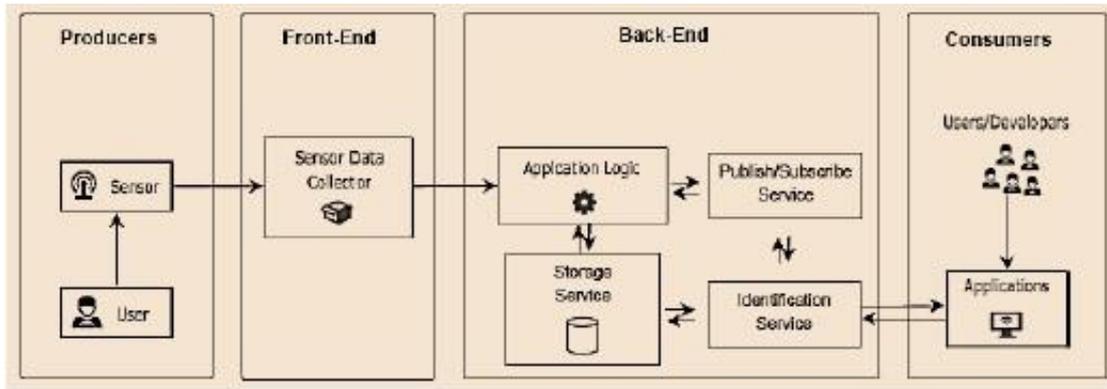


Figure 4 : IoT Specialized Service-Oriented Architecture

A specific instance of Service-Oriented Architecture for the case of Internet of Things applications, upon which our system was based, is displayed in Figure 4. It is divided into four parts that interact in the implementation of an application within the Internet of Things. On the left are the Producers, which include the sensors that produce the information and can interact with users. The Front-End includes the service which plays the role of a gateway between the data sent by the sensors and the data managed by the application through the Back-End. The Back-End includes all general purpose services for user authentication, subscription creation and data storage of the application. All these services communicate with the Application Logic, which makes use of rules, checks and conditions for the transmission of information to the individual services. Finally, the Consumers include either end users or other applications that communicate with the application programming interface (API) provided.

2.3.6. Existing Technologies

2.3.6.1. FiWare and Intellicloud

FIWARE is a non-commercial platform that offers general purpose (Generic Enablers - GEs) and special purpose services (Specific Enablers - SEs) followed by simple application programming interfaces (APIs) to enable the implementation of "smart" applications. Generic Enablers are provided by cloud computing infrastructure as PaaS for application development. The combination of such services constitutes a Specific Enabler, which contributes to solving a more complex problem. Moreover, it enables the user to acquire the required expertise to become familiar with the services as tests and results of these services by users who have already used them are provided. Services are distributed free of charge and are public so that everyone has access to them. This makes the platform attractive to users who want to create their own applications. The benefits of FIWARE don't stop at the distribution of services in the form of software, but also allows developers to obtain infrastructure as a service (IaaS), creating virtual machines and allocating computing resources (memory, processing power, storage space).

The Intellicloud cloud infrastructure was designed, implemented and maintained by the Intelligent Systems laboratory of the Technical University of Crete. The purpose of this infrastructure is to provide computing resources for application development on the cloud. The infrastructure is hosted entirely at the Technical University of Crete and currently includes 128 processing cores, 284 GB of RAM and 12 TB of hard drive storage. The software responsible for the operation of Intellicloud is based on Openstack Grizzly.

2.3.6.2. Generic & Specific Enablers

Generic Enablers are nothing other than common purpose services shared online as PaaS from cloud providers for the creation of applications. The architecture such services abide to and to whose rules they comply with, is REST followed by an API that allows access to them. The basic characteristics of a Generic Enabler are its simplicity and ease of use at any time requested by the user. Such reusable services are very important on a Cloud as they facilitate programmers in developing more complex systems.

The connection and communication of more than one Generic Enabler (GE) creates a Specific Enabler (SE) with the goal to implement a more complex function. In the event that the developer of the Specific Enabler wishes to integrate a new service, it can be done very easily without having to interfere with individual parts/GEs that make up the SE. Another important advantage is the replacement of one individual service/GE if, for example, a new improved version of the service to be replaced has been made available. Thus the SE remains updated without having to change the whole system entirely.

Developers need not know how these enablers were implemented, with the application programming interface (API) of each service being the only thing needed to use them for the services they offer and integrate them into their system.

2.3.6.3. HTTP & REST

HTTP ^[8] (Hypertext Transfer Protocol) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred. HTTP is the main protocol used in Web browsers to transfer data between a server and a client. The four main methods defined by this protocol are of the type CRUD (Create - Read - Update - Delete) and are listed below :

- GET, when the client requests a specific resource from the server.
- POST, when the client sends data to the server, for example for insertion into a database.
- PUT, when the client sends data to an existing resource (resource), to renew its value.
- DELETE, when the client wants to delete a particular resource from the server.

These all occur on the client side when sending a request to the server for a particular method. On the server side, for every request asked to process, it responds to the client with a status code in order for him to know if, for example, the resource requested was found. These code numbers are divided into five broad categories:

- Informational - 1XX
- Successful - 2XX
- Redirection - 3XX
- Client Error - 4XX
- Server Error - 5XX

REST ^[9] is an architecture style that consists of rules and restrictions for the creation of software and services hosted on the internet. These specific services communicate via the HTTP protocol (Hypertext Transfer Protocol) using the methods GET, POST, PUT, DELETE, etc. used for requests from clients to servers on the Internet. In this model, client-server is based on the REST architecture. By using the methods, customers can start requests (request) to servers, which in turn return the appropriate response, usually encoded in JSON or XML format.

An **API** (Application Programming Interface) consists of a set of guidelines and programming standards for access to online services. Providers of services offer these so-called APIs to the public, so that developers can design applications based on them. In addition, APIs allow a program to interact with another without involving the user. For example, when a web application offers the user the ability to login with his Facebook account, the web application uses the API provided by Facebook for user identification. In other words, we assign the identification of the users of our application to an external service (Facebook, in this example) to provide additional security, reliability and convenience to the user.

2.3.6.4. JSON (JavaScript Object Notation)

JSON ^[10] (JavaScript Object Notation) is a text format for the serialization of structured data. It is derived from the object literals of JavaScript. JSON can represent four primitive types (strings, numbers, booleans, and null) and two structured types (objects and arrays). In modern web services the JSON data model is used as an alternative to XML (Extensible Markup Language), for communication between server and client or server to another server. The advantages in relation to XML is many, some of which are listed below:

- Analysis of a JSON file is much easier than that of an XML file.
- In XML unlike JSON, there is a lot of redundancy, requiring more processing as a result.
- In contrast to XML, it is legible to humans.

From the above, the superiority of the JSON description in terms of information exchange on the Web in recent years, in relation to XML, is clear.

2.3.7. Virtualization of Resources

In cloud computing particular attention is given by each provider in the sharing of computing resources, so that consumers remain satisfied with the services provided. The virtualization of resources allows multiple operating system images to run on the same physical resources and physical and virtual resources to be pooled to serve multiple users according to user demand. The software responsible for the distribution of the available resources of the Cloud called is called a Hypervisor ^[11].

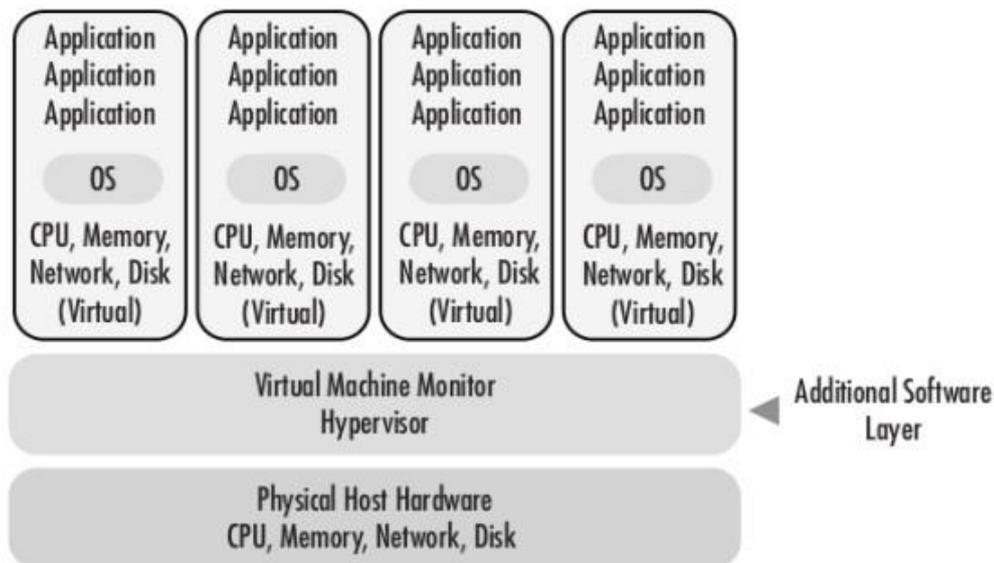


Figure 5 : Physical Infrastructure – Hypervisor – Virtual Machine Relation

The additional layer of software, visible in Figure 5, between physical infrastructure (CPU, memory, network) and virtual machines is referred to as Virtual Machine Monitor - Hypervisor. In essence it is an operating system that serves requests made by the virtual machines to the physical infrastructure for the use of specific computing resources.

Figure 5 models the interaction between the three entities of the Cloud Infrastructure that play a key role in customer service.

- Virtual machines are a set of virtual resources (virtual processing power, virtual memory, virtual storage, etc.), a guest operating system and applications running on this operating system.
- The Hypervisor, which distributes resources to the virtual machines according to their requirements.
- The Physical Resources constitute the actual processing power, memory, storage and network of the Cloud Infrastructure.

3. Design and Implementation

3.1. Architecture

The implementation of our system is based on a form of the Service-oriented architecture analyzed in chapter 2.3.4, which means it is composed of smaller sub-systems, or services. These systems, when properly programmed can communicate with each other and produce the information required. The components of this architecture are :

- **Front End :**
 - **Producers (Sensor Layer) :** The devices and sensors of the home that produce information and can be controlled through our application.
 - **Access Point :** The access point through which all the information produced by the devices and sensors goes through, as well as all the requests made by the users through the application and the responses of the devices and sensors.
- **Back End :** The main part of our system which is entirely on the Cloud and manages the data received from the devices and sensors and the user access and the requests they make.
- **Consumers (Users) :** The final component of the architecture which includes the users that access our system and the device and sensor information of their homes and control them as well as the User Interface through which they do this.

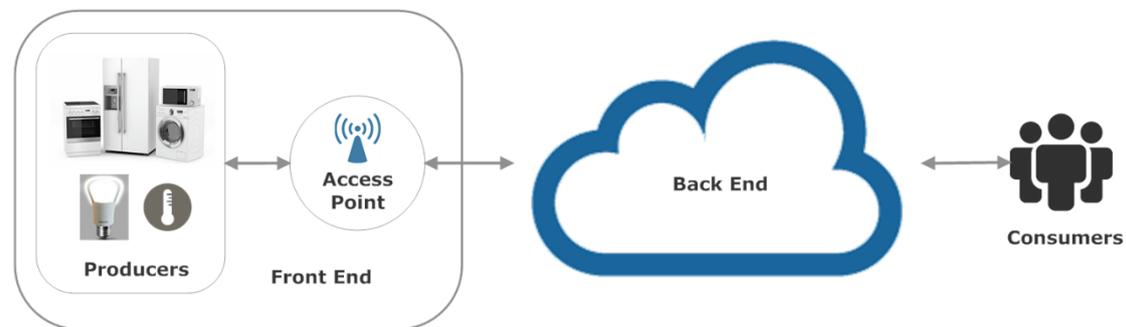


Figure 6 : General Architecture

3.1.1. Front End - Sensor Layer & Access Point

The Front End in our architecture is composed of two parts. The Sensor Layer and the Access Point.

