ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria e Architettura
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

AMBER:
A CLOUD SERVICE ARCHITECTURE PROPOSAL

Elaborata nel corso di: Applicazioni e Servizi Web

Relatore:
Prof. SILVIA MIRRI
Co-relatori:
VITTORIO GHINI

Tesi di Laurea di:
FEDERICO FOSCHINI

ANNO ACCADEMICO 2016 - 2017
SESSIONE I

KEYWORDS

Cloud
Architecture
Internet Of Things
Big Data
Smartness

"Your time is limited,
so don't waste it living someone else's life.
Don't be trapped by dogma - which is
living with the results of other people's thinking.
Don't let the noise of others' opinions
drown out your own inner voice.
And most important, have the courage
to follow your heart and intuition.
They somehow already know
what you truly want to become.
Everything else is secondary.
Don't settle.
Stay hungry. Stay foolish."

Contents

Introduction

Nowadays, the *Cloud* is one of the most important contexts as concern the improvement and the grow up of the related technologies. In fact, it's the main technology used to design and build most of modern solutions.

Furthermore, this context is growing in parallel to the importance and value of the *data*. In fact, in any context, company and business area, the value of this concept is growth up because it's possible to improve the qualitative and quantitative aspects related to the business.

The concept of *data* is strictly coupled with the *Internet Of Things* and *Big Data* concepts. In fact, these two concepts and related technologies are constantly growing up coupled with the *Cloud* and the *data* contexts.

The *IoT* is linked to the *data* because the related main technologies are used to collect any kind of data (eg in the industry automation the *IoT* technologies are used to collect the industrial machines data).

The *Big Data* is coupled with the *data* due to the main scope of this context methodologies and technologies. In fact, they are used to analyze, manage and extract useful informations from the collected data (eg through the *Batch Analysis* it's possible to extract informations from the collected historical information).

More in details, the *Internet Of Things* and the *Big Data* contexts can be referred to the *Cloud* considering it as the enabling technology and platforms. In fact, the *IoT* and the *Big Data* can be considerately improved using the *Cloud* platforms and technologies.

Thus, the relation between these contexts will be considered as a milestone as concerns the solution design and building in order to focus on the solution future generations.

On the basis of these contexts new kind and type of solutions were born. In fact, the improvement of the main technologies, platforms and environments are the fundamentals of new solutions contexts.

The *Cloud*, *Internet Of Things* and *Big Data* contexts are strictly coupled with many modern and innovative solutions contexts.

All these solutions have as fundamental the concept of *Smartness* associated with many environments and aspects which have as primary target the support and help to the people.

In fact, the *Smartness* has been referenced to main targets such as buildings, naming them as *Smart Buildings*, in order to improve the people living aspects related to the buildings (eg there are solutions to increase energy efficiency, reduce energy consumption so as to improve the environmental sustainability of the building).

The *Smartness* can also be related to the cities, in fact, the *Smart Cities* contexts is referenced to the solutions to improve the people living aspects related to the cities (eg there are solutions to improve navigation by high energy sustainability vehicles to reduce emissions of pollutants).

Another important context strictly coupled with the *Smart Buildings* and *Smart Cities* ones is the *Accessibility*.

Thus, this context has a primary role in modern solutions dedicated to the improvement of the quality related to the living aspects focusing on the special needs required by people whose have disabilities or physical impairments (eg people whose have to move with wheel chairs).

On the basis of the main concepts and issues related to the *Accessibility* contexts, the implemented case study will be described and analyzed in the Chapter 6. The case study and its scopes have been selected focusing on the primary role of this context in modern solutions.

Therefore, *Cloud*, *Internet Of Things* and *Big Data* are the main concepts coupled with this thesis context because all of them have a key role in several modern solutions design and build processes.

One of key aspects treated and analyzed in this thesis is the *Cloud* platforms architecture. The concept of *Architecture* has been used focusing on the nature of the *Cloud*, because it has been refined with the introduction of this context.

In fact, this term was used by considering the analyzed *Cloud* Platform services and components as the structural parts of the platforms themselves.

This choice was made because the *Architecture* of the software system represents the structural organization of the system, including its software components, the properties externally visible from each of them and the relationships between the parties [1].

Thus, in this case, it was decided to refer to the services as direct components of the platforms by abstracting from the individual internal ones.

On the basis of this platform services abstraction, it has been chosen to use the term *Architecture* as a shrinkage of *Cloud Service Architecture* [2].

On the basis of these main concepts, this thesis proposes the definition of an architecture model that aims to support the design of different types of *Cloud* solutions to facilitate, simplify and improve modeling and design processes.

The proposed architecture is the result of an accurate study of the key issues with *Cloud* design and its related aspects of the necessary effort and resources, focusing on platforms and technologies related to real solutions and scenarios.

One of the goals of this model is transparency over technological aspects. For this reason, it will be described how this architecture model can act as a wrapper between the solution and the platform that hosts it, thus making the technology complex closely linked to the platform transparency.

Another aspect of particular interest is linked to the modules that make up the architecture, as they can be distributed and adopted on different *Cloud* platforms, without affecting their interactions, features and functionality, because of their independence in distribution and evolution. In fact, one of the key features of this architecture is being oriented towards multi-platform solutions, a key feature for *Cloud* solutions.

The proposed architecture aim also to be an helper to support the modeling and design of multi-contexts solutions. In fact, it aims to support the main *Cloud* solution types such as *Internet Of Things*, *Big Data* and *Cloud* generic applications.

On the basis of the main aspects defined about the target solutions, one of the key features of this architecture is being oriented towards multi-contexts solutions, another key feature for *Cloud* solutions.

The modeling and design processes related this kind of solutions is particularly complex and it requires an high effort due to their complexity and extended pool of coupled issues.

More in details, the architecture that will be defined and described in this thesis allows to optimize the effort and resources needed for modeling and designing multi-platform *Cloud* solutions, since the complexity and the associated issues with this phase are greatly reduced in case it is adopted the proposed architecture.

At the end, focusing on this thesis structure and concepts, its outline will be the following one:

1. **Background Knowledge**: chapter to introduce and describe the main background knowledge needed to understand the main concepts, aspects and technologies treated in this thesis.

   This chapter is particularly important because these main knowledge represents the fundamentals related to this thesis proposed architecture and its objectives, features and scopes.

2. **Architecture Design**: chapter related to the modeling, design and definition of the proposed architecture in order to provide its overall description. This chapter will define all main aspects of the treated architecture and the related modules and components.

   This chapter will describe, design and present all the aspects, features and capabilities related to the proposed architecture.

   Furthermore, all the internal components and modules will be presented in order to describe all their modeling and design details.

   All the definitions, descriptions and analysis will be done focusing on high abstraction level because the main scope of this thesis is to model and design the target architecture, so the technological details won't be treated.

   Thus, the focus will be set on the key aspects of this architecture scopes, features and capabilities.

3. **Cloud Platforms**: chapter related to the description of the main *Cloud* platforms that can be adopted to realize and develop the proposed architecture.

   In this chapter *Amazon Web Services*, *Microsoft Azure* and *Google Cloud Platform* will be described and analyzed as concerns their architecture, advantages, disadvantages and an overall evaluation.

   The choice has been based on many factors related to the realization of *Cloud* solutions because the focus has to be put on the platforms that will receive the best upgrades, improvements and innovations.

Furthermore, the Gartner [3] has evaluated the selected platforms as the best three public *Cloud* platforms focusing on their performances and most relevant aspects for customers (eg the focus has been set on key properties such as computing performances and pricing) [4][5][6].

The *Cloud* platforms and technologies have an extremely high speed development, upgrading and improvement so it's not easy to find recent descriptions or evaluations about them. In fact, the most updated sources are the official documentations provided by the platforms vendors, because most of the analysis are referred to open source platforms while the selected ones are proprietary [7][8][9][10].

Thus, the evaluation factors selected to analyze the chosen platforms will be based on many qualitative aspects as treated in the report *The Future of Cloud Computing Opportunities for European Cloud Computing beyond 2010* [11] (eg the development processes support tools will be one of the evaluation factors of interest).

4. **Technological Evaluation**: chapter related to the technological evaluation of the proposed architecture in order to provide the description referred to the technological aspects to build the target architecture.

   In this chapter will be presented the proposed architecture main evaluations referencing to the main *Cloud* platforms. Thus, all these evaluations will support the overall quality and the feasibility of this thesis presented architecture.

5. **Case Study**: chapter to describe two main case studies, one of them will be focused on the architecture implementation logical abstractions while the other one will be focused on an implemented system in order to assess the target architecture main characteristics and feasibility.

   First of all, the main logical case study will be provided so as to define how the proposed architecture can be developed in the main scenario.

   Then, an implemented case study will be described so as to provide an overview of the adoption of the target architecture in a real scenario.

   The implemented case study will be based on the porting of an existing Web based solution to the *Cloud* adopting the proposed architecture.

The chosen solution is *UniSAS*, a system thought to improve the accessibility level of any building. In fact, this system acts as an helper to automatically open, close and manage any kind of doors and other structure that could limit the accessibility and mobility of a person.

The main objective is to help all the people situated in a building but, focusing on the accessibility and mobility issues, the main targets of this system are the disabled people (eg wheelchair users) or those people with physical impairments.

Furthermore, this system can act also as an access controller so as to control and manage access privileges. In fact, all the rules related to places and rooms accessibility can be managed by using a dedicated tools.

Focusing on the building management processes and analysis, *UniSAS* can be used to collect and manage data so as to improve all the maintenance actions and processes. This capability has a primary role in modern solutions because all the processes are managed through specific data analysis.

Thus, these case studies will present the main aspects, characteristics, features and capabilities of the proposed architecture so as to assess its overall quality and feasibility.

6. **Future Developments**: chapter related to the proposed architecture future developments in order to describe how it can be extended, upgraded and improved.

   The developments whose will be described could represents the starting point of the improvement process of the presented architecture.

   All the future developments defined have been designed basing on the main aspects coupled with the *Cloud* contexts.

Chapter 1

Background Knowledge

In this chapter, the fundamentals concepts and knowledges on which is based the proposed architecture and related technologies that will be dealt with in the latter chapters will be introduced and described.

The background knowledges selection process has been focused on the most important concepts and issues affecting the proposed architecture context in order to offer an overview of their characteristics and properties. In particular, the selected areas are the following ones:

1. **Cloud**: it represents the main context related to the proposed architecture in terms of analysis, modeling, designing and development.

2. **Internet of Things**: the proposed architecture contains modules dedicated to this context. Furthermore, it has a leading role in the *Cloud* solutions scope, because the main solutions are oriented to this context.

3. **Big Data**: the proposed architecture contains modules devoted to this context. Furthermore, it has a leading role in the *Cloud* solutions scope, because the main solutions are oriented to the data analysis.

In the following sections of this chapter, such areas and related issues will be described, so as to support the understanding process of the concepts and thematics treated during the modeling, designing and development of the proposed architecture, because it is strictly coupled with these contexts.

1.1 Cloud

Nowadays, the *Cloud* is one of the most important commonly-used technologies. This technology is particularly interesting due to the solutions related to the collection, management and analysis of every type of data and information. These kind of solutions are the most interesting in industrial and enterprise solutions.

The most relevant definition of *Cloud* has been provided by the *Nist* [12]:

"Cloud Computing is a flexible execution environment that enables network access and on-demand to a shared set of configurable computing resources (eg. networks, servers, storage devices, applications and services) as services at various levels of granularity. These services can be quickly requested, supplied and released with minimal management effort from the user and minimal interaction with the provider."

In the following sections will be provided the definitions, types and descriptions related to the *Cloud* environments and technologies, so it will be provided an holistic vision of these fundamental context for these technologies.

1.1.1 Cloud Computing Models

In the *Cloud* context, there are many different models related to the technology and its related using strategy.

The four kinds of model of *Cloud Computing* are:

- **IaaS**: this models is defined as *Infrastructure as a Service* and it offers as services all resources related to memory, computation and elaboration. In this model the services user can use all resources as virtual machines, servers and related infrastructure in order to allows the elaboration and computation on provided hardware. In this model, referencing to the *ISO OSI* standard one [13], the user will give the complete management about all the levels related to infrastructure above the physic hardware levels, so all the applications and services will be completely managed by the user, while the hardware and physic infrastructure will be managed by the provider.

- **PaaS**: this model is defined as *Platform as a Service* and it offers a complete platform to develop complete services and applications as services. The user can design and develop the solutions, using the tools and support systems provided by the platform. In this model, referencing to the *ISO OSI* standard model [13], the provider will manage and maintain the levels below the application one, while the user will use the tools related to the higher levels of the standard model.

- **SaaS**: this model is defined as *Software as a Service* and it offers complete software solutions and software as services, completely managed by the provider and the user can only use and interact with these software. In this model, referencing to the *ISO OSI* standard model [13], the provider will manage and maintain all levels of the standard model related to the applications. Generally, a solution *SaaS* has been designed and developed using a *PaaS*.

- **DaaS**: this model is defined as *Data as a Service* and it offers access and management to the user's data basing on specific factors about the authentication and management as services. In this model of *Cloud Computing* are provided to the users all data contained in the provided infrastructure as they resided locally on the user's machine.

These models of *Cloud Computing* differ basing on the context and target features that has importance for the target final user. When the designer and developers have to choose and decide about the type of *Cloud* to use and adopt, they will maintain the focus on targets and interesting contexts related to the solutions.

1.1.2 Type of Cloud

In this section, the principal types of *Cloud Computing* will be described together with their differences relating to the infrastructure, target features and the way to use them.

Relating to the *Cloud Computing* main types, there are four main types:

- **Public**: type of *Cloud Computing* related to the the public usage context, where the resources are provided by a provider through the Internet network to all interested users. The only constraints are related to the authentication forms and mode and to the resource usage, basing on many different plans e costs related to each one.

- **Private**: type of *Cloud Computing* related to the private usage context, where the resources are provided to a single customer, so the *Cloud* services can be used only by the target customer. This customer can also be a provider in order to use its *Cloud Computing* in its company and in order to provide its internal features and tools by the same product to the company costumers.

- **Shared**: type of *Cloud Computing* related to the scenario with a limited pool of costumers where they are well defined and specified because they work in the same context in order to share resources, information and data using a common and maintained platform (eg. many Universities could use a *Shared Cloud Computing* because they work together in the same contexts).

- **Hybrid**: type of *Cloud Computing* created by mixing the other types. In this type, the provider have to manage all constraints of each type that has been mixed in order to maintain all properties and characteristics of the selected types.

Basing on the described types of *Cloud Computing* provided by companies, the selection of the target *Cloud* type to adopt has to be done by a focused analysis where the attention and the main target are the users who can access and use the solution to provide, in order to choose the *Cloud* type that best fits to the context and scope of the target solution.

1.1.3 Advantages of the Cloud

Using the *Cloud* will give many advantages and benefits to the costumers and users and they affect particularly two context: economic and technical.

The main important economic advantages and benefits given by the adoption of *Cloud Computing* are the following ones:

- **Costs Reduction**: the *Cloud* solutions will reduce the costs related to the maintenance, management and upgrade of the infrastructures, because all the costs are related to the resources usage. Basing on the *"pay-per-what-you-use"* model, all the infrastructure and platform costs are affected by the resources actually used by the costumer. In the traditional infrastructure, the costs are strictly coupled with the hardware resources available and when an upgrade is required, these costs are greater because the hardware needs to be upgraded. In the *Cloud* solutions, instead, the resource requests and usage is managed by the costumer who can require and release resources depending by the needs, in a flexible way. This model of resources management and requirement allow the costumer to reduce considerably the costs.

- **Flexibility**: all the resources provided to the costumer can be managed in a flexible way by the user so It's possible to update the resources requested and released in order to optimize the related costs and available solutions.

- **Focus to the Core Business**: the infrastructures and platforms are property of the provider and the related maintenance, management and support are all its charges. In this scenario, all the customer's human resources can focus their work and capabilities to the *Company Core Business*, the primary company context with absolute importance. Another important factor affects the human resources pool and skills because, in this scenario, a company needs to *Core Business Skilled Human Resources* and it doesn't need to other infrastructure skilled human resources and this assumption will reduce the human resources number.

The main important technical advantages and benefits given by the adoption of *Cloud Computing* are the following ones:

- **Development Time Reduction**: thanks to the available tools provided by the infrastructure and the platforms, the development became faster than the traditional solutions, because all the tools, components and infrastructure provided are completely maintained and supported by the provider so the designers and developers can focus only to the target solution.

- **Test Phase Validation**: using a *Cloud Platform*, the test and production environments are perfectly identical so the one used during the test phase will be the same of the production release. In the tradition *On-Premises* solutions, the test phase has to be made using a production similar environment, so there are many differences that can influence the test phase, so the test can't be perfectly valid for the production final release. In the *Cloud* scenarios, the environments are perfectly identical so there are no differences and all done tests will be valid for the production environment and the test phase will be valid for the final release.

- **Scalability**: using a *Cloud Platform* allows to require and release resources by needs and by the usage of many monitoring and control systems related to the available solutions (eg. systems that monitor the users' requests to the solutions). It's possible to require more resources in order to improve the solution scalability basing on requests from the users in order to satisfy all the requirements related to the new scenario.

- **Greater Mobility**: using a *Cloud Platform* allows to provide all data, information and solutions through the network, so they will be available anytime and anywhere to the users who request them. The constraints affecting this scenario are related to the mobility principles, in fact, in this scenario the data and applications will be available only if the mobility constraints are satisfied (eg. the users' devices must have connectivity).

- **Security**: using a *Cloud Platform*, all data and information are centralized and specific security policies are applied to them basing on the costumer and users requests. This factors reduce the risks related to the information and data losing caused by physical theft of the company hardware and materials.

- **Disaster Recovery**: using a *Cloud Platform*, all data and applications are protected by the adoption of specific policies and mechanisms related to the the *Disaster Recovery*, in order to recover extremely dangerous scenarios and related critical issues.

1.1.4 Disadvantages of the Cloud

Using the *Cloud* will give many advantages to the user but, like all contexts and innovations, it will give also many disadvantages. These disadvantages are related particularly to the security context and to the platform provider addiction.

The main security disadvantages given by the adoption of *Cloud Computing* are the following ones:

- **Losing Control of Data**: using the *Cloud*, it's not possible to maintain the complete control of the company data contained in the *Cloud Platform*, because they are relied on the external provider.

- **Lack of Security Standard**: the *Cloud* is growing faster and faster and this growth was manifested in the last years, so there is an important lack of standards related to the security context of this king of systems.

- **Privacy**: all the data contained in a *Cloud Platform* are distributed and located on many geographic areas and each of these areas has different legislation from the others in terms of privacy related to the information and data context. Due to these differences, there can be many critical issues related to the privacy.

The main disadvantages affecting the addiction to the platform provider are the following ones:

- **Influence about the Maintenance**: there is no influence about the quality, frequency and duration about the maintenance interventions made by the *Cloud Platform* provider.
- **Migration to other Providers**: due to the lack of market standards, the migration process from a provider to another one is particularly difficulty, because there are many issues related to the technical, economic and contract contexts.
- **Customization**: the *Cloud Platform* provider can provide all the components, tools and infrastructure related to its platform and the related upgrade and customization have to be decided by it. In this scenario, the provider often reject the customization requests made by the costumer in order to avoid important and critical variations on the platform.
- **Technology Decisions**: all the decisions affecting the technology innovations have to be made by the provider and this scenario could slow down the innovation process of the costumer solutions.

1.2 IoT - Internet Of Things

The *Internet Of Things* (from which the acronym IoT has been introduced) is one of most interesting and important contexts affecting the *Cloud* solutions and related technologies [14].