The **Sensor Layer** consists of the devices and the sensors of each home which operate and produce information. Each device supports various functions (operation mode, operation program, etc.) and provides various information (device status, temperature, consumption, etc.) They are connected to the access point of the home through which communication with the system is enabled. Their operation can be monitored and controlled by the users through the application.

The **Access Point** is the device that facilitates communication between the devices and sensors with the system. All devices and sensors are connected to it. Whenever a device generates a notification, it sends it to the access point which in turn transmits it to the system. When a user requests information or makes a command, that request is sent to the access point which then transmits it to the desired device or devices. In the case that the Home Automation function is enabled, the system makes direct requests to the access point in the same way, without any involvement from the user.

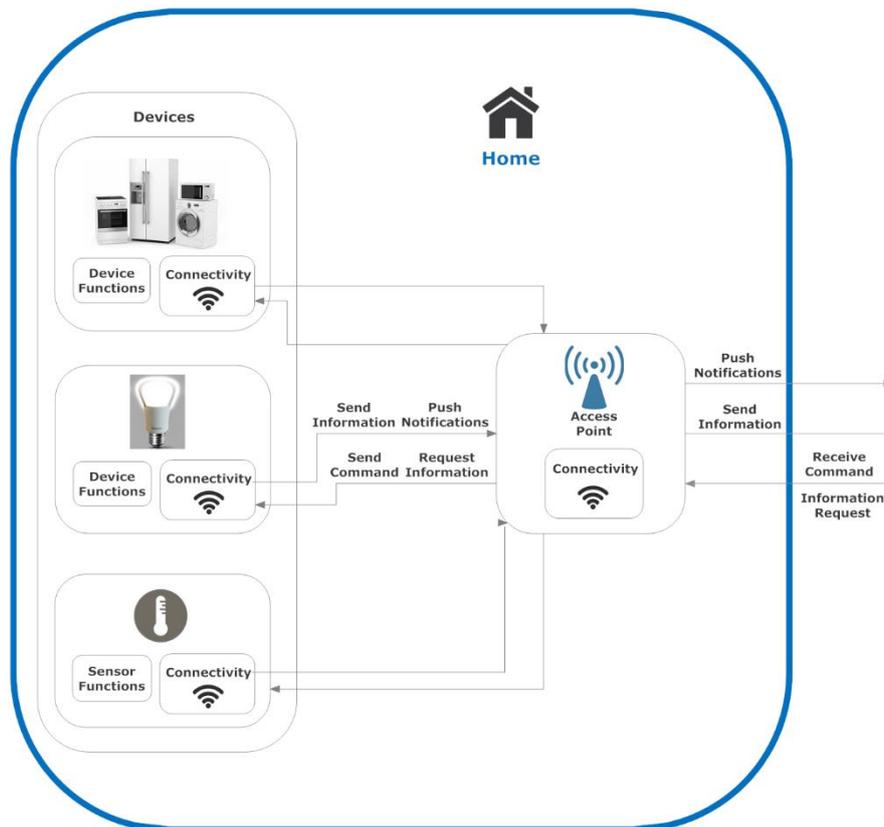


Figure 7 : Front End

3.1.2. Back End & Users

The **Back End** consists of Generic Enablers used for the various function of our system. Specifically the Keyrock Identity Management GE is used for the authentication of the users accessing the application, the Storage GE is responsible for storing and managing the user, device and home information and the Home Automation GE is responsible for managing certain functions of the home according to the preferences defined by the users. The Application is the central component of the system which utilizes the Generic Enablers mentioned above to manage the system's function and produce the desired results. The application gathers all the required information and provides it to the users, as well as all the functions that are available for the user to user to control through the User Interface which runs on the application.

Users can access the system through a web browser on any device with an internet connection. Each user is associated with one home which he can manage through the application, monitoring and controlling devices, receiving notifications and managing home users and settings.

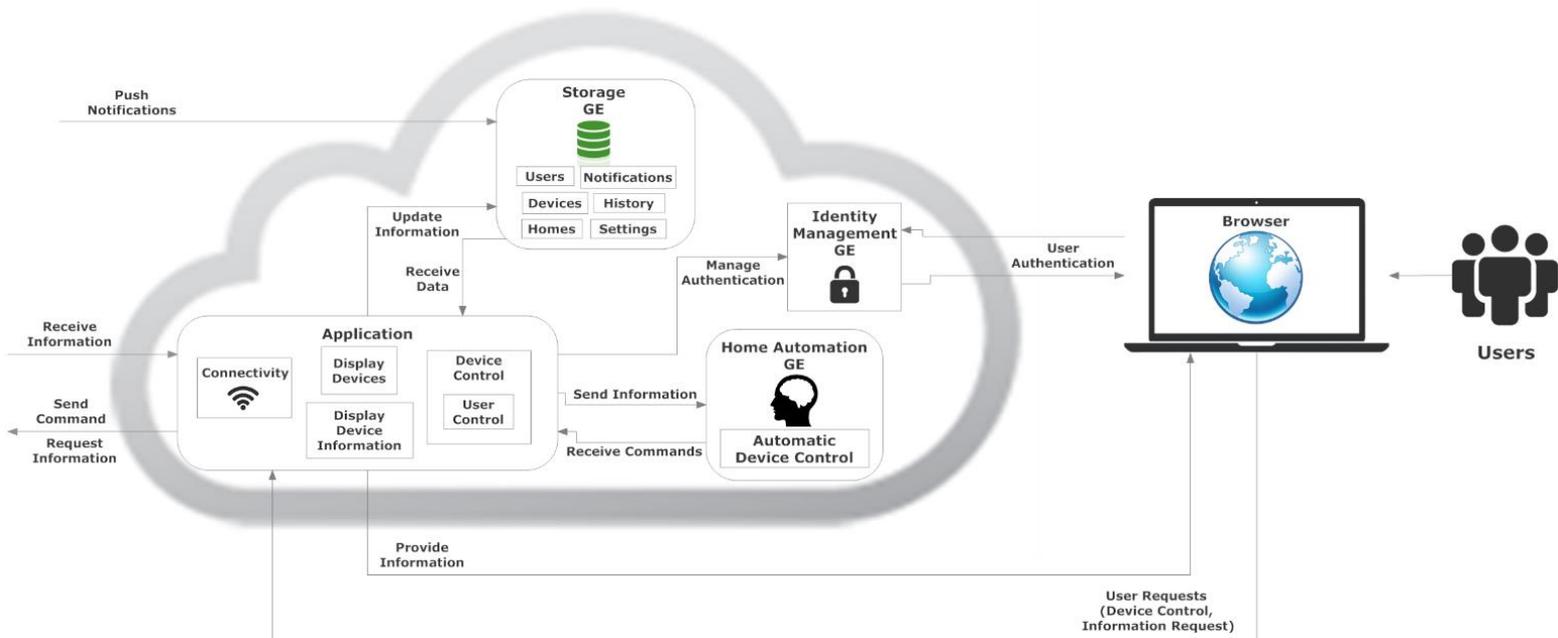


Figure 8 : Back End & Users

3.1.3. Data Flow

To better understand the use of the system we have to view the devices' functions and user's actions and the transfer of the required data. To do that we must create a data flow of each action, which is analyzed below :

1. Inserting a new Home into the system.

If a new user logs in to the system and hasn't been assigned to an existing home he must access the application and select the only available option, which is the "Create New Home" button, inserting his username, his user privileges and his FIWARE account e-mail and password, as well as a home id, port and ip address, into the available fields and submitting them.

2. Inserting a new User into the system.

To insert a new user the user must have full privileges. If he does he must select the "Insert New User" option on the application's main page, inserting a username, the user's privileges and the new user's FIWARE account e-mail and password into the available fields and submitting them.

The only way for a new user to be assigned to an existing home is if he is added by a full privileged user of the same home. This is done for security reasons.

3. View Home Devices.

To view the home's devices, the user simply has to access the application and all the home's devices will appear on the application's main page.

4. Inserting a new Device.

To insert a new device, the user must have full privileges. If he does, he must select the “Insert New Device” option on the application’s main page, insert the device’s id and type into the available fields and submit it. In order for this to have an actual effect the user must have connected a device with that id and of that type to the access point at his home. If not, the new device inserted on the system won’t correspond to an actual device and, thus, the user won’t be able to access or control it.

5. Deleting a Device.

To delete a devices, the user has to access the application and the home’s devices will appear on the application’s main page. If he has full privileges, a red “x” will appear under each device. He can simply click on the “x” button under the device he wishes to delete. A confirmation window will appear. If he selects “ok”, the device will be deleted.

6. View Device Information.

To view a device’s information, the user simply has to click on the device he wishes to view on the application’s main page. He will then be redirected to a page where the device’s information will appear. In order for information to appear, a device with that id and of that type must be connected to the access point at his home. The application will send a request to the home’s access point requesting information for the selected device. The access point receives the request and transmits it to the desired the device. The device then receives and processes the request, returning the desired data which it sends to the access point and the access point, in turn, to the application where it is processed and displayed.

7. Control Device.

To control a device, the user has to click on the device he wishes to control on the application’s main page. The user will be redirected to a page where the device’s information will appear. If the user has full privileges, he will be able to alter some of the visible fields. After making the changes he wishes, he must click on the submit button. If he wishes to only turn the device on or pause it (this function is available only for devices that support it), or turn it off, he can simply click on the start/pause or stop buttons respectively. After the user performs the action, the application will send a request to the home’s access point with the desired command for the selected device. The access point receives the request and transmits it to the desired the device. The device then receives and processes the request, performs the operations selected by the user and returns the device’s updated information which it sends to the access point and the access point, in turn, to the application where it is processed and displayed.

8. Enabling/Disabling Home Automation

To enable or disable the home automation function, the user must have full privileges. If he does, he can do so by simply clicking on the “Auto” button on the top part of the application’s main page. If the “Auto” button is gray, home automation is disabled, if it is green, then it is enabled.

9. Updating Home Automation Settings.

To update the home's home automation settings, the user must have full privileges. If he does, he must select the "Update Settings" option on the application's main page, select if he wants home automation enabled, what temperature he wants the home to be set at and whether it should be set for warm or cold and submit it.

10. Updating Home Data.

To update the home's data, the user must have full privileges. If he does, he must select the "Update Home" option on the application's main page, insert a port number and an ip address and submit it.

11. View Home Users.

To view the home's users, the user must select the "View Home Users" option on the application's main page. He will then be redirected to a page where all the home's users are listed.

12. Delete User.

To delete a user, the user must select the "View Home Users" option on the application's main page. He will then be redirected to a page where all the home's users are listed. If he has full privileges, a red "x" will appear next to each user. He can simply click on the "x" button next to the user he wishes to delete. A confirmation window will appear. If he selects "ok", the device will be deleted.