One of the most accredited and complete definitions of the *Internet Of Things* has been provided by ITU[15]:

> *The Internet of Things is a global infrastructure for the information society, it enables the interconnection of things, physical and virtual, based on existing and evolving technologies for interoperability of information and communications.*

Nowadays, this context has a primary role and an high importance so there are many definitions and specifications that differs from the one provided by the ITU. They specially differs as concerns the technological specifications detail level but the main concepts and context specifications are consistent to the definition previously described one that as been provided by the ITU.

In the following sections, it will be provided the main definitions, concepts and aspects related to the *Internet Of Things* in order to provide an holistic overview of this context that represents a fundamental affecting the proposed architecture.

1.2.1 Internet of Things Fundamentals

The introduction of the *Internet of Things* has to be based to the main fundamentals affecting this context in order to focus the attention to the main aspects and concepts related to the IoT.

The main concepts related to the IoT range over to many contexts so it's not possible to base the target fundamentals identification, analysis and classification process on a single context.

The main selected fundamentals affecting the IoT are the following one:

1. **Analog Data**: this type of data represents the evaluation of the physic real world characteristics and properties (eg. it's possible to analyze the aspects related to the natural properties as temperature and humidity in order to analyze and evaluate them by dedicated

sensors). This type of data is defined as *analog* doe to its values domain has an high cardinality.

2. **Constant and Persistent Connectivity**: the *IoT* solutions and related devices needs to be constantly connected and this connection has to be persistent and stable in order to make them as always available. This factor allows to obtain the following advantages:

   - **Constant Monitoring**: the constant monitoring of the devices allows to provide an high definition and a real-time knowledge of the condition, state and properties affecting the devices and target environment where they act (eg. the industrial devices monitoring and the related environment where they work).
   - **Maintenance**: thanks to the constant monitoring it's possible to execute maintenance interventions on the devices basing on particular aspects and criteria evaluation (eg. it's possible to update the devices firmware and affected systems when it's necessary to improve the devices performances or fix any bugs).
   - **Motivation**: the advantages given by the constant and persistent connectivity provide the fundamentals to encourage innovation of products and makes the systems more attractive for potential buyers.

3. **Real Time**: the *real time* concept applied to the *IoT* context refers to the fact that the data will be provided as soon it is acquired (eg. it can be acquired by a sensor device). This concept, coupled with the interaction mechanisms and the evaluation of the scenarios basing on defined criteria, allow to considerably improve the solutions reactivity. This factor allows to merge the environment related to the *Operative Technologies* with the environment affecting the *Information and Communications Technologies*.

The *IoT* context is strictly coupled with many other contexts.

One of these contexts is the *Big Data* that will be described in the section 1.3, because it influences the main concepts and aspects affecting the *IoT* context in order to improve and enrich it in terms of characteristics and scopes [16].

1.2.2 IoT Technologies

The *Internet Of Things* context includes many technologies, characterized by many contexts in terms of type and scope of solutions building with different maturity levels.

In order to provide an exhaustive overview related to the state of art of these technologies and the affected future scenarios relating to the development, innovation and design processes, it's necessary to introduce and describe the technologies that enable and support the IoT in terms of characteristics, architectural and functional properties.

The main technologies related to the IoT context are the following ones:

- **RFId**: acronym of *Radio Frequency Identification*, it's the easiest technology to use that enables the devices integration in an IoT solution. It's one of the earliest technologies introduced in this context thanks to its properties relating to energy consumption, ease of use and consistency.

  There are two main types of this technology:

  - **Passive RFId**: *RFId* type that groups the standards related to the automatic identification through the radio frequency that doesn't require battery powered devices. This characteristic is relevant because the energy saving is one of the most important aspects and targets affecting the IoT context.
  - **Active RFId**: *RFId* type that provide more functionalities than the passive one thanks to the battery powered devices used in this type solutions. The powering of these devices through an on board battery allows to improve the performances affecting the communication and it enables the autonomous behaviour introduction on these devices without the interaction with others like in the *Passive RFId* scenarios.

- **Personal Communication**: it's a technology that groups the standards affecting the short range communications, identified with the acronym *PAN (Personal Area Network)*. This kind of networks has been introduced in order to support solutions and applications characterized by narrow-band communications where a there is particular

focus on the energy saving. The most diffused technologies related to this context are *Bluetooth* and *Bluetooth Low Energy*.

- **WiFi**: this is one of the most used technologies supported by protocols that enable the *wireless* access to local area network characterized by long range communications. This technology is supported by protocols that allows to transmit large amount of data in order to provide an high performances support to the communications, in terms of speed and data transmission capability. This technology has high energy consumptions so it introduces many limitations to its adoption in IoT common solutions. The only solutions where it's possible to adopt this technology are those where it's needed a constant and persistent *wireless* connectivity and it's available a constant power supply to the devices, so there are no limitations related to energy consumptions (eg. this technology can be used in the indoor localization solutions).

- **Wireless Bus**: *wireless* technology which groups all the standards that represent an alternative to the classic *wireless* solutions. This type of technology allows to design and build communication architectures particularly sophisticated because it's based on the principles and concepts related to the *field bus*.

- **NFC**: acronym of *Near Field Communication*, it's a technology that allows to give easy and secure interactions in a bi-directional communication model between devices so there is no need related to a physical contact between them.

- **LoRaWAN**: it's a technology that implements the specifications affecting the *Low Power Area Networks (LPWAN)*, territorial networks with limited or extended dimension characterized by reduced power consumption. This technology is intended for pervasive use within the IoT context, because it provides a complete interoperability between devices with any needs related to complex and critical installations and infrastructure. Therefore, it provides a more elastic support than traditional communication technologies, with no constraints and limitations related to the IoT solutions building.

- **Mesh Low Power Networks**: it's a technology affecting networks composed by *low power* nodes and devices with reduced energy con-

sumptions, characterized by complex architectures with capabilities to auto-configure itself in order to support packages dynamic routing.

- **PLC**: acronym of *Power Line Communication*, it's a technology characterized by the informations transmission supported by the use of the devices power supply signal. In fact, using the same signal to the devices powering and the informations transmission, it's require that in this technologies solutions can be used many protocols, split up by different power signals voltage levels, in order to provide many different data-rates and communication ranges.

- **Mobile Networks**: it's a mobile communication technology that groups all the standards commonly-used by the mobile phones and devices such as *GRPS*, *GSM (commonly named 2G)*, *HSPDA (commonly named 3G)* and the most recent *LTE (commonly named 4G)*. These technologies have an high energy consumption so it's possible to adopt them in the *IoT* solutions where it's available a stable and constant devices power supply in order to support their functionalities and communications.

The described technologies compose the pool of the main technologies used in the *Internet Of Things* solutions context, but there is a constant and continuous research related to these technologies innovation as concern communication and interaction aspects and concepts.

The major scopes of these research processes are to improve these technologies performances and reduce the energy consumption.

1.2.3 IoT Protocols

The *Internet Of Things* include an extended pool of technologies and protocols related to the interactions and communications between devices and they are characterized by different applicative contexts and scopes.

In order to provide a depth description about the *IoT*, it's necessary to introduce and provide an overview of its protocols and their description. Some protocols used in the *IoT* solutions have been introduced previously (eg. the *IP*), so the focus will be set to the protocols introduced and improved to support the *IoT* context solutions.

The selected main protocols relating the *IoT* context are the following ones:

- **MQTT**: acronym of *Message Queue Telemetry Transport*, it's a protocol whose purpose is to transmit all the collected data by the devices to the target platforms.
- **XMPP**: acronym of *Extensible Messaging and Presence Protocol*, it's a protocol introduced to support the instant messaging communication in order to interconnect all the target nodes. In this scenario the collected data will be sent to the target platforms as messages.
- **DDS**: acronym of *Data Distribution Service*, it's a protocol dedicated to the direct communication of data between devices, so it provides a powerful support for this kind of communications and interactions.
- **AMQP**: acronym of *Advanced Message Queuing Protocol*, it's a protocol based on the data management queue oriented, so it's supported by queue management components and systems, in order to provide a powerful support to the communication between the infrastructures nodes that act as servers.

The described protocols has not been introduced specifically to support the *IoT* context, but they have been particularly improved and evolved thanks the rapid escalation of this context grow up.

These protocols need guarantees as concerns the communications and transmissions reliability, so they are fully supported and based on the *TCP* protocol as data transfer protocol. This factor allows to build *Internet Of Things* solutions extremely efficient and stable as concern the data transfer.

At the end, it's necessary to describe that the major producers and vendors of software and hardware products that will be used in *Internet of Things* solutions, in the last years have made a relevant change of their vision because they adopted an innovative vision affecting the research and development of *IoT* technologies. In fact, they have been grouped in many consortia and associations, in order to define standards related to the *IoT* context as concerns the design, modeling and development of protocols, best practices and methodologies.

This approach will bring to a complete openness scenario where these systems and platforms can interact without constraints and limitations.

Another advantage provided by this kind of collaboration affects the innovation of technologies and mechanisms because these collaboration can improve and speed up the technological innovation as concerns the design and development process and, at the same time, will improve the resolution of problems and critical situations related to these processes. Therefore, this vision will improve the quality level related to this kind of technologies.

1.2.4 IoT Critical Issues

The *Internet of Things* provided many innovations and advantages affecting the device management and data collection, but at the same time has introduced also many problems and critical issues.

The most relevant critical issues are related to the IT security as concern the applicative scenarios of these technologies. This context has been selected because this kind of issues has been defined as the most important and critical by many analysis and studies. The critical issues affecting systems architectures and development has not been selected as relevant, because they are related to the systems building process so they don't directly affect the IoT context technologies.

The introduction and the diffusion of the *Internet of Things* technologies have revealed many security critical issues, due the exponentially grow up of the quantity of interconnected devices has increased considerably the IT security attacks impact surface.

The increase of this impact surface represents an important factor, because in all IT security contexts the main scope is to reduce this surface in order to enhancement the systems defenses by increasing the difficulty level for the attackers.