13. View Device control History.

To view the home's devices' control history, the user must select the "View Device History" option on the application's main page. He will then be redirected to a page where all the control operations performed on the home's devices are listed.

14. Clear Device control History.

To clear the home's devices' control history, the user must select the "View Device History" option on the application's main page. He will then be redirected to a page where all the control operations performed on the home's devices are listed. If he has full privileges, the option "Clear History" will be visible at the top right of the screen. By clicking it the entire device control history for the home will be erased.

15. Notification creation.

A notification is created when a device completes its operation. When that happens the device sends a notification to the access point which, in turn, pushes the notification to the system on the cloud, by posting the data to the system's Storage GE where it is stored.

16. View Device Notifications.

To view the home's devices' notifications, the user simply has to access the application. If there are any, the notification icon on the top right of the application's main page will be red. If not, it will be gray. Assuming the icon is red, by clicking on it a list of notifications will appear. The notifications consist of the id of the device that sent it followed by the notification's text.

17. Delete a Device Notification.

To delete the home's devices' notifications, the user simply has to access the application. If there are any, the notification icon on the top right of the application's main page will be red. If not, it will be gray. Assuming the icon is red, by clicking on it a list of notifications will appear. The notifications consist of the id of the device that sent it followed by the notification's text. By clicking on the notification of a device, that notification, along with any other notification from the same device with the same content, will be deleted.

3.1.4. GEs & Application

3.1.4.1. Keyrock Identity Management GE

The Keyrock Identity Management (Keyrock IDM) GE ^[12] is a Generic Enabler provided by FIWARE lab. It covers a number of aspects involving users' access to networks, services and applications, including secure and private authentication from users to devices, networks and services, authorization & trust management, user profile management, privacy-preserving disposition of personal data, Single Sign-On (SSO) to service domains and Identity Federation towards applications. The Identity Manager is the central component that provides a bridge between identity management systems at connectivity-level and application-level. Furthermore, Identity Management is used for authorizing foreign services to access personal data stored in a secure environment, provided the owner of the data gives consent to access the data; which also implies certain user authentication.

In our system we utilize Keyrock Identity Management for the authentication of the users, who must have an FIWARE account. Keyrock IDM provides security regarding the users' personal data, it uses the https protocol for the secure transmission of user data (username, password, etc.) and provides an API for application developers to use, which are the main reasons we have chosen to use it.

3.1.4.2. Storage GE

Storage GE is a service implemented for the needs of our system for data storage in a MySQL ¹ relational type database. This service is based on REST architecture and supports the insertion, deletion and updating of users, devices, homes, home settings and device control history and notifications. Each user can only access and modify the data associated to his home.

¹ <https://en.wikipedia.org/wiki/MySQL>

The basic methods supported by this service are presented in the following table :

| Method | URL (Base:147.27.50.169/Database_Access_GE) | Data (JSON Object) | Action |
|--------|--|---|---|
| GET | /users/{home} | - | Retrieve users of home='home' |
| GET | /users/{username} | - | Retrieve user with username='username' |
| GET | /users/{username, home} | - | Retrieve user with username='username' of home='home' |
| POST | /users | {username, email, password, home, privileges} | Create new user with provided data |
| DELETE | /users | {username} | Delete user with username='username' |
| GET | /devices/{id} | - | Retrieve devices of home='home' |
| GET | /devices/{home} | - | Retrieve device with id='id' |
| GET | /devices/{id, home} | - | Retrieve device with id='id' of home='home' |
| POST | /devices | {id, type, home} | Create new device with provided data |
| DELETE | /devices | {id, home} | Delete device with id='id' of home='home' |
| GET | /homes/{home} | - | Retrieve home with home id='home' |
| POST | /homes | {home, port, ip address} | Create new home with provided data |
| UPDATE | /homes | {home, port, ip_address} | Update home with home id = 'home' with provided data |
| DELETE | /homes | {home} | Delete home with home id = 'home' |
| GET | /settings/home | - | Retrieve settings for home='home' |
| POST | /settings | {activated, temperature, mode, home} | Create new settings for home with provided data |
| UPDATE | /settings | {activated, home} | Activate/Deactivate home automation for home with id='home' |
| UPDATE | /settings | {activated, temperature, mode, home, update} | Update settings for home='home' with provided data |
| GET | /history/id | - | Retrieve device history for device with id='id' |

| | | | |
|---------------|---------------------|---|--|
| GET | /history/username | - | Retrieve device history for device of user with username='username' |
| GET | /history/home | - | Retrieve device history for device of home='home' |
| POST | /history | {id, home, username, action, value, date} | Insert new event for device with id='id' of home='home' with provided data |
| DELETE | /history | {home} | Delete device history for home='home' |
| GET | /notifications/home | - | Retrieve notifications for home='home' |
| POST | /notifications | {text, home} | Insert new notification for home='home' with content='text' |
| DELETE | /notifications | {home, text, delete} | Delete notifications for home='home' |
| DELETE | /notifications | | Delete notifications for home='home' with content='text' |

Table 1 : Storage GE REST API

3.1.4.3. Home Automation GE

Home Automation GE is a service implemented for the needs of our system to enable and manage home automation capabilities. If the home automation function is enabled for a home, this service is provided with the home's current temperature and, after retrieving the home automation settings of the home, it provides a response to the system with a command (e.g. activate air conditioning, do nothing, etc.). This service is based on REST architecture, and currently responds to GET requests containing the home's id and temperature.

The home's temperature control automation is the only function that has currently been implemented, but other functions can be added to the existing system as parts of the service, simply using the existing architecture.

3.1.4.4. Application

The application is the central component of the system. It utilizes the services of the implemented GEs to carry out the system's operations. It is where the communication between the system and the access point is executed. The User Interface is located in the application, collecting and processing the required information to be presented to the user and enabling him to control the devices and operate the system in a simple and user friendly way.

In short, the application main part of the system which manages all the information and functions available to the user and implements the system's functionality by utilizing all other system components and enabling communication with the user's home.

3.2 System Implementation

3.2.1. Front End Implementation (Sensor Layer)

3.2.1.1. Device Implementation

Each home consists of devices and sensors. There is a wide range of “smart“ devices, appliances and sensors on the market. But the creation of a home with actual devices would require the purchase of a fair amount of them, which would be very costly, especially for just for the purpose of a thesis.

So, to better represent a complete smart home the decision was made to simulate such a home by implementing software programs that would each simulate a different device, its operation and its functions.

A series of programs were implemented to simulate the lights, air conditioners, ovens, washing machines, refrigerators and temperature sensors of the home.

These programs were written in Java and they all run on a single computer and they all share the same ip address. For the simulation of more than one device of the same type, different instances of the same program can be launched. Each program and each instance is assigned a different id and a different port numbers. Each program can have more than one port numbers assigned, depending on their communication requirements.

Below we analyze the operation of each program individually.

Light

The Light program is composed of three classes. Main, Light and Client_Thread.

The operations supported by the Light are: turning the device on and off and setting the percentage at which it operates (dimming the light). The information it can provide is its power state (on/off), the percentage it is operating at, the amount of time it has been operating, the amount of power it consumes (kW) and the total amount of power it has consumed (kW/h), as well as the device’s various values (id, power state, etc.).

The **Light** class contains all the device’s values (id, power state, operating time, etc.) and functions. Every time a function is executed, the operating time of the device is calculated by the `getOperating_time` function. Also, the functions that perform the operations named above are implemented in this class, as well as the `getInfo` function, which returns the sum of important device information.

In the **Main** class, an instance with a unique id and unique port number is created. An instance of the Light class is created with the given id and a client thread is initiated with the unique port number, for communication with the server thread running on the access point.

The main function then remains idle until a flag is raised from the client thread once a command is received. When this occurs, the command is processed and the appropriate function of the Light class instance is executed with the given parameters.

The result of the function is sent to the client thread and along with the raise of a flag to inform the thread a result was returned. The Main class's flag is lowered and the main function remains idle once again, until a new command is received.

The **Client_Thread** class is common for all devices, so we will only analyze it once here.

In the Client_Thread class, a socket is opened with the shared ip address and the unique port number set at the start of the program's execution when a thread of this class is initiated in the Main class.

Once it is opened, a connection is made to the access point through this socket. After this, the thread remains idle, waiting to receive data sent from the access point.

When this happens, the data is read and it is sent to the Main class along with a flag raise to inform it that a command has been sent. The thread then remains idle again until the Main class sends the return result and raises the thread's flag.

Then the return value is sent to the access point through the open socket, which then closes. The thread's execution then returns to the beginning where the socket is once again opened on the same port and it once again remains idle until new data from the access point is received.

The thread will stop execution if an exit command is received, right after the command is passed on to the main function where the entire program's execution will be terminated.

Fridge

The Fridge program is composed of three classes. Main, Fridge and Client_Thread.

The operations supported by the Fridge are: turning the device on and off, setting the level (out of 7 levels) at which the device will operate and setting the desired temperature. The information it can provide is its power state (on/off), the set temperature, the time remaining until it completes its operation, the amount of power it consumes (kW), which is dependent on the set operating level as well as the named consumption, and the total amount of power it has consumed (kW/h), as well as the device's various values (id, power state, etc.).

The **Fridge** class contains all the device's values (id, power state, operating time, etc.) and functions. Every time a function is executed, the operating time of the device is calculated by the getOperating_time function. Also, the functions that perform the operations named above are implemented in this class, as well as the getInfo function, which returns the sum of important device information.

The function of the **Main** class is identical to that of the Main class of the Light program analyzed above.

The **Client_Thread** class is common for all devices, and was analyzed as part of the Light program in this chapter.

Air Conditioner

The Air Conditioner program is composed of four classes. Main, Air_Conditioner, Client_Thread and Notification_Server_Thread.

The operations supported by the Air Conditioner are: turning the device on and off or pausing its operation, setting the program (warm, cold), setting the desired temperature, fan speed and vent position and, if the user wants to, setting the duration it should operate for before it turns off automatically. The information it can provide is its power state (on/off/paused), the set temperature, fan speed and vent position, the time remaining until it completes its operation, its thermal output (btu) the amount of power it consumes (kW) and the total amount of power it has consumed (kW/h), as well as the device's various values (id, power state, etc.).

The **Air_Conditioner** class contains all the device's values (id, power state, operating time, etc.) and functions. Every time a function is executed, the operating time of the device is calculated by the getOperating_time function. Also, the functions that perform the operations named above are implemented in this class, as well as the getInfo function, which returns the sum of important device information and the getRemaining_time function, which calculates the time remaining until the device completes its operation and, once it does, switches it off.

In the **Main** class, an instance with a unique id and three unique port numbers is created. An instance of the Air_Conditioner class is initiated with the given id, a client thread is initiated with the first port number, for communication with the server thread running on the access point, another client thread is created with the second port number through which the device will receive sensor temperature information from the access point and a server thread is initiated with the third port number, which sends notifications to the client thread running on the access point to be pushed from there to the Storage GE on the Cloud.

The main function then remains idle until a flag is raised from one of the client threads once a command or sensor information is received. When this occurs, the received data is processed and the appropriate function of the Air_Conditioner class instance is executed with the given parameters.

The result of the function is sent to the client thread along with the raise of a flag to inform the thread a result was returned. The Main class's flag is lowered and the main function remains idle once again, until a new command is received.