At the end, basing on this analysis, the IoT critical issues can't be defined as constraints or limitations for the diffusion and the improvement of this context technologies, because the advantages are particularly relevant for many kind of solutions (eg. the improvement of the monitoring related to the industrial production processes that provide a support to improve production performances and the related quality level), so this factor will be an important incentive to enhancement the systems security in order to reduce and limit this kind of issues.

1.3 Big Data

The *Big Data* is one of the most relevant context, because it's constantly spreading due to the high importance of the data and related analysis in modern solutions [17].

The *Big Data* term refers to a collection of data and affected management and analysis structures. The treated data have size and complexity so high that innovative tools are required, because the traditional data analytics tools can't manage this kind of data due to their characteristics. This need regards all the data management and analysis life-cycle, in order to provide a complete tools collection to manage all the data management processes[18].

In order to manage and handling chunk of data having dimension and complexity exponentially higher than the traditional scenarios, it's necessary to provide a pool of platforms, infrastructures and tools that have to be always available and scalable that can manage and distribute the computational loads on many distributed units, in order to manage amounts of *Zettabyte* (millions of *Terabyte*) data.

Basing on the requirements and needs required by the platforms and infrastructures dedicated to this technological context, the fundamental is to have a communication and interaction system that has to be reliable, efficient, extended with high performances.

The many data sources who supply this kind of data stores are related to many different contexts, from the social media context (eg. the shared post tracking), to the *Internet of Things* [19] (eg. the data collection relating to the building indoor environment properties like temperature and humidity).

The collected data can be used in many areas and types of analysis, from the historical data analysis (eg. the marketing analysis related to the previous months), to the innovative scenarios predictive analysis (eg. the predictive analysis relating to the predictive maintenance of vehicles).

This heterogeneous pool of contexts related to data collecting and analysis refers to the *Big Data*, so it will become one of most important contexts for modern solutions because the data analysis will be the companies primary target.

1.3.1 Big Data Properties

The *Big Data* properties can be defined and described basing on a model named as *4-V Model*, because all the properties name start with "V".

These properties are the following ones:

- **Volume**: it's a property related to the data sets dimension.
- **Velocity**: it's a property related to the data generation and collection speed as refers the data analysis and management system supply. This property will be increased in future development in order to obtain analysis and management systems that acts almost as *real-time*.
- **Variety**: it's a property related to the type and heterogeneity related to the data contained in the data sets. These data are collected by different sources in terms of contexts, environments and structural characteristics (structured and unstructured).
- **Veracity**: it's a property related to the data accuracy, in order to keep "bad" data from accumulating in the system. Thus, this property requires to keep the data only if they are trusted and validated.

In many studies and analysis, the *4-V Model* has been extended with other *V-Properties*, because the base *4-V Model* does not contains all the main properties related to the properties definition and description about the *Big Data*.

The added properties are the following ones:

- **Variability**: it's a property related to the possible inconsistence of the data sets contained data. This kind of scenario can be a problem and it can generate problems and critical scenarios if the main data management and analysis system is data-consistence based.
- **Visualization**: it's a property related to the modalities affecting the target informations representation, in order to allow the end users to visualize large amounts of complex data in an easy and effective way.
- **Value**: it's a property related to value of the analyzed and processed data. This property is the end game of *Big Data* analysis and processes. After addressing volume, velocity, variety, veracity, variability

and visualization, which takes a lot of time, effort and resources, it's possible to get the effective value of the data.

The model choice has to be based on many factors related to the data context, design and development methodologies and the used technologies. In fact, the most used model is the *4-V Model*, because it contains the fundamentals properties related to the *Big Data* context. The other models are less used because they increase the *Big Data* systems design complexity, because they add constraints related to properties complex to manage.

1.3.2 Big Data Dimensions

The *Big Data* context define many dimensions related to the use of this technology that depend on requirements and needs of the final users as concern the data and related analysis.

The main dimensions related to the *Big Data* context are the following ones:

- **Totality**: the final users are interested and focused on the data elaboration and analysis process affecting the data set contained data.
- **Exploration**: the final users require to use analytical approaches to analyze the data. The used schema is defined and adapted basing on the target queries to do and the related data nature and type.
- **Frequency**: the final users require high performances and efficiency relating to the data analysis, in order to use many different *Business Intelligence* techniques to analyze the target data.
- **Dependency**: the primary need is to balance between the investment related to technologies that have to do highly efficients and reliable and the investment affecting innovative technologies.

Therefore, it's possible to define that the *Big Data* context is strictly coupled with the high dimension and quantity data management technologies and methodologies, but this context require also to have high performances and efficiency systems in order to satisfy the final users' requirements and needs.

1.3.3 Traditional Context vs Big Data Context

The *Big Data* context is significantly different from the traditional context relating to data analysis and management, so it's different the used approach relating to the data life-cycle management. In this section will be described the main differences relating this comparison.

The main differences between these approaches are the following ones:

- the analysis related to the traditional context are based on a data portion exploration, because the complete data set can not be completely analyzed; in the *Big Data* context instead, the complete data set is explored.

- in the traditional context, first is formulated an hypothesis, then it will be verified through the analysis of the considered and selected data set portion; in the *Big Data* context, instead, the existing relation between the data contained in the target set are explored and identified.

- the analysis related to the traditional context are based on the data previously elaborated and archived in *Data Warehouses* or in *Data Marts*; in the *Big Data* context, instead, the analysis are real-time executed on progressively generated and collected data.

Basing on these factors and aspects, there are many differences between the traditional approach and the *Big Data* one as concern the analysis techniques and methodologies, too. In fact, there are many differences between the analysis based on the *Business Intelligence* and the *Big Data* context ones.

The *Business Intelligence* is based on the descriptive statistic supplied with target data have an high density information content. These data are used to execute measurements and surveys, so these operations are executed on a limited data sets portion that contains cleaned and defined through simple models.

The *Big Data* analysis, instead, use inferential statistics and identification of nonlinear systems in order to deduce and extract relations and correlations between big data sets that contains heterogeneous data.

These data are not directly correlated, so through these context analysis and models (eg. the predictive complex models), it's possible to extract and define relations between them in order to define behaviours and results.

Eric D. Brown has perfectly summed up these differences by means of these the following definition [20]:

"Business Intelligence helps you find answers to known questions. The Big Data help to find the questions I do not know but that you would like to do."

1.3.4 Big Data Representation Models

The *Big Data* context data volume and the use of unstructured heterogeneous data don't allow to use traditional *RDBMS* (*Relational Database Management System*), because they can not archive and analyze data at the same time and the related performances are not sufficiently high. Therefore, it has been necessary to adopt high scalability systems and *NoSQL* (*Not only SQL*) *DBMS* (*Database Management System*).

NoSQL is a movement which promote software systems to manage large amount of data that has to be distributed and scalable, where the data persistence is not strictly coupled with the relational model [18]. There are many types of *NoSQL* systems that can be grouped by the adopted data representation model (eg. the most used *NoSQL* systems are based on the documentary representation of data or the systems based on the object representation of the data [21]).

The *NoSQL* systems don't adopt the *ACID* protocol (this protocol define the constraints related to the data *Atomicity*, *Consistency*, *Isolation* and *Durability* of the data), because this type of systems is not relational models based. Therefore, this type of systems provides a better support to the availability, efficiency and scalability of the informations.

The *NoSQL* databases are often *schema-less*, so they don't require a well defined and fixed schema, in order to avoid the dependency on the relations and correlations between data during the querying processes, so this factor can improve performances and scalability.

Chapter 2

Architecture Design

In this chapter the description of the proposed architecture will be outlined in order to provide the related modeling starting from its definition and then detailing as concern the design of the architecture internal components and modules.

The approach used to define and describe the proposed architecture is the *Top-Down* one because the analysis has been done starting from an high abstraction level in order to go more in depth with progressive steps. In fact, the progressive increase of the detail level has been done using many *zooming* techniques in order to end at the detail level related to the internal components and entities.

This approach has been chosen because the primary requirement is to abstract from the internal detailed functionalities related to the dependences and implementation related to the *Cloud* nature of this architecture. In fact, using this approach let us model and design this architecture without influences by factors directly dependent from the technical aspects and implementations of the *Cloud* platforms and infrastructures.

Basing on the provided considerations and analysis, the main scope is to model and design an architecture which aim to be *Platform Technology Independent* in order to make it as suitable for any type of *Cloud* infrastructure and platform.

2.1 Architecture Introduction

This section will provide an introduction of the proposed architecture as concerns its main scope and scenario. Furthermore, being this architecture an *R&D* (*Research and Development*) project, there are no fixed and well defined requirements so this base scenario will be the most abstract one.

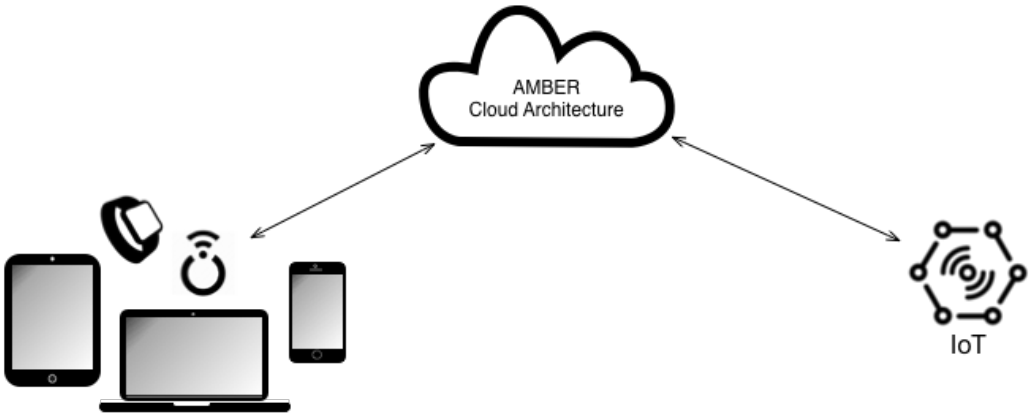

Figure 2.1: Architecture scope introduction.