During the main function's operation, the remaining operation time of the device is checked. If an operation time was set and if the time remaining is now zero and the device has switched off, a notification is sent to the server thread and its flag is raised.

The **Notification_Server_Thread** class is common for all devices that support notifications, so we will only analyze it once here.

In the `Notification_Server_Thread` class, a server socket is opened with the unique port number set at the start of the program's execution when a thread of this class is initiated in the `Main` class.

Once it is opened, a connection is made to the access point through this socket. After this, the thread remains idle, waiting for a flag to be raised when a notification is received from the device.

When this happens, the notification is sent to the access point through the opened socket, the flag is lowered and the thread's execution then returns to the beginning where the socket is once again opened on the same port and it once again remains idle until a new notification is received from the access point.

The **Client_Thread** class is common for all devices, and was analyzed as part of the `Light` program in this chapter.

Washing Machine

The `Washing Machine` program is composed of four classes. `Main`, `Washing_Machine`, `Client_Thread` and `Notification_Server_Thread`.

The operations supported by the `Washing Machine` are: turning the device on and off or pausing its operation, setting the program (Synthetics, Wool, Eco, etc.) and setting the desired temperature. The information it can provide is its power state (on/off/paused), the set temperature, the time remaining until it completes its operation, the amount of power it consumes (kW), which is dependent on the program selected as well as the named consumption, and the total amount of power it has consumed (kW/h), as well as the device's various values (id, power state, etc.).

The **Washing_Machine** class contains all the device's values (id, power state, operating time, etc.) and functions. Every time a function is executed, the operating time of the device is calculated by the `getOperating_time` function. Also, the functions that perform the operations named above are implemented in this class, as well as the `getInfo` function, which returns the sum of important device information and the `getRemaining_time` function, which calculates the time remaining until the device completes its operation and, once it does, switches it off.

In the **Main** class, an instance with a unique id and two unique port numbers is created. An instance of the `Washing_Machine` class is initiated with the given id, a client thread is initiated with the first unique port number, for communication with the server thread running on the access point and a server thread is initiated with the second unique port number, which sends notifications to the client thread running on the access point to be pushed from there to the `Storage GE` on the `Cloud`.

The main function then remains idle until a flag is raised from the client thread once a command is received. When this occurs, the command is processed and the appropriate function of the Washing Machine class instance is executed with the given parameters.

The result of the function is sent to the client thread along with the raise of a flag to inform the thread a result was returned. The Main class's flag is lowered and the main function remains idle once again, until a new command is received.

During the main function's operation, the remaining operation time of the device is checked. If an operation time was set and if the time remaining is now zero and the device has switched off, a notification is sent to the server thread and its flag is raised.

The **Notification_Server_Thread** class is common for all devices that support notifications, and was analyzed as part of the Air Conditioner program in this chapter.

The **Client_Thread** class is common for all devices, and was analyzed as part of the Light program in this chapter.

Oven

The Oven program is composed of four classes. Main, Oven, Client_Thread and Notification_Server_Thread.

The operations supported by the Oven are: turning the device on and off or pausing its operation, setting the program (Up & Down Resistances, Fan, Grill, etc.) and setting the desired temperature. The information it can provide is its power state (on/off/paused), the set temperature, the time remaining until it completes its operation, the amount of power it consumes (kW), which is dependent on the program selected as well as the named consumption, and the total amount of power it has consumed (kW/h), as well as the device's various values (id, power state, etc.).

The **Oven** class contains all the device's values (id, power state, operating time, etc.) and functions. Every time a function is executed, the operating time of the device is calculated by the `getOperating_time` function. Also, the functions that perform the operations named above are implemented in this class, as well as the `getInfo` function, which returns the sum of important device information and the `getRemaining_time` function, which calculates the time remaining until the device completes its operation and, once it does, switches it off.

The function of the **Main** class is identical to that of the Main class of the Washing Machine program analyzed above.

The **Notification_Server_Thread** class is common for all devices that support notifications, and was analyzed as part of the Air Conditioner program in this chapter.

The **Client_Thread** class is common for all devices, and was analyzed as part of the Light program in this chapter.

Temperature Sensor

The Temperature Sensor program is composed of three classes. Main, Temperature_Sensor and Sensor_Server_Thread.

The operation supported by the Temperature Sensor is calculating the room's temperature and providing the resulting value.

The **Temperature_Sensor** class contains the sensor's values (id, current temperature) and functions. The class's main function, calculate_temperature, calculates the room's temperature through a simulated model which takes into account the heat output of the operating air conditioners, ovens and fridges as well as the outside temperature and taking into account factors such as the room's volume, air mass and the heat capacity of air constant and returns the result.

In the **Main** class, an instance with a unique id and a unique port number is created. An instance of the Temperature_Sensor class is initiated with the given id and a server thread is initiated with the port number, which sends the current temperature to the client thread running on the access point.

The main function then remains idle for ten seconds until it calculates the current temperature once more and sends the updated temperature to the server thread along with the raise of a flag to inform the thread a new temperature is available to send. The main function remains idle once again for 10 seconds and this process is repeated.

In the **Sensor_Server_Thread** class, a server socket is opened with the unique port number set at the start of the program's execution when a thread of this class is initiated in the Main class.

Once it is opened, a connection is made to the access point through this socket. After this the thread remains idle for ten seconds, waiting for a new temperature to be calculated. Once it is, it is sent to access point through the socket.

The thread then remains idle until the access point replies with information about the operating air conditioners, ovens and fridges as well as the total thermal output of the air conditioners and ovens in operation.

The information is processed and then sent to the Main class where the existing information is updated.

The thread's execution then returns to the beginning where the socket is once again opened on the same port and it once again remains idle for ten seconds when a new temperature will have been calculated.

3.2.1.2. Access Point Implementation

Following the creation of programs to simulate the home's devices, we created a program to simulate the access point of the home that implements communication with the devices and our system on the Cloud.

This program was written in Java and it runs on a different computer which is considered as the access point of the home.

The **Access Point** is composed of five classes. **Main**, **Server_Thread**, **PHP_Server_Thread**, **Push_Notifications_Thread** and **Sensor_Client_Thread**.

In the **Main** class, a server thread is created for each device of the home, a client thread for receiving sensor data is created for each sensor of the home, a server thread for communication with the application on the Cloud and a push notification thread for sending the notifications received from devices to the Storage GE of our system, each with a unique port number.

The main function then remains idle until a flag is raised from one of the threads once a device replies with the requested information or once a device notification or a command from the application on the Cloud is received.

When this occurs, the received data is processed and data is sent to the appropriate thread. Either a command from the application is sent to the server thread dedicated to the desired device, or a notification is sent to the push notification thread, or a response from a device is sent to the application on the Cloud and the flag is lowered.

The main function then returns to the beginning and if there is another raised flag the process above is repeated, otherwise it remains idle until a flag is raised.

The **Main** class also contains information about the devices' current power states which are updated regularly and are used to be sent to the temperature sensors whose temperature calculation factors in the operation of these devices.

In the **Server_Thread** class, a server socket is opened with the unique port number set at the start of the program's execution when a thread of this class is initiated in the **Main** class.

Once it is opened, a connection is made to the device through this socket. After this, the thread remains idle, waiting for a flag to be raised when a command is received for the device or when updated sensor information is available for the device.

If a command was received, a flag is raised and the command is sent to the device through the socket and the thread remains idle until a response is received from the device. When that happens, the response is sent to the main function for it to be sent to the **php_server_thread** and from there to the application, and the flag is lowered.

If updated sensor information is available, a sensor flag is raised and the information is sent to the device through the socket. The device then responds with its updated power state, which the thread then sends to the **Main** class where the devices' power states are stored and the sensor flag is then lowered.

The thread's execution then returns to the beginning where the socket is once again opened on the same port and if a flag is raised the process above is repeated, otherwise it once again remains idle until a new command or updated sensor information is received.

In the **PHP_Server_Thread** class, a server socket is opened with the unique port number set at the start of the program's execution when a thread of this class is initiated in the Main class.

Once it is opened, the thread remains idle until a connection is made with the application through this socket. Data is then received through the socket. The data is processed and must be verified. If the identity of the sender is to be confirmed, then the command received can be executed. This process was added to ensure security of the system.

After the data is processed, the command and its parameters are sent to the server thread to be sent to the appropriate device.

After this, the thread remains idle, waiting for a flag to be raised when a reply is received from the device. When this happens, the reply is sent to the application over the socket and the flag is lowered and the socket is closed.

The thread's execution then returns to the beginning where the socket is once again opened on the same port and it once again remains idle until a new command or updated sensor information is received.

In the **Push_Notifications_Thread** class a socket is opened with the unique port number set at the start of the program's execution when a thread of this class is initiated in the Main class.

Once it is opened, a connection is made to the access point through this socket. After this, the thread remains idle, waiting to receive a notification from the device.

When this happens the notification is sent to the sendPost function which processes the notification and sends it using a POST method of the HTTP protocol to send it to the Storage GE of our system.

The thread's execution then returns to the beginning where the socket is once again opened on the same port and it once again remains idle until a new notification is received from the device.

In the **Sensor_Client_Thread** class a socket is opened with the unique port number set at the start of the program's execution when a thread of this class is initiated in the Main class.

Once it is opened, a connection is made to the sensor through this socket. The thread then remains idle, waiting to receive data sent from the sensor.

When this happens, the data is processed and the included information is sent to the Main class along with a request for the current power state of the devices. The power states are returned and are sent to the sensor through the socket.

The thread's execution then returns to the beginning where the socket is once again opened on the same port and it once again remains idle until new data from the sensor is received.

3.2.1.3. Home registration and device installation

A home is defined by its access point and its devices. As we do not have an actual access point or devices, but only programs that simulate their functions, there is the need for minor customizations to the programs' code in order for them to operate correctly.

For every device that we want the home to have, a new thread has to be created in the access point with the variables of the device (id, port numbers). If we wish to add a new device, we must also alter the access point Main class code by adding the thread which corresponds to the device.

To create a device or an access point, a program must be executed. The access point and every device program can be exported as a .jar file which will run on the designated computer. Before exporting the program, though, the program's variables must be configured. So the device program's Main class's code must be altered with the desired device id and port numbers, which must correspond to the ones declared in the access point's code.

There is only the need to alter a few lines. The programs' code is parameterized so the alteration of the values will be transferred to the entire program.

Also, in order for a device which we declare in our system's application to be controllable, a program of the same type and with the same id as the declared device has to be launched. So the desired device program will be customized with the declared id and new ports and the creation of a thread will be added to the access point's program with the same ports and device id. The device will then be visible and completely controllable through the application.

Finally, for communication between the access point and the application to be possible, a specific port must be opened on the home's internet router. The port's number must be the one declared in the home's information stored in the database and should also match the port number declared in the access point program's code

3.2.2. Back End Implementation

3.2.2.1. Virtual machine creation on Intellicloud

Network Topology Creation

The network topology creation was created by defining an internal virtual network and virtual router so that communication with external networks can be accomplished.

This communication is based on rules and protocols defined by us based on the needs and requirements of the services that interact with our system. The topology is shown in the figure below.