Basing on Figure 2.1, the main architecture subsystems are grouped by their context and they are the following:

- **AMBER Cloud Architecture**: this subsystem represents a solution which building process has been based on the proposed architecture. One of the main aims is to obtain a *Platform Technology Independent* architecture so the focus will be set to the *Cloud* main concepts and aspects without referencing to the specific *Cloud Platforms* specifications.

- **IoT Devices**: this subsystem contains all the devices to collect data and supply the architecture by sending them to it. These devices type is related to the *Internet Of Things* context and they must have all the capabilities and features to collect data and provide them to the architecture subsystem, so these devices have to be integrated with sensors to collect data and components. Then, they will sent them

to the architecture (eg the temperature sensors will send the collected data periodically). In fact, these subsystem will be necessary in the *IoT* context solutions based on the proposed architecture.

- **Devices**: this subsystem contains all the devices to support the users and external systems as concern the architecture interaction to use its features and capabilities (eg a device can support the user by the provisioning of reports and data monitoring). These devices can also be *IoT* devices because they can be used by the users to interact with the architecture and, at the same time, the architecture can be supplied with collected by them (eg a device can collect data relating to the users' position tracking by background processes and services).

A relevant aspect related to the main scenario previously described is the modality of interaction between the subsystems, in fact, basing on Figure 2.1, it's possible to define that the interactions are fully bidirectional in order to provide more flexibility and elasticity to the architecture.

Therefore, this architecture can be used to design and build any *Cloud* solution without any constraint relating the interactions and communications between its subsystems and components.

Basing on this analysis, the main scope of the proposed architecture is to provide a valid and consistent model to design and build a *Cloud* solution. In fact, as described in the following sections, this architecture can be adapted to design any *Cloud* solution by selecting some of its internal modules and components.

This aspect is extremely important because the selecting process of some architecture internal modules and components won't compromise or lose the consistency and validity of the architecture in terms of interactions, features and capabilities.

This factor provide a great advantage because the modeling and designing phase is one of the most complex phases of the solutions building in terms of resources and time. Therefore, this approach can help these phases in order to speed up them without losing consistency and validity.

2.2 Architecture Main Overview

This section will present the description of the architecture main overview in order to provide a definition of the chosen architectural model.

As previously described, the chosen approach is the *Top-Down* one in order to describe the architecture modeling using progressive steps, starting from an high detail level and stopping when the low detail level will be reached, proceeding with progressive steps.

Figure 2.2 represents the architecture modeling main overview.

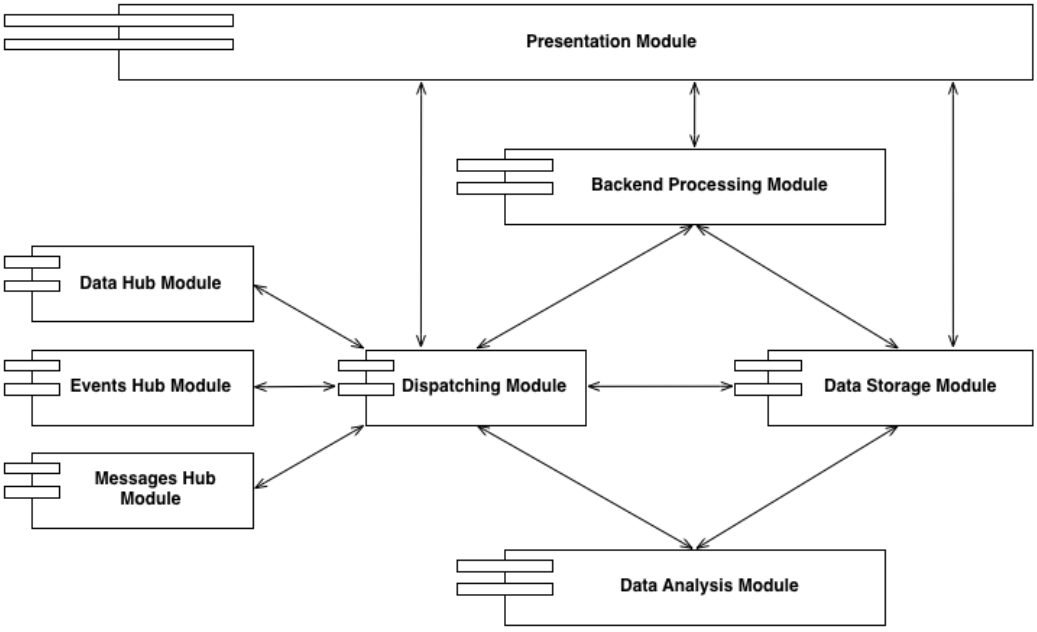

Figure 2.2: Architecture Main Overview.

The modules represented in Figure 2.2 are defined as independent in order to provide more flexibility and elasticity relating to the architecture composition. In fact, using this approach, it's possible to adopt some modules to design solutions so this architecture can be adopted for any kind of solutions to design and build.

Basing on Figure 2.2, in the next paragraphs will be described the internal modules in terms of scopes, features and capabilities.

The architecture internal modules are the following:

- **Data Hub Module**: module to support the communications and interactions between the architecture based solution and the external devices and systems as concern the data transmissions and related communications. Therefore, this module will act as data flow manager so as to focus its work in this specific context because the concept and definition of the data is one of the most important scopes of many solutions.

- **Events Hub Module**: module to support the communications and interactions between the architecture based solution and the external devices and systems as concern the events occurrence informations transmissions and related communications. Therefore, this module will act as events occurrence informations flow manager in order to focus its work in this specific context because this is one of the most important scopes of many solutions.

- **Messages Hub Module**: module to support the communications and interactions between the architecture based solution and the external devices and systems as concern the messages transmissions and related communications. Therefore, this module will act as messages flow manager so as to focus its work in this specific context because this is one of the most important scopes of many solutions.

- **Dispatching Module**: module to support the management, dispatching and routing of the data relating to the architecture internal modules. In fact, this module will manage the communication and transmission of all kind of data between the architecture modules.

- **Data Analysis Module**: module to support and execute the data analysis based on many different approaches, technologies and methodologies selected basing on the solutions requirements. This is module is particularly important because data analysis is the main scope of several solutions. Therefore, this module will be supported by many tools, infrastructures and technologies to analyze the target data depending on the target solution requirements.

- **Data Storage Module**: module to manage the persistent data storage with high levels of reliability and consistency. This module will be supported by many tools, infrastructures and technologies to manage the data storage basing on the target solution requirements. Therefore, this module will be one of the most important modules in the proposed architecture based solutions because the data storage is one of the modern solutions primary scopes.

- **Backend Processing Module**: module to support and manage the backend processes required by the solutions needs and scopes. This module will be supported by many tools, infrastructures and technologies to manage the backend processes basing on the target solution requirements.

- **Presentation Module**: module to provide data, features and capabilities related to the architecture based solutions. It will provide tools, services and applications to manage the solutions capabilities and features. The users will use this module tools, applications and services to obtain data informations and manage the built solution (eg the users can obtain data reports relating to the analyzed data).

The architecture modules scopes, features and capabilities has been introduced and defined with an high abstraction level in order to avoid the focus on implementation and technological details related to their internal composition and the specifications related to the specific target solutions to design and built.

In the following sections will be provided the description of all the introduced modules and related internal composition in order to give an holistic model of them in terms of structure, interaction and behaviour.

2.3 Data Hub Module

This section will provide the definition and description related to the *Data Hub Module*. This is the module devoted to the communications and interactions support between the architecture based solution and the external devices and systems as concern the data transmissions and related communications.

These types of communications and interactions will be provided by this module because there are many contexts and types of data in terms of definition, format and structure.

This modeling choice has been done on the basis of the specific needs for the performances and characteristics that the communications and interactions must have. In fact, the definition of a specific module to provide these features and capabilities allows to obtain high performances relating to transmission speed, maintainability and reliability.

These advantages couldn't be provided in case of multi-purpose module or multi-context module.

Figure 2.3 represents this module architectural model.

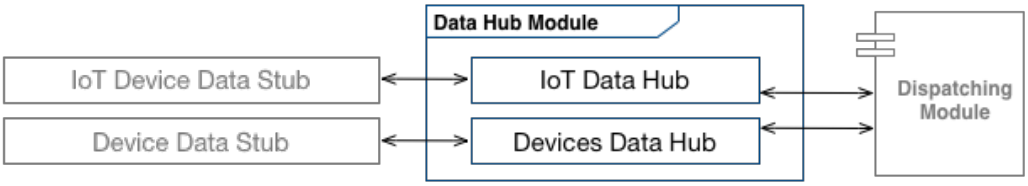

Figure 2.3: Data Hub Architectural Model.

On the basis of Figure 2.3, it's possible to infers that in this module architectural model there are many other external components in order to provide a better description of this module and related scopes.

In fact, in the target figure there are the following external components:

- **IoT Device Data Stub**: architecture external component located on the *Internet Of Things* devices. It's dedicated to manage the communications, interactions and transmissions between the *IoT* devices and the *Data Hub Module*.
- **Device Data Stub**: architecture external component located on the *general purpose* devices. It's devoted to manage the communications, interactions and transmissions between the *general purpose* devices and the *Data Hub Module*.
- **Dispatching Module**: architecture internal module to support the management, dispatching and routing of the data relating to the architecture internal modules.

The architecture external components located on the external devices and external systems are not interested by this architecture because their unique feature is to transmit data to this module by satisfaction of the constraints and specifications provided by this module.

On the basis of this factor, the architecture and its outbound modules will be more elastic and flexible in order to follow the guidelines related to the proposed architecture.

In the following paragraphs will be provided the definition and the description related to this module internal components so as to focus on their features and capabilities.

In the *Data Hub Module* module there are the following internal components:

- **IoT Data Hub**: component to support and manage the communication, interaction and transmission between this module and the external *IoT* devices.
- **Devices Data Hub**: component to support and manage the communication, interaction and transmission between this module and the external *general purpose* devices.

The defined components allow to model, design and manage this module independently from the external devices, systems and other architecture internal modules because the design of its internal specific entities can not be treated as base entities in order to manage this module independently from their design details.