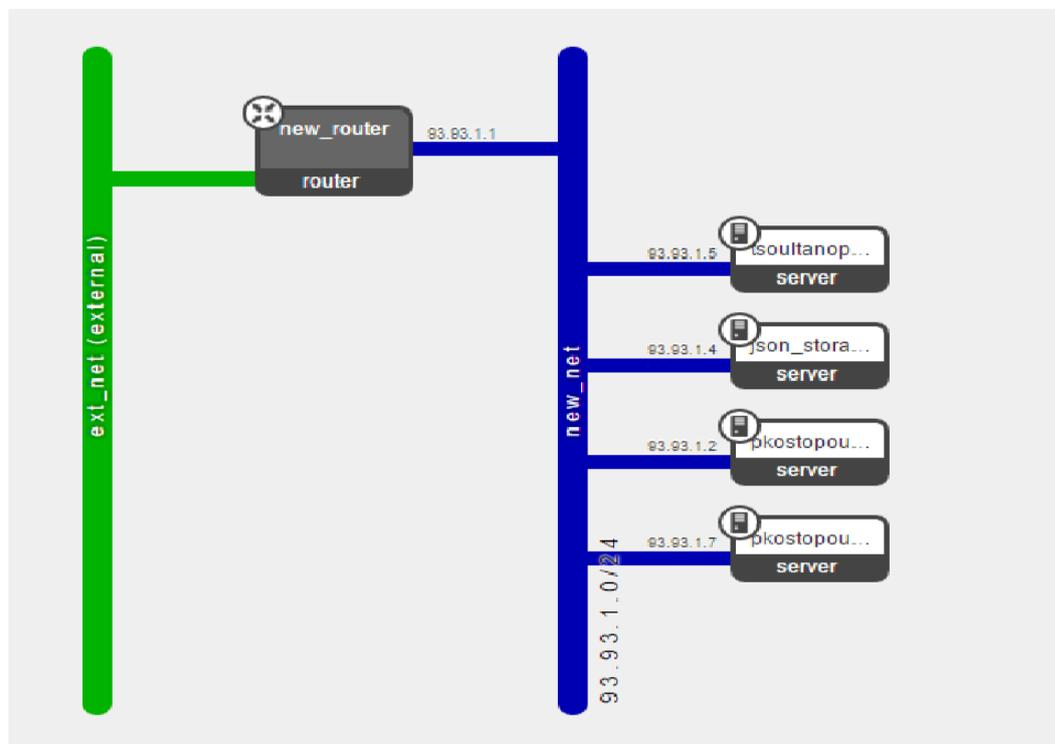


Figure 9 : Network Topology

Launching an Instance

By Launching an Instance we create an Image. We customized the Image based on our needs, selecting Ubuntu 12.04 LTS-64 bit as the operating system on a small size instance, allocating 1 processing core, 2 GB RAM and 20 GB hard disk storage. The instance's customization was completed by assigning it an ip address.

Launch Instance
✕

Details
Access & Security
Networking
Volume Options
Post-Creation

Instance Source

Image ▼

Image

Ubuntu12.04LTS-64 ▼

Instance Name

Flavor

m1.small ▼

Instance Count

1

Specify the details for launching an instance.

The chart below shows the resources used by this project in relation to the project's quotas.

Flavor Details

| | |
|-----------------------|----------|
| Name | m1.small |
| VCPUs | 1 |
| Root Disk | 20 GB |
| Ephemeral Disk | 0 GB |
| Total Disk | 20 GB |
| RAM | 2,048 MB |

Project Quotas

Number of Instances (4) 6 Available

Number of VCPUs (4) 16 Available

Total RAM (8,192 MB) 43,008 MB Available

Cancel

Launch

Figure 10 : Instance customization

Application Development Tools and Virtual Machine

After creating the virtual machine, we needed to install the tools needed for the implementation of our system. The PuTTY ¹ tool was used for installation and access to the images.

We installed Apache so that the virtual machine can offer server services, PHP libraries whose tools are used for server communication and making calls to the services and MySQL to create and manage the services' database. Additionally, cURL ² libraries were installed, for making HTTP requests in PHP.

¹ <https://en.wikipedia.org/wiki/PuTTY>

² <https://en.wikipedia.org/wiki/CURL>

3.2.2.2. Use of Keyrock Identity Management GE

For the user authentication of our system we use the Keyrock Identity Management GE which is available at the address <https://account.lab.fiware.org/>.

Initially we use the id and secret, unique to our application registered on FIWARE, to request a code from the page on account.lab.fiware.org/. This is accomplished by redirection to this page which includes the application's id, where the user inserts his username and password. After acquiring and processing this information, the page redirects to our application's page providing the requested code.

A POST request, containing the received code and the encoded id and secret used before, is then made to the Keyrock IdM GE,.

The response contains an access token. Finally, a GET request containing the token is made to the Keyrock IdM GE.

If the authentication is successful the authenticated user's information is returned, his username is added to the Session and he is redirected to the application's main page, which he now has access to.

3.2.2.3. Storage GE Implementation

The system's Storage GE runs on the virtual machine on the address http://147.27.50.169/Database_Access_GE/. The GE is composed of six .php pages: users, devices, homes, settings, history and notifications.

Each page handles HTTP requests made to it and provides the data requested as a reply. The function of each of these pages is presented below.

Users

Initially a connection is made to the system's sql database using PDO ¹.

Then the HTTP requests are handled. Five requests, one insertion, one deletion and three retrieval, are supported. If the requests are of the correct format and contain the required information. A function is then called depending on the request.

In each function makes a query of the database is made with the data contained in the request. The result of the query is returned and is then encoded in JSON format and sent as a response to the request.

The supported requests are listed in the table below :

| Method | URL (Base:147.27.50.169/Database_Access_GE) | Data (JSON Object) | Action |
|--------|--|-----------------------|--|
| GET | /users/{home} | - | Retrieve users of home='home' |
| GET | /users/{username} | - | Retrieve user with username='username' |

¹ <http://php.net/manual/en/intro.pdo.php>

| | | | |
|---------------|-------------------------|---|---|
| GET | /users/{username, home} | - | Retrieve user with username='username' of home='home' |
| POST | /users | {username, email, password, home, privileges} | Create new user with provided data |
| DELETE | /users | {username} | Delete user with username='username' |

Devices

As in the users page, initially a connection to the sql database is made.

The HTTP requests are then handled. Five requests, one insertion, one deletion and three retrieval, are supported.

A function is then called which queries the database with the provided data and returns the result, which is then encoded in JSON format and sent as a response to the request.

The supported requests are listed in the table below :

| Method | URL (Base:147.27.50.169/Database_Access_GE) | Data (JSON Object) | Action |
|---------------|--|-----------------------|---|
| GET | /devices/{id} | - | Retrieve devices of home='home' |
| GET | /devices/{home} | - | Retrieve device with id='id' |
| GET | /devices/{id, home} | - | Retrieve device with id='id' of home='home' |
| POST | /devices | {id, type, home} | Create new device with provided data |
| DELETE | /devices | {id, home} | Delete device with id='id' of home='home' |

Homes

As in the users page, initially a connection to the sql database is made.

The HTTP requests are then handled. Four requests, one insertion, one deletion one update and one retrieval, are supported.

A function is then called which queries the database with the provided data and returns the result, which is then encoded in JSON format and sent as a response to the request.

The supported requests are listed in the table below :

| Method | URL (Base:147.27.50.169/Database_Access_GE) | Data (JSON Object) | Action |
|------------|--|-----------------------|-----------------------------------|
| GET | /homes/{home} | - | Retrieve home with home id='home' |

| | | | |
|---------------|--------|----------------------------------|--|
| POST | /homes | {home, port, ip address} | Create new home with provided data |
| UPDATE | /homes | {home, port, ip_address, update} | Update home with home id = 'home' with provided data |
| DELETE | /homes | {home} | Delete home with home id = 'home' |

Settings

As in the users page, initially a connection to the sql database is made.

The HTTP requests are then handled. Four requests, one insertion, two update and one retrieval, are supported.

A function is then called which queries the database with the provided data and returns the result, which is then encoded in JSON format and sent as a response to the request.

The supported requests are listed in the table below :

| Method | URL (Base:147.27.50.169/Database_Access_GE) | Data (JSON Object) | Action |
|---------------|---|--|---|
| GET | /settings/home | - | Retrieve settings for home='home' |
| POST | /settings | {activated, temperature, mode, home} | Create new settings for home with provided data |
| UPDATE | /settings | {activated, home} | Activate/Deactivate home automation for home with id='home' |
| UPDATE | /settings | {activated, temperature, mode, home, update} | Update settings for home='home' |

History

As in the users page, initially a connection to the sql database is made.

The HTTP requests are then handled. Five requests, one insertion, one deletion and three retrieval, are supported.

A function is then called which queries the database with the provided data and returns the result, which is then encoded in JSON format and sent as a response to the request.

The supported requests are listed in the table below :

| Method | URL (Base:147.27.50.169/Database_Access_GE) | Data (JSON Object) | Action |
|---------------|---|------------------------------|---|
| GET | /history/id | - | Retrieve device history for device with id='id' |

| | | | |
|---------------|-------------------|---|--|
| GET | /history/username | - | Retrieve device history for device of user with username='username' |
| GET | /history/home | - | Retrieve device history for device of home='home' |
| POST | /history | {id, home, username, action, value, date} | Insert new event for device with id='id' of home='home' with provided data |
| DELETE | /history | {home} | Delete device history for home='home' |

Notifications

As in the users page, initially a connection to the sql database is made.

The HTTP requests are then handled. Four requests, one insertion, two deletion and one retrieval, are supported.

A function is then called which queries the database with the provided data and returns the result, which is then encoded in JSON format and sent as a response to the request.

The supported requests are listed in the table below :

| Method | URL (Base:147.27.50.169/Database_Access_GE) | Data (JSON Object) | Action |
|---------------|--|-----------------------|---|
| GET | /notifications/home | - | Retrieve notifications for home='home' |
| POST | /notifications | {text, home} | Insert new notification for home='home' with content='text' |
| DELETE | /notifications | {home, text, delete} | Delete notifications for home='home' |
| DELETE | /notifications | | Delete notifications for home='home' with content='text' |

3.2.2.4. Home Automation GE Implementation

The system's Home Automation GE runs on the virtual machine on the address http://147.27.50.169/Home_Automation_GE/. The GE is currently only composed of one .php page, air_conditioning.

This operates along the system's auto page (analyzed in chapter 3.2.2.5) to provide home automation functions.

The function of this page is presented below.

Air_conditioning

Initially the HTTP request is handled. To access the service a GET request containing the home's id and current temperature.

The home id is then used to retrieve the home's settings by making a request to the Storage GE. This is performed by calling a function which performs a GET request to the GE with the home id and returns the GE's response.

The current temperature provided from the GET request and the threshold temperature contained in the settings are compared and, depending on the mode (cold or warm) selected in the settings, decides whether the air conditioning should be activated or not and which mode they should operate at.

This command is then encoded in JSON format and sent as a response to the original GET request.

3.2.2.5. Application Implementation

The system's Application runs on the virtual machine on the address http://147.27.50.169/Smart_Home/. The Application is composed of sixteen .php pages.

The function of each of these pages is presented below.

Index

The index page contains a Login button which, when pressed, initiates the user authentication process.

If the user is already logged in, which means the session contains his username, he is redirected to the main page.

If the user tries to access any other page without having logged in, he is redirected to this page.

Login

The login page implements the process required to authenticate the user through the Keyrock Identity Management GE.

Our application's unique id and secret is stored in variables. If a code isn't stored in the session, a redirection which includes the application's id is made to the account.lab.fiware.org/ page, where the user has to insert his FIWARE account username and password. The page then redirects back to the login page, providing a code, which is then stored in a variable.