In the following sections will be described the feature and capabilities of the *Data Hub Module* internal components in order to provide their complete overview.

2.3.1 IoT Data Hub

The *IoT Data Hub* main scope is to support and manage the communication, interaction and transmission between this module and the external *Internet of Things* devices specific to the *IoT* context.

This component has to communicate and interact with the components located on the architecture external *IoT* devices and systems and their *stub* components.

This component has to communicate and interact also with the *Dispatching Module* in order to provide the collected data to other architecture internal modules.

On the basis of this communication and interaction scenario, this component has to communicate and interact with other components in a bidirectional way.

Therefore, this component has to manage incoming and outgoing communications so as to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined in order to provide this component as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.3.2 Devices Data Hub

The *Devices Data Hub* main scope is to support and manage the communication, interaction and transmission between this module and the external *general purpose* devices, not specific to any particular context.

This component has to communicate and interact with the components located on the architecture external *general purpose* devices and systems and their *stub* components.

This component has to communicate and interact also with the *Dispatching Module* so as to provide the collected data to other architecture internal modules.

Basing on this communication and interaction scenario, this component has to communicate and interact with other components in a bidirectional way. Therefore, this component has to manage incoming and outgoing communications in order to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined so as to provide this component as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.4 Events Hub Module

This section will present the definition and description related to the *Events Hub Module*. This is the module devoted to the communications and interactions between the architecture based solution and the external devices and systems as concern the events occurrences informations transmissions and related communications.

These types of communications and interactions will be provided by this module because there are many contexts and types of events occurrences informations in terms of definition, format and structure.

This modeling choice has been done basing on the need specific for the performances and characteristics that the communications and interactions must have. In fact, the definition of a specific module to provide these features and capabilities allows to obtain high performances relating to transmission speed, maintainability and reliability.

These advantages couldn't be provided in case of multi-purpose module or multi-context module.

Figure 2.4 represents this module architectural model.

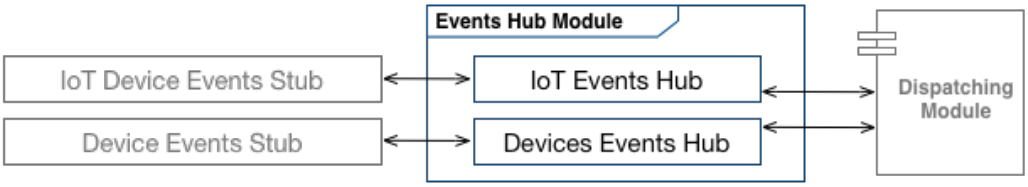

Figure 2.4: Events Hub Architectural Model.

On the basis of Figure 2.4 it's possible to infers that in this module architectural model there are many other external components so as to provide a better description of this module and related scopes.

In fact, in the target figure there are the following external components:

- **IoT Device Events Stub**: architecture external component located on the *Internet Of Things* devices. It's dedicated to the manage the communications, interactions and transmissions between the *IoT* devices and the *Event Hub Module*.

- **Device Events Stub**: architecture external component located on the *general purpose* devices. It's devoted to manage the communications, interactions and transmissions between the *general purpose* devices and the *Event Hub Module*.

- **Dispatching Module**: architecture internal module to support the management, dispatching and routing of the events occurrences informations relating to the architecture internal modules.

The architecture external components located on the external devices and external systems are not interested by this architecture because their unique feature is to transmit the events occurrences informations to this module by satisfaction of the constraints and specifications provided by this module.

On the basis of this factor, the architecture and its outbound modules will be more elastic and flexible in order to follow the guidelines related to the proposed architecture.

In the following paragraphs will be provided the definition and the description related to this module internal components so as to focus on their features and capabilities.

In the *Events Hub Module* module there are the following internal components:

- **IoT Events Hub**: component to support and manage the communication, interaction and transmission between this module and the external IoT devices.

- **Devices Events Hub**: component to support and manage the communication, interaction and transmission between this module and the external *general purpose* devices.

The defined components allow to model, design and manage this module independently from the external devices, systems and other architecture internal modules because the design of its internal specific entities can not be treated as base entities in order to manage this module independently from their design details.

In the following sections will be described the feature and capabilities of the *Event Hub Module* internal components so as to provide their complete overview.

2.4.1 IoT Events Hub

The *IoT Events Hub* main scope is to support and manage the communication, interaction and transmission between this module and the external *Internet of Things* devices specific to the *IoT* context.

This component has to communicate and interact with the components located on the architecture external *IoT* devices and systems and their *stub* components.

This component has to communicate and interact also with the *Dispatching Module* in order to provide the collected data to other architecture internal modules.

On the basis of this communication and interaction scenario, this component has to communicate and interact with other components in a bidirectional way.

Therefore, this component has to manage incoming and outgoing communications so as to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined in order to provide this component as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.4.2 Devices Events Hub

The *Devices Events Hub* main scope is to support and manage the communication, interaction and transmission between this module and the external *general purpose* devices, not specific to any particular context.

This component has to communicate and interact with the components located on the architecture external *general purpose* devices and systems and their *stub* components.

This component has to communicate and interact also with the *Dispatching Module* so as to provide the collected data to other architecture internal modules.

Basing on this communication and interaction scenario, this component has to communicate and interact with other components in a bidirectional way.

Therefore, this component has to manage incoming and outgoing communications in order to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined in order to provide this component as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.5 Messages Hub Module

In this section will be provided the definition and description related to the *Messages Hub Module*. This is the module devoted to support the communications and interactions between the architecture based solution and the external devices and systems as concern the messages transmissions and related communications.

These types of communications and interactions will be provided by this module because there are many contexts and types of messages in terms of definition, format and structure.

This modeling choice has been done basing on the need specific for the performances and characteristics that the communications and interactions must have. In fact, the definition of a specific module to provide these features and capabilities allows to obtain high performances relating to transmission speed, maintainability and reliability.

These advantages couldn't be provided in case of multi-purpose module or multi-context module.

Figure 2.5 represents this module architectural model.

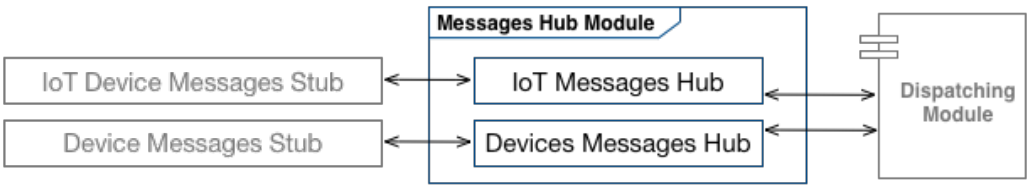

Figure 2.5: Messages Hub Architectural Model.

On the basis of Figure 2.5, it's possible to infers that in this module architectural model there are many other external components so as to provide a better description of this module and related scopes.

In fact, in the target figure there are the following external components:

- **IoT Messages Stub**: architecture external component located on the *Internet Of Things* devices. It's dedicated to the manage the communications, interactions and transmissions between the *IoT* devices and the *Messages Hub Module*.
- **Device Messages Stub**: architecture external component located on the *general purpose* devices. It's devoted to manage the communications, interactions and transmissions between the *general purpose* devices and the *Messages Hub Module*.
- **Dispatching Module**: architecture internal module to support the management, dispatching and routing of the messages relating to the architecture internal modules.

The architecture external components located on the external devices and external systems are not interested by this architecture because their unique feature is to transmit data to this module by satisfaction of the constraints and specifications provided by this module.

Basing on this factor, the architecture and its outbound modules will be more elastic and flexible in order to follow the guidelines related to the proposed architecture.

In the following paragraphs will be provided the definition and the description related to this module internal components so as to focus on their features and capabilities.

In the *Messages Hub Module* module there are the following internal components:

- **IoT Messages Hub**: component to support and manage the communication, interaction and transmission between this module and the external *IoT* devices.
- **Devices Messages Hub**: component to support and manage the communication, interaction and transmission between this module and the external *general purpose* devices.

The defined components allow to model, design and manage this module independently from the external devices, systems and other architecture internal modules because the design of its internal specific entities can not be treated as base entities in order to manage this module independently from their design details.

In the following sections will be described the feature and capabilities of the *Messages Hub Module* internal components in order to provide their complete overview.

2.5.1 IoT Messages Hub

The *IoT Messages Hub* main scope is to support and manage the communication, interaction and transmission between this module and the external *Internet of Things* devices specific to the *IoT* context.

This component has to communicate and interact with the components located on the architecture external *IoT* devices and systems and their *stub* components.

This component has to communicate and interact also with the *Dispatching Module* so as to provide the collected messages to other architecture internal modules.

On the basis of this communication and interaction scenario, this component has to communicate and interact with other components in a bidirectional way.

Therefore, this component has to manage incoming and outgoing communications in order to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined so as to provide this component as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.5.2 Devices Messages Hub

The *Devices Messages Hub* main scope is to support and manage the communication, interaction and transmission between this module and the external *general purpose* devices, not specific to any particular context.

This component has to communicate and interact with the components located on the architecture external *general purpose* devices and systems

and their *stub* components.

This component has to communicate and interact also with the *Dispatching Module* in order to provide the collected messages to other architecture internal modules.

Basing on this communication and interaction scenario, this component has to communicate and interact with other components in a bidirectional way.

Therefore, this component has to manage incoming and outgoing communications so as to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined in order to provide this component as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.6 Dispatching Module

This section will provide the definition and description related to the *Dispatching Module*. This is the module devoted to support the management, dispatching and routing of the data relating to the architecture internal modules.

This module has a primary role in the proposed architecture because all the internal components main interactions and communications depends on it and it has also to maintain them in order to guarantee reliability and stability.

These types of communications and interactions will be provided by this module because there are many contexts and types of messages in terms of definition, format and structure.

This modeling choice has been done basing on the need specific for the performances and characteristics that the communications and interactions must have. In fact, the definition of a specific module to provide these features and capabilities allows to obtain high performances relating to transmission speed, maintainability and reliability.