The id and secret are encoded and sent, along with the received code, in a POST request to the Keyrock IdM GE. If the request is successful, the GE's response, in JSON format, will contain an access token.

The response is decoded and a GET request containing the token is sent to the Keyrock IdM GE. If the request is successful a response, in JSON format, will be sent.

The response is decoded and the username contained is it is stored in the session. The page then redirects to the application's main page.

Logout

The logout page is accessed when the user clicks on the application's logout button. The session's data is then deleted and the page then redirects to the index page.

Main

At the beginning of the main page the session is initiated. If the user hasn't logged in, the main page redirects to the login page.

A function is called to obtain the user's home id and privileges by making a GET request to the system's Storage GE, using the username stored in the session.

The home id is then also stored in the session and three functions are called with it as a parameter. One to request the home's devices, one to request the home's notifications and one to request the home's home automation settings.

At the top of the page there a header is displayed where a link to the main page, a button to enable/disable the home automation and a link to logout are located.

The button's colour is decided by checking the home's automation settings retrieved earlier and setting the button's colour to green if it is enabled and gray if it is disabled. When the button is clicked, a function is called which uses which uses AJAX¹ to send an UPDATE request, without refreshing the page, to the system's Storage GE to update the home's settings, activating or deactivating the home automation

If the user is new and hasn't registered a home yet, nothing else appears except for an "Insert New Home" button which, when clicked, a function is called which displays a hidden div containing a form in which a home id, port and ip address are inserted and submitted to create a new home. The form information is submitted to the insert_home page which makes a POST request to the system's Storage GE with the provided data.

If the user isn't new and is already assigned to an existing home, the following content is displayed.

Bellow the header, the user's home id is displayed.

At the top right part of the page, beneath the header, there is a notification icon which, when clicked, displays all the home's notifications beneath it. If there are notifications the button is gray and, if there are, it is red. This result is accomplished by checking the notifications retrieved earlier and with one function which displays a hidden div where the retrieved notifications are displayed when then the notification icon is clicked. By clicking one of the displayed notifications, a function is called which uses AJAX to run the delete_notification page's script, without refreshing the page, to send a request to the Storage GE to delete the home's notification whose content is identical to the notification which was clicked.

¹ [https://en.wikipedia.org/wiki/Ajax_\(programming\)](https://en.wikipedia.org/wiki/Ajax_(programming))

The user's home's devices are displayed under that. For each type of device a different image is displayed. The device's id is displayed above each device and, if the user's privileges retrieved earlier are full (so users with limited privileges are unable to perform this action), a red "x" button is displayed under it, which, when clicked, calls a function which uses AJAX to delete that device by making a POST request to the system's Storage GE to delete the home's device whose id is identical to the device who's "x" button was clicked, without refreshing the page, after the user selects "ok" in the confirmation windows that appears.

The content after that is displayed only to users with full privileges.

An "Insert User" button is displayed. When this button is clicked, a function is called which displays a hidden div containing a form in which a username, e-mail address, password and privileges are inserted and submitted to create a new user of the home. The form information is submitted to the insert_user page which makes a POST request to the system's Storage GE with the provided data.

This is used for existing home users with full privileges to add new users to the home, which is the only way for new users to be added to an existing home, for security reasons.

Next to it an "An "Insert Device" button is displayed. When this button is clicked, a function is called which displays a hidden div containing a form in which a device id and device type are inserted and submitted to create a new device of the home. The form information is submitted to the insert_device page which makes a POST request to the system's Storage GE with the provided data.

Next to that, an "Update Home" button is displayed. When this button is clicked, a function is called which displays a hidden div containing a form in which a port number and ip address are inserted and submitted to update the home's information. The form information is submitted to the update_home page which makes an UPDATE request to the system's Storage GE with the provided data.

Next to it, an "Update Settings" button is displayed. When this button is clicked, a function is called which displays a hidden div containing a form in which a value which shows if the home's Home Automation should be enabled and its temperature and operation mode are inserted and submitted to update the home's home automation settings. The form information is submitted to the update_settings page which makes an UPDATE request to the system's Storage GE with the provided data.

After these, two more buttons are displayed next to them.

The View Home Users button which redirects the user to the view_users page where the home's users are displayed.

And the View Device History button which redirects the user to the view_history page where the home's devices operation history are displayed.

View Users

The view_users page is where user is redirected when the "View Home Users" button is clicked on the main page.

At first, a function is called to obtain all the users of the home the user is assigned to by making a GET request to the system's Storage GE, using the home id.

Then two functions are called. One to request the home's home automation settings for the header's button and one to request the user's privileges so it can be decided whether all the page's functions will be available to him, by making two GET requests to the system's Storage GE, using the home id and the username stored in the session.

Under the application's pages' header, the users of the home the user is assigned to are displayed. If the user has full privileges a red "x" icon is displayed next to each user. By clicking on it a confirmation windows appears. If "ok" is selected, a function is called which uses AJAX to run the delete_user page's script, without refreshing the page, to send a request to the Storage GE to delete the home's user whose id is identical to the user's who was clicked.

View History

The view_users page is where user is redirected when the "View History" button is clicked on the main page.

At first two functions are called. One to request the home's home automation settings for the header's button and one to request the user's privileges so it can be decided whether all the page's functions will be available to him, by making two GET requests to the system's Storage GE, using the home id and the username stored in the session.

If the user has full privileges a red "Clear History" button will be visible at the top right corner. When clicked a confirmation windows will appear. If "ok" is selected, a function which uses AJAX to clear the history by making a POST request to the system's Storage GE to delete the home's history whose id is identical to the user's home's, without refreshing the page.

View Device

The view_users page is where user is redirected when he clicks on a device's icon on the main page.

At first two functions are called. One to request the home's home automation settings for the header's button and one to request the home's port number and ip address, by making two GET requests to the system's Storage GE, using the home id and the username stored in the session.

The device's id and type are also retrieved from the GET data included in the page's url, as well as information as to whether a command was previously selected for the device and what the command was.

If a command had been selected, a function is called containing the command, its value and the home's port, ip address and id.

The function opens a socket and uses the ip address and port number to make a connection to the home's access point. If the connection is successful it sends the command and its value, along with the home id. Only if the home id sent matches the home id declared in the access point will the command be executed. This verification adds security. The function then exits, without waiting for a reply.

Another function is then called, which update's the home's device control history with this latest command and the date and time it was made on by making a POST request to the system's Storage GE to insert a new event to the home's history whose id is identical to the device's home's.

After that, a function is called which opens a socket and uses the ip address and port number to make a connection to the home's access point. If the connection is successful it sends a command, along with the home id, requesting information about the device. Only if the home id sent matches the home id declared in the access point will the command be executed. This verification adds security. The function then waits for the access point's reply. Once a reply is received, the function exits and returns the reply.

The reply is then processed to retrieve the information contained in it.

The device's id and image are displayed.

Underneath, two buttons are displayed. A start/pause and a stop button.

When the start/pause button is clicked, a command is sent to the device to begin operation, if it is paused or stopped, or to pause its operation, if it is already in operation and only if the device supports pausing. If it doesn't, the play/pause button simply sends a command for the device to begin operation.

When the stop button is clicked, a command is sent for the device to turn the device off. This also stops the device's program and clears any remaining time.

These buttons are actually links to the view_device page, but also contain the command and its value in the url, which are retrieved by the GET method and executed at the beginning, as we mentioned above.

Under that, the device's information, such as power state (on/off/paused), program, consumption, remaining operation time, etc., is displayed.

Certain of the displayed values, such as temperature, remaining operation time, etc, can be altered. The user simply has to select the desired value and click the submit button next to it. The command and its value is then submitted to the view_device page, which is retrieved and executed at the beginning, as we mentioned above.

If the home's home automation is enabled and the displayed device is an air conditioner, the user can't alter any of these values, or control its power state (on/off/paused), because the device's operation is controlled by the home automation.

Insert Home

The insert_home page is where the data is submitted by the "Insert New Home" button's form in the main page.

The home, port and ip address values submitted are inserted into an array. cURL commands are then used to make a POST request to the system's Storage GE to insert a home with the values listed in the array.

The username, email, password, privileges and home values submitted are inserted into a second array. cURL commands are then used to make a POST request to the system's Storage GE to insert a user with the values listed in the array to the system's database.

Insert Device

The insert_device page is where the data is submitted by the “Insert New Device” button’s form in the main page.

The device id, device type, and home values submitted are inserted into an array. cURL commands are then used to make a POST request to the system’s Storage GE to insert a device with the values listed in the array to the system’s database.

Update Homes

The update_home page is where the data is submitted by the “Update Home” button’s form in the main page.

The home, port and ip address values submitted, as well as a required update value, are inserted into an array. cURL commands are then used to make an UPDATE request to the system’s Storage GE to update the home with the values listed in the array.

Update Settings

The update_settings page is where the data is submitted by the “Update Settings” button’s form in the main page.

The temperature, mode and activated and home values submitted, as well as a required update value required, are inserted into an array. cURL commands are then used to make an UPDATE request to the system’s Storage GE to update the settings with the values listed in the array.

Delete User

The delete_user page is where the data is submitted by the button for deleting a user in the view_users page.

Delete Device

The delete_user page is where the data is submitted by the button for deleting a device in the main page.

Delete Notification

The delete_notification page is where the data is submitted by the button for deleting a device in the main page.

Auto

The execution of this page's script is added to our system's Cron ¹ scheduler, so it is executed automatically every minute.

Initially, the settings of all the homes registered on our system are retrieved by a function which makes a GET request to the system's Storage GE.

For every home whose home automation is enabled, two functions are called, one which makes a GET request to the system's Storage GE to retrieve their port and ip address and one which makes a GET request to the system's Storage GE to retrieve each home's devices.

A function is then called which opens a socket and uses the ip address and port number to make a connection to the home's access point. If the connection is successful it sends a command, along with the home id, requesting the home's temperature. Only if the home id sent matches the home id declared in the access point will the command be executed. This verification adds security. The function then waits for the access point's reply. Once a reply is received, the function exits and returns the reply.

The reply is processed and the temperature's value is received.

A function is then called which makes a GET request to the system's Home Automation GE, containing the home's id and temperature, to receive a command.

The command is processed and, for each of the home's air conditioner devices, a function is called with the command as a parameter.

The function opens a socket and uses the home's ip address and port number to make a connection to its access point. If the connection is successful it sends the command, along with the home id, to be executed by the device. Only if the home id sent matches the home id declared in the access point will the command be executed. This verification adds security. The function then exits without waiting for a reply.

After commands are sent to the air conditioner devices of all the homes that have home automation enabled, the page's function ends, until it is executed again.

3.2.2.6. User

The user can access the system through a web browser on any device with an internet connection.

As the system's application is implemented as .php pages, these pages are requested and sent to the web browser, which then displays them.

The system's User Interface is implemented as part of the login, main, view_device, view_users and view_history pages functions analyzed in chapter 3.2.2.5.

Through these pages the user can log into, access and interact with the system's application, from anywhere at any time, monitoring and controlling the devices of his home, as well as managing the home's other aspects, such as its users.

¹ <https://en.wikipedia.org/wiki/Cron>

3.2.3. Use of the Application (UI Examples)

In this chapter we are going to present the User Interface of our application in certain use case scenarios.