These advantages couldn't be provided in case of multi-purpose module or multi-context module.

Figure 2.6 represents this module architectural model.

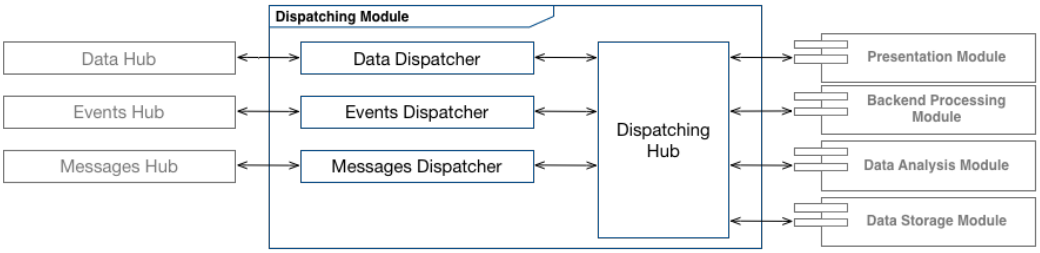

Figure 2.6: Dispatching Module Architectural Model.

Basing on Figure 2.6, it's possible to infers that in this module architectural model there are many other external components so as to provide a better description of this module and related scopes.

In fact, in the target figure there are the following external components:

- **Data Hub Module**: module to support the communications and interactions between the architecture based solution and the external devices and systems as concern the data transmissions and related communications (reference to the section 2.3).
- **Events Hub Module**: module to support the communications and interactions between the architecture based solution and the external devices and systems as concern the events occurrence informations transmissions and related communications (reference to the section 2.4).
- **Messages Hub Module**: module to support the communications and interactions between the architecture based solution and the external devices and systems as concern the messages transmissions and related communications (reference to the section 2.5).
- **Data Analysis Module**: module to support and execute the data analysis based on many different approaches, technologies and methodologies selected basing on the solutions requirements.
- **Data Storage Module**: module to manage the persistent data storage with high levels of reliability and consistency.

- **Backend Processing Module**: module to support and manage the backend processes required by the solutions needs and scopes.

- **Presentation Module**: module to provide data, features and capabilities related to the architecture based solutions. It will provide tools, services and applications to manage the solutions capabilities and features.

This module external components reported in Figure 2.6 are not interested by this module analysis because their definition and description have been provided in the others sections so as to focus on their independence.

On the basis of this factor and modularization, this module will be more elastic and flexible in order to follow the guidelines related to the proposed architecture.

In the following paragraphs will be provided the definition and the description related to this module internal components so as to focus on their features and capabilities.

In details, in the *Dispatching Hub Module* module there are the following internal components:

- **Data Dispatcher**: component to managed and support the reception, sending and routing of all the informations related to the *Data Hub Module* as concerns the incoming and outgoing data flow.

- **Events Dispatcher**: component to managed and support the reception, sending and routing of all the informations related to the *Events Hub Module* as concerns the incoming and outgoing of the flow related to the events occurrences informations.

- **Messages Dispatcher**: component to managed and support the reception, sending and routing of all the informations related to the *Messages Hub Module* as concerns the incoming and outgoing flow related to the messages.

- **Dispatching Hub**: component to support and manage the communications, interactions and transmissions between the architecture outbound dispatcher components (*Data Dispatcher*, *Events Dispatcher* and *Messages Dispatcher*) and the other architecture internal modules. This component acts as broker for all the main communications,

interactions and transmissions related to the architecture modules informations. In fact, it's involved also in the communications between other architecture internal modules, not only between them and the outbound dispatcher components.

The defined components allow to model, design and manage this module independently from the other architecture internal modules because the design of its internal specific entities can not be treated as base entities so as to manage this module independently from their design details.

In the following sections will be described the feature and capabilities of the *Dispatching Module* internal components in order to provide their complete overview.

2.6.1 Data Dispatcher

The *Data Dispatcher* main scope is to managed and support the reception, sending and routing of all the informations related to the *Data Hub Module* as concerns the incoming and outgoing data flow relating to that module.

This component has to communicate and interact with the *Data Hub Module* and the *Dispatching Module* so as to provide the collected data to other architecture internal modules.

Therefore, this is the unique access point for this specific data flow related to the architecture external devices and systems.

On the basis of this communication and interaction scenario, this component has to communicate and interact with other components in a bidirectional way.

Therefore, this component has to manage incoming and outgoing communications in order to provide a more flexible component that make this component suitable to any kind of interaction and communication.

All the design and development details and specifications won't be defined so as to provide this component as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.6.2 Events Dispatcher

The *Events Dispatcher* main scope is to managed and support the reception, sending and routing of all the informations related to the *Events Hub Mod-*

ule as concerns the incoming and outgoing flow of the events occurrences informations relating to that module.

This component has to communicate and interact with the *Events Hub Module* and the *Dispatching Module* so as to provide the collected informations to other architecture internal modules.

Therefore, this is the unique access point for this specific informations flow related to the architecture external devices and systems.

Basing on this communication and interaction scenario, this component has to communicate and interact with other components in a bidirectional way.

Therefore, this component has to manage incoming and outgoing communications in order to provide a more flexible component that make this component suitable to any kind of interaction and communication.

All the design and development details and specifications won't be defined so as to provide this component as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.6.3 Messages Dispatcher

The *Messages Dispatcher* main scope is to managed and support the reception, sending and routing of all the informations related to the *Messages Hub Module* as concerns the incoming and outgoing messages flow relating to that module.

This component has to communicate and interact with the *Messages Hub Module* and the *Dispatching Module* in order to provide the collected messages to other architecture internal modules.

Therefore, this is the unique access point for this specific messages flow related to the architecture external devices and systems.

On the basis of this communication and interaction scenario, this component has to communicate and interact with other components in a bidirectional way.

Therefore, this component has to manage incoming and outgoing communications so as to provide a more flexible component that make this component suitable to any kind of interaction and communication.

All the design and development details and specifications won't be defined in order to provide this component as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.6.4 Dispatching Hub

The *Dispatching Hub* main scope is to support and manage the communications, interactions and transmissions between the architecture outbound dispatcher components (*Data Dispatcher*, *Events Dispatcher* and *Messages Dispatcher*) and the other architecture internal modules.

This component acts as broker for all the main communications, interactions and transmissions related to the architecture modules informations. In fact, it's involved also in the communications between other architecture internal modules, not only between them and the outbound dispatchers.

Basing on this communication and interaction scenario, this component has to communicate and interact with other components in a bidirectional way.

Therefore, this component has to manage incoming and outgoing communications so as to provide a more flexible component that make this component suitable to any kind of interaction and communication.

All the design and development details and specifications won't be defined in order to provide this component as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.7 Data Analysis Module

This section will present the definition and description related to the *Data Analysis Module*. This is the module devoted to support and execute the data analysis based on many different approaches, technologies and methodologies selected basing on the solutions requirements.

This is module is particularly important because in most solutions have the data analysis as main scope. Therefore, this module will be supported by many tools, infrastructures and technologies to analyze the target data depending on the target solution requirements.

This module has to manage large amount of many contexts and types of data in terms of definition, format and structure. This modeling choice has been done basing on the need specific for the performances and characteristics that the communications and interactions must have.

Thus, the definition of a specific module to provide these features and capabilities allows to obtain high performances relating to management and elaboration speed, maintainability and reliability.

These advantages couldn't be provided in case of multi-purpose module or multi-context module.

Figure 2.7 represents this module architectural model.

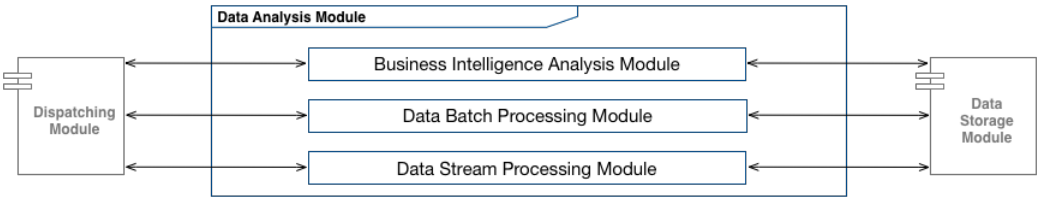

Figure 2.7: Data Analysis Architectural Model.

On the basis of Figure 2.7, it's possible to infers that in this module architectural model there are many other external components in order to provide a better description of this module and related scopes.

In fact, in the target figure there are the following external components:

- **Dispatching Module**: module to support the management, dispatching and routing of the data relating to the architecture internal modules (reference to the section 2.6).
- **Data Storage Module**: module to manage the persistent data storage with high levels of reliability and consistency. This module will be supported by many tools, infrastructures and technologies to manage the data storage basing on the target solution requirements.

This module external components reported in Figure 2.7 are not interested by this module analysis because their definition and description have been provided in the others sections so as to focus on their independence.

On the basis of this factor and modularization, this module will be more elastic and flexible in order to follow the guidelines related to the proposed architecture.

In the following paragraphs will be provided the definition and the description related to this module internal components in order to focus on their features and capabilities.

In details, in the *Data Analysis Module* module there are the following internal components:

- **Business Intelligence Analysis Module**: internal module to manage, support and execute the *Business Intelligence* (commonly named as *BI*) analysis relating to the architecture modules managed and collected data.
- **Data Batch Analysis Module**: internal module to manage, support and execute the data *Batch* analysis relating to the architecture modules managed and collected data.
- **Data Stream Analysis Module**: internal module to manage, support and execute the data *Stream* analysis relating to the architecture modules managed and collected data.

The defined modules allow to model, design and manage this module independently from the other architecture internal modules because the design of its internal specific entities can not be treated as base entities so as to manage this module independently from their design details.

Thus, the described modules are not components so their internal complexity depends on the related implementations, tools and used technologies.

In the following sections will be described the feature and capabilities of the *Data Analysis Module* internal components in order to provide their complete overview.