Use Case 1 – A new User entering the Application

The user enters the application's url, which is http://147.27.50.169/Smart_Home/. The following page is displayed :



Figure 11 : Index Page

When the user clicks on the “Login” button, the login process is initiated and he is redirected to the following page where he must insert his FIWARE account's e-mail and password.

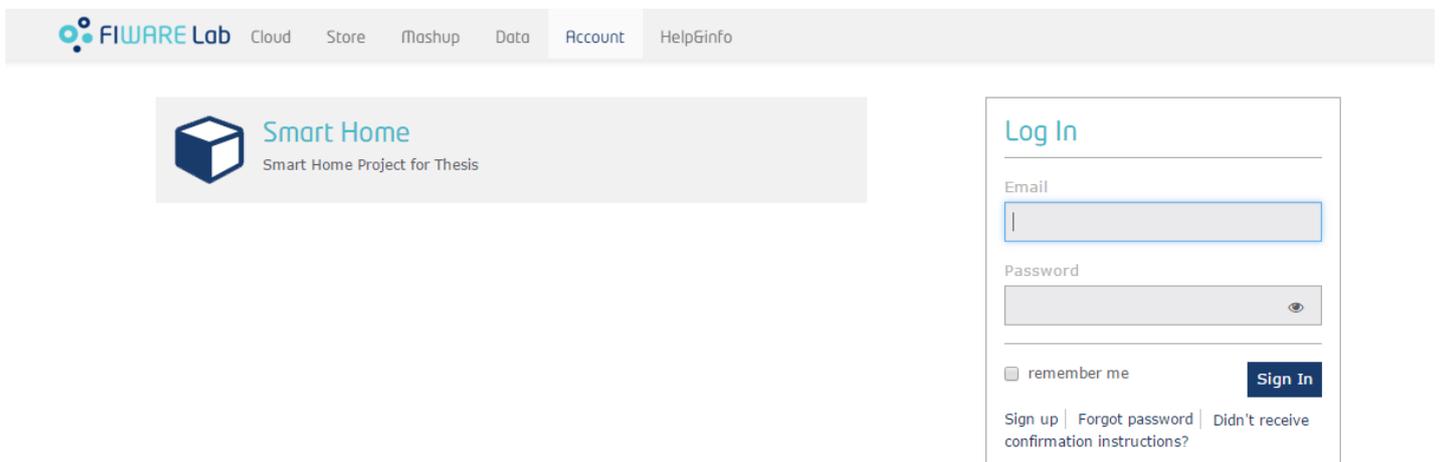


Figure 12 : FIWARE Login Page

If the data entered by the user is correct, he will be authenticated and redirected to the application's main page.

Since he is a new user he will have to enter his information and the information of his home.

The main page displayed for new users is shown below.

The screenshot shows a web application interface. At the top, there is a blue navigation bar with the text 'Home', 'Auto', and 'Logout' in white. Below this is a light blue rectangular area containing a form. At the top of this area is a button labeled 'Insert New Home'. The form consists of several input fields: 'Username', 'E-mail', 'Password', 'Home ID', 'Port Number', and 'IP Address'. There is also a dropdown menu for 'Privileges' with 'Limited' selected. A 'Submit' button is located at the bottom of the form.

Figure 13 : New User Main Page View

Use Case 2 – Existing User logging into the Application

The user enters the application's url, which is http://147.27.50.169/Smart_Home/.

If he isn't already logged in, the page shown in Figure 11 is displayed. After the user clicks on the "Login" button he is redirected to the page shown in Figure 12. After he inserts his credentials and is authenticated, he is redirected to the application's main page.

The page displayed is different for users with full and restricted privileges.

The page displayed to full privilege users is :



Figure 14 : Full Privilege Main Page View

The page displayed to restricted users is :



Figure 15 : Restricted Main Page View

Use Case 3 - Inserting a new User

Once a user with full privileges logs in and accesses the main page, he can click on the “Insert New User” button, visible in Figure 14, and insert the new user’s information in the form that appears under the button. The form that appears is :



Insert New User

Username :

E-mail :

Password :

Privileges : Limited ▾

Submit

Figure 16 : Insert New User

Use Case 4 – Inserting a new Device

Once a user with full privileges logs in and accesses the main page, he can click on the “Insert New Device” button, visible in Figure 14, and insert the new device’s information in the form that appears under the button. The form that appears is :



Insert New Device

ID :

Type : Light ▾

Submit

Figure 17 : Insert New Device

Use Case 5 – Updating Home Information

Once a user with full privileges logs in and accesses the main page, he can click on the “Update Home” button, visible in Figure 14, and insert the home’s information in the form that appears under the button. The form that appears is :

Update Home

Port Number :

IP Address :

Submit

Figure 18 : Update Home Information

Use Case 6 – Update Home Automation Settings

Once a user with full privileges logs in and accesses the main page, he can click on the “Update Settings” button, visible in Figure 14, and insert the new settings in the form that appears under the button. The form that appears is :

Update Settings

Auto-Mode :

Temperature :

Mode : Cold ▼

Submit

Figure 19 : Update Home Automation Settings

Use Case 7 – Viewing and Deleting Home Users

Once a user with full privileges logs in and accesses the main page, he can click on the “View Home Users” button, visible in Figure 14. He is then redirected to the view_users page shown below.

| | | | | |
|--------|-----------------|-------|---------|---|
| User_1 | user1@email.com | home1 | full | ✘ |
| User_2 | user2@email.com | home1 | limited | ✘ |

Figure 20 : View_users Page

In this page the user can view the information of the all the home’s users. If he wants to delete an existing user he can simply click on the red “x” button next to the user’s name.

Use Case 8 - Viewing and Deleting the Device Control History

Once a user with full privileges logs in and accesses the main page, he can click on the “View Device History” button, visible in Figure 14. He is then redirected to the view_history page shown below.

Home Auto Logout

Clear History

| | | | | | |
|-------------------|-------|-------------|-----------------|------|---------------------|
| Light2 | home1 | panos_kosto | switch_Off | 0 | 2016-01-22 00:37:41 |
| Air_Conditioner_1 | home1 | panos_kosto | set_Fan_speed | 0 | 2016-01-22 00:37:19 |
| Light1 | home1 | panos_kosto | switch_Off | 0 | 2016-01-22 00:37:04 |
| Air_Conditioner_1 | home1 | panos_kosto | Switch_On | 1 | 2016-01-22 00:36:55 |
| Light2 | home1 | panos_kosto | switch_On | 1 | 2016-01-22 00:36:37 |
| Light1 | home1 | panos_kosto | switch_On | 1 | 2016-01-22 00:36:10 |
| Air_Conditioner_1 | home1 | panos_kosto | switch_Off | 0 | 2016-01-22 00:35:44 |
| Air_Conditioner_1 | home1 | panos_kosto | set_Temperature | 26 | 2016-01-22 00:35:35 |
| Air_Conditioner_1 | home1 | panos_kosto | Switch_On | 1 | 2016-01-22 00:35:25 |
| Air_Conditioner_1 | home1 | panos_kosto | set_Program | warm | 2016-01-22 00:35:16 |
| Air_Conditioner_1 | home1 | panos_kosto | Switch_On | 1 | 2016-01-22 00:34:16 |
| Air_Conditioner_1 | home1 | panos_kosto | switch_Off | 0 | 2016-01-22 00:33:42 |
| Air_Conditioner_1 | home1 | panos_kosto | Switch_On | 1 | 2016-01-22 00:33:33 |

Figure 21 : View_history Page

In this page the user can view the information about all the device control operations performed. If he wants, he can clear the history by clicking on the “Clear History” button on the top right corner of the page.

Use Case 9 – Viewing and Deleting Notifications

Once a user logs in and accesses the main page, he can click on the notifications icon, visible in Figure 14. If it is red a list of notifications will appear, as shown below.

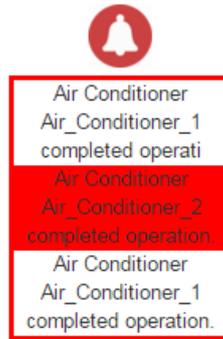


Figure 22 : View Notifications

If the user wishes to delete a notification he can hover over the desired notification, which will turn read, as shown above, and click on it.

Use Case 10 – View and Control Device

Once a user logs in and accesses the main page, he can click on the icon of the device he wishes to control, visible in Figure 14. He is then redirected to the view_device page where the information of the selected device is displayed, as shown below.

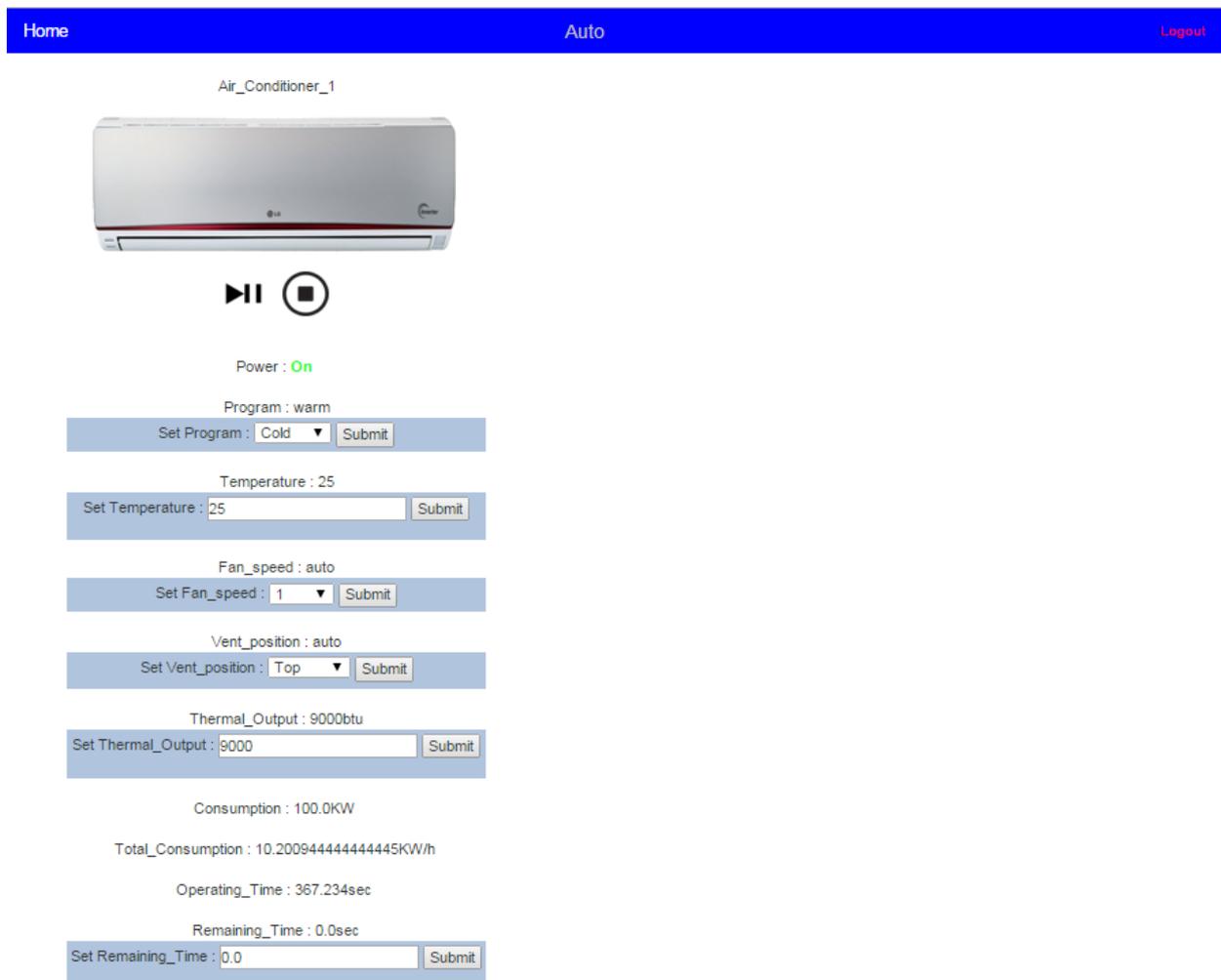


Figure 23 : View_device Page

The user can control the device's power state (on/off/paused) by clicking on one of the two buttons under the device's icon.

He can also control other aspects of the device's operation (such as program, remaining operation time, etc.) by altering one of the values in the available form and clicking submit.

The page will then be refreshed with the device's updated information.

Use Case 11 – Delete Device

Once a user with full privileges logs in and accesses the main page, he can delete a device by clicking on the red "x" button under the device he wishes to delete, visible in Figure 14. The device will be deleted and will also be removed from the page.

Use Case 12 – Enabling Home Automation

Once a user logs in, he can enable the home's home automation mode by clicking on the "Home" button on any of the pages he can access, shown in Figures 14, 15, 20, 21 and 23.

If the home automation is enabled, the "Home" button is displayed in green and, by clicking it, home automation is disabled and the button becomes gray.

If it is disabled, the "Home" button is displayed in gray and, by clicking it, home automation is enabled and the button becomes green.

When Home Automation is enabled, the view_device page's view is altered. The user can no longer control the device's functions and a message is displayed, informing the user that Automatic Mode is enabled.

This and the results of Home Automation on device control are shown the below.

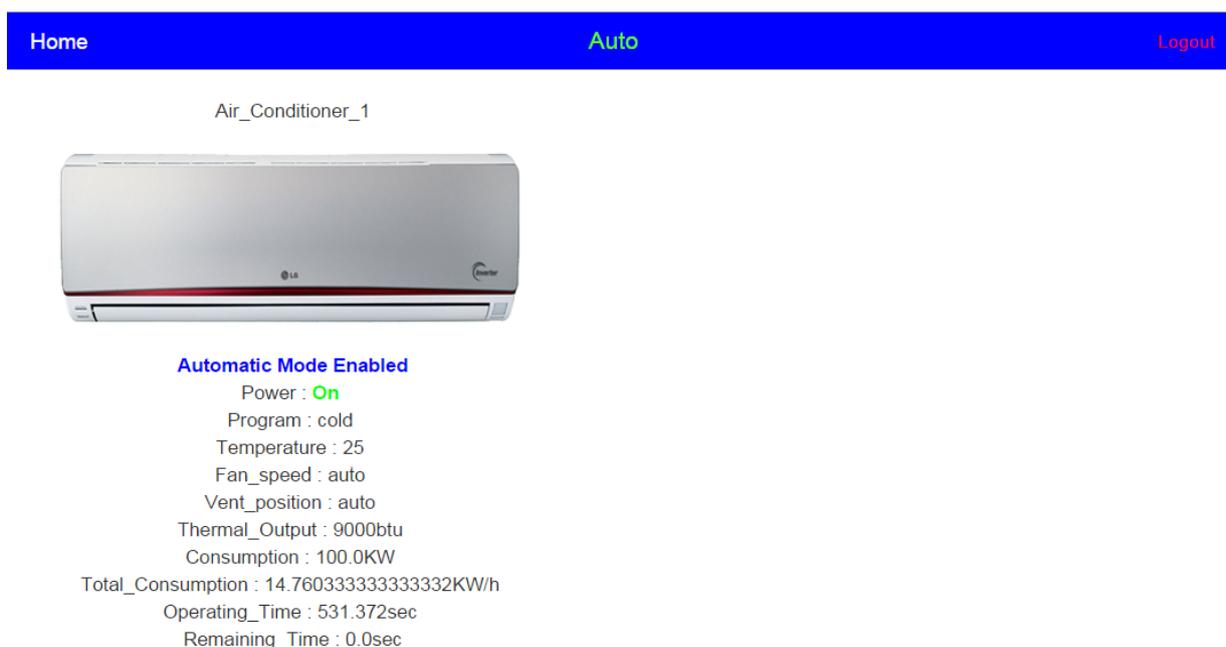


Figure 24 : View_device Page with Home Automation Enabled

In the Figure above the user has just enabled Home Automation.

Control of the device is no longer possible and the message informing him Home Automation is enabled is visible.

As we can see, the device is switched on. This was done by the user before Home Automation was enabled.

In the figure below it can be seen that time has elapsed by the “Operating_Time” value which has increased, and that the device is now switched off. This action was performed automatically by the system because the home’s temperature is below the set threshold and the air conditioner no longer needs to be in operation.

HomeAutoLogout

Air_Conditioner_1



Automatic Mode Enabled
Power : **Off**
Program : cold
Temperature : 25
Fan_speed : auto
Vent_position : auto
Thermal_Output : 9000btu
Consumption : 100.0KW
Total_Consumption : 14.93180555555554KW/h
Operating_Time : 537.545sec
Remaining_Time : 0.0sec

Figure 25 : View_device Page with Home Automation Enabled (2)

These are the application’s basic use case scenarios and it’s User Interface.

4. Conclusions and Future Work

4.1. Conclusions

The goal of this thesis was to create a system for managing smart homes which would take advantage of Cloud Computing's capabilities, combined with the idea of the Internet of Things. To accomplish this, we also had to implement the devices and access point which compose the smart home. We will now summarize our initial goals, in terms of functionality, which were achieved.

Through this service the user can create an account. After creating and logging in to his account he can register his home by registering all the wirelessly controllable devices that are connected to his home's main access point¹. After doing so he will have control over all the registered devices through this platform and will also be able to set rules for the home automation in the case he wishes to have certain aspects of his home controlled automatically.

The functions supported are :

- Registering a Home
- Viewing/Controlling Devices
- Inserting/Deleting/Modifying Device
- Viewing Home Users
- Inserting/Deleting User
- Limited/Unlimited user control rights.
- Enabling/Disabling Home Automation
- Modifying Home Automation Settings
- Viewing/Deleting Device Control History
- Viewing/Deleting Device Notifications

And the advantages it offers are :

- It is a universal control platform for all home devices.
- Any new device can be registered and controlled on the system after it has been connected to the access point.
- It gives users the ability to monitor device operation.
- User is notified about important device updates.
- It aids in consumption management and cost reduction.
- It reduces the effort and time required from the user.
- The ability to set limited control privileges for certain users increases security.
- **Ease of access.** Users can access the service from any place at any time, as long as have an internet connection is available.

- **Elasticity.** If a user needs to add or remove a large number of devices, resources can acquire be acquired or release resources to support his increased or decreased needs.
- **Reduced cost.** Users do not need to purchase equipment and aren't responsible for equipment upgrades and maintenance, which reduces cost greatly. Also, they are not left with unnecessary equipment once they decide they no longer need resources or the service entirely.
- **Pay per use.** Users are charged based on the use of the system and the resources they are using. This way the know how much they are paying and can manage it according to their needs.

The use of Generic Enabler services proved to be an important part of the system. By adding these to the system's implementation, we were able to fully make it fully functional and meet the specifications that we had originally set, as well as introduce Cloud Computing capabilities and benefits. The flexibility offered by the Intellicloud infrastructure, by providing computing resources dynamically and on-demand, was instrumental in completing the task.

4.2. Restrictions

We faced a number of restrictions and difficulties during the system's implementation process. We present them in this chapter.

- The Keyrock Identity Management GE has a long response time (approximately 30 seconds), which adds a significant waiting time for the user's log in process. This adds a negative effect to the user's experience.
- The time required for a command to be sent from the application to the access point, then to the device and from the device back to the access point and then back to the application (approximately 5 seconds in total) adds a considerable waiting time for the user. Also, in the case that the user performs a control command, the waiting time is doubled (approximately 10 seconds). This also affects the user's experience negatively.
- A common communication format had to be defined, which all the simulated devices and the access point would use for sending data. Not all actual devices use this format, though, as many use their own, depending on the manufacturer. This means that, by using actual devices, communication will have to be performed differently depending on the device, which requires constant reprogramming and customization of the system, or access point, for each new device
- The communication between the application and the access point which is performed through sockets, which, despite the authentication performed on the messages exchanged, is not a secure enough way to perform home control operations. Since the system is based on Cloud Computing, the communication should be performed by the use of a secure service. Such a service, though, would require a far more complex and demanding implementation of the home's access point, which surpasses an implementation for the purposes of a thesis.

4.3. Future Work

Our system is implemented in a way that extensions can easily be added in order to extend its functionality.

- New types of devices can be added to extend the simulated home's capabilities.
- The capabilities of the existing devices can be extended, by adding additional operational programs, notifications about more of the devices functions, etc.
- Additional Home Automation functionalities can be added by the addition of pages that handle different aspects of the home, such as automatic lighting, controlling devices to achieve less consumption (eco mode), etc.
- Reprogramming of the access point to include a database and a server for handling HTTP requests, so that communication can be performed using REST architecture.
- Following the reprogramming of the access point, a service can be added to the system which performs communication with the access point, sending commands and requesting information, using the HTTP protocol. Adding this service and replacing the existing communication implementation, extends the system's functionality as one complete service, composed of entirely of individual services, on the cloud.
- Actual devices can be purchased and added to a home to display the system's functionality in the real world.
- An actual access point can be purchased and added to the home, replacing the simulated access point. The devices will then be connected to it and an real world smart home will be available to control with the system.

Bibliography

[1] Overview of the Internet of things, by the ITU

https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.2060-201206-I!!PDF-E&type=items

[2] The NIST Definition of Cloud, by Peter Mell, Timothy Grance

<http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf>

[3] Smart Devices

https://en.wikipedia.org/wiki/Smart_device

[4] Amazon Web Services

https://en.wikipedia.org/wiki/Amazon_Web_Services

[5] IBM SmartCloud

https://en.wikipedia.org/wiki/IBM_cloud_computing

[6] Microsoft Azure

https://en.wikipedia.org/wiki/Microsoft_Azure

[7] Service-oriented architecture, by Perrey R., Lycett M.

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1210138&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs_all.jsp%3Farnumber%3D1210138

[8] Hypertext Transfer Protocol -- HTTP/1.1, by ielding Roy, Gettys James, Mogul Jeffrey, Henrik Frystyk, Masinter Larry, Leach Paul J, Berners-Lee Tim

<https://tools.ietf.org/html/rfc2616>

[9] REST Representational State Transfer, by Michael Jakl

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.7334&rep=rep1&type=pdf>

[10] The application/json Media Type for Javascript Object Notation (JSON)

<https://tools.ietf.org/html/rfc4627>

[11] Virtualization and Cloud Computing, by Yuping Xin and Yongzhao Zhan

http://link.springer.com/chapter/10.1007/978-3-642-27323-0_39

[12] Keyrock Identity Management GE

<http://catalogue.fiware.org/enablers/identity-management-keyrock>