2.7.1 Business Intelligence Analysis Module

The *Business Intelligence Analysis Module* has to support and manage the *Business Intelligence* analysis execution relating to the solutions data.

This part of the architecture has been modeled and designed as a module due to its complexity, so the internal composition can not be defined as a component because it has to provide many tools, technologies and capabilities in order to support this kind of data analysis.

This module has to communicate and interact with the *Dispatching Hub Module* so as to provide the analysis capabilities, features and results about the data analysis to other architecture internal modules.

It has to communicate and interact also with the *Data Storage Module* in order to storage the data analysis results in order to maintain them in a persistent and consistent mode.

This module will receive and retrieve the data to analyze from the *Dispatching Hub Module* (data reception from the other architecture internal model) and the *Data Storage Module* (retrieve the data from the storage).

Thus, this is the unique access point for this specific data flow related to the solutions data analysis.

On the basis of this communication and interaction scenario, this module has to communicate and interact with other components in a bidirectional way.

Therefore, this module has to manage incoming and outgoing communications in order to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined so as to provide this module as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.7.2 Data Batch Analysis Module

The *Data Batch Analysis Module* has to support and manage the *Batch* analysis execution relating to the solutions data.

Therefore this module is focused on the execution of this kind of data analysis in order to obtain high performances, reliability and efficiency relating to large amounts of data.

This part of the architecture has been modeled and designed as a module due to its complexity, so the internal composition can not be defined as a component because it has to provide many tools, technologies and capabilities so as to support this kind of data analysis.

This module has to communicate and interact with the *Dispatching Hub Module* in order to provide the analysis capabilities, features and results about the data analysis to other architecture internal modules.

It has to communicate and interact also with the *Data Storage Module* so as to storage the data analysis results in order to maintain them in a persistent and consistent mode.

This module will receive and retrieve the data to analyze from the *Dispatching Hub Module* (data reception from the other architecture internal

model) and the *Data Storage Module* (retrieve the data from the storage).

Therefore, this is the unique access point for this specific data flow related to the solutions data analysis.

Basing on this communication and interaction scenario, this module has to communicate and interact with other components in a bidirectional way.

Therefore, this module has to manage incoming and outgoing communications in order to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined so as to provide this module as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.7.3 Data Stream Analysis Module

The *Data Stream Analysis Module* has to support and manage the *Real-Time Stream* analysis execution relating to the solutions data.

Therefore this module is focused on the execution of this kind of data analysis so as to obtain high performances, reliability and efficiency relating to portions and subpart of large amounts of data.

These analysis execution has to produce results obtained by the analysis of target data in short time periods in order to provide them with high frequency.

This part of the architecture has been modeled and designed as a module due to its complexity, so the internal composition can not be defined as a component because it has to provide many tools, technologies and capabilities in order to support this kind of data analysis.

This module has to communicate and interact with the *Dispatching Hub Module* so as to provide the analysis capabilities, features and results about the data analysis to other architecture internal modules.

It has to communicate and interact also with the *Data Storage Module* in order to storage the data analysis results so as to maintain them in a persistent and consistent mode.

This module will receive and retrieve the data to analyze from the *Dispatching Hub Module* (data reception from the other architecture internal model) and the *Data Storage Module* (retrieve the data from the storage).

Therefore, this is the unique access point for this specific data flow related to the solutions data analysis.

On the basis of this communication and interaction scenario, this module has to communicate and interact with other components in a bidirectional way.

Therefore, this module has to manage incoming and outgoing communications in order to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined in order to provide this module as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.8 Data Storage Module

In this section will be provided the definition and description related to the *Data Storage Module*. This is the module devoted to manage the persistent data storage with high levels of reliability and consistency.

This module will be supported by many tools, infrastructures and technologies to manage the data storage basing on the target solution requirements.

This module has a primary role in the proposed architecture because all the architecture based solutions data will be maintained and managed by this module.

All the modern solutions are oriented to the concept and context of data as concerns the management, maintenance and related analysis processes suppling.

Thus, this module will be one of the most important modules in the solutions based on the proposed architecture.

The *Data Storage Module* has to manage and maintain large amounts of data with many different type, definition and structure (eg it has to manage structured and unstructured data). All the managed data has to be retrieved in order to be analyzed and represented to other modules.

All these types of informations and data will be managed by this module because there are many contexts and types of data in terms of definition, format and structure.

This modeling choice has been done basing on the need specific for the performances and characteristics that the communications and interactions must have.

Therefore, the definition of a specific module to provide these features and capabilities allows to obtain high performances relating to management, maintainability and reliability.

These advantages couldn't be provided in case of multi-purpose module or multi-context module.

Figure 2.8 represents this module architectural model.

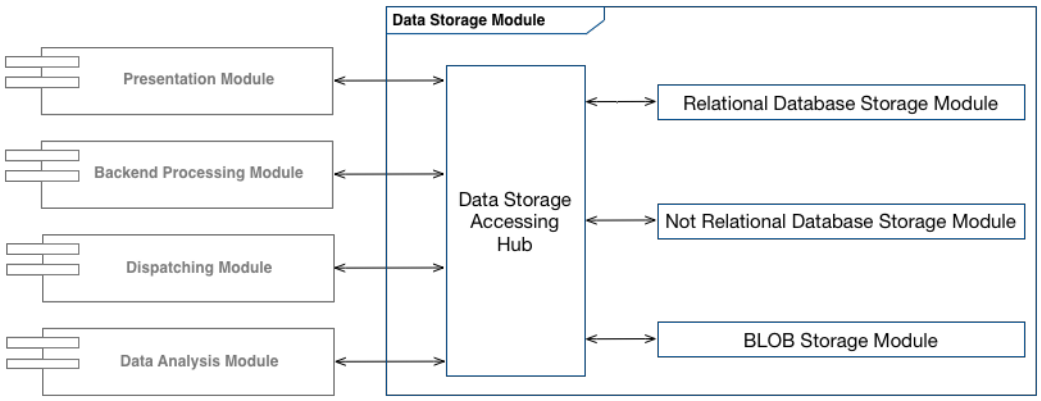

Figure 2.8: Data Storage Module Architectural Model.

On the basis of Figure 2.8, it's possible to infers that in this module' architectural model there are many other external components so as to provide a better description of this module and related scopes.

In fact, in the target figure there are the following external components:

- **Presentation Module**: module to provide data, features and capabilities related to the architecture based solutions. It will provide tools, services and applications to manage the solutions capabilities and features.

- **Backend Processing Module**: module to support and manage the backend processes required by the solutions needs and scopes.

- **Dispatching Module**: module to support the management, dispatching and routing of the data relating to the architecture internal modules (reference to section 2.6).

- **Data Analysis Module**: module to support and execute the data analysis based on many different approaches, technologies and methodologies selected basing on the solutions requirements (reference to section 2.7).

This module external components reported in Figure 2.8 are not interested by this module analysis because their definition and description have been provided in the others sections so as to focus on their independence.

On the basis of this factor and modularization, this module will be more elastic and flexible in order to follow the guidelines related to the proposed architecture.

In the following paragraphs will be provided the definition and the description related to this module internal components in order to focus on their features and capabilities.

In details, in the *Data Storage Module* module there are the following internal components:

- **Relational Database Storage Module**: module to manage the persistent storage of the data basing on *Relational DataBase Management System*. This module target data are structured and well-defined modeled in terms of definition and type so it's possible to deduce and build a *Relational Model*.

- **Not Relational Database Storage Module**: module to manage the persistent storage of the data basing on *Non Relational DataBase Management System*. This module target data are partially unstructured and they are not completely modeled in terms of definition and type so it's not possible to deduce and build a *Relational Model*.

- **BLOB Storage Module**: module to manage the persistent storage of the data basing on *BLOB Storage Management System*. This module target data are unstructured, heterogeneous without any constraints relating to their format and type. This kind of data can have extremely large dimension and they can not be directly managed by a *DataBase Management System*.

- **Data Storage Accessing Hub**: component to support and manage the communications, interactions and transmissions between the other architecture modules and the other *Data Storage Module* internal modules (*Relational Database Storage Module*, *Not Relational Database Storage Module* and *BLOB Storage Module*). This component acts as broker for all the main communications, interactions and transmissions related to the architecture modules data.

The defined modules and components allow to model, design and manage this module independently from the other architecture internal modules because the design of its internal specific entities can not be treated as base entities so as to manage this module independently from their design details.

In the following sections will be described the feature and capabilities of the *Data Storage Module* internal components and modules in order to provide their complete overview.

2.8.1 Relational Database Storage Module

The *Relational Database Storage Module* has to manage the persistent storage of the data basing on *Relational DataBase Management System*.

This module is particularly important because, as defined in the section 2.7 relatively to the primary role of the data in modern solutions context.

This part of the architecture has been modeled and designed as a module due to its complexity, so the internal composition can not be defined as a component because it has to provide many tools, technologies and capabilities in order to support this kind of data storage relatively to the extended pool of technologies and components as concern the *Relational* data.

This module has to communicate and interact with the *Data Storage Accessing Hub* so as to manage the communications, interactions and transmissions between this module and the other architecture modules in order to provide the contained data to them.

Basing on this communication and interaction scenario, this module has to communicate and interact with other components in a bidirectional way.

Therefore, this module has to manage incoming and outgoing communications in order to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined so as to provide this module as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.8.2 Not Relational Database Storage Module

The *Not Relational Database Storage Module* has to manage the persistent storage of the data basing on *Not Relational DataBase Management System*.

This module is particularly important because, as defined in the section 2.7 relatively to the primary role of the data in modern solutions context.

This part of the architecture has been modeled and designed as a module due to its complexity, so the internal composition can not be defined as a component because it has to provide many tools, technologies and capabilities so as to support this kind of data storage relatively to the extended pool of technologies and components as concern the *Relational* data.

This module has to communicate and interact with the *Data Storage Accessing Hub* in order to manage the communications, interactions and transmissions between this module and the other architecture modules so as to provide the contained data to them,

Basing on this communication and interaction scenario, this module has to communicate and interact with other components in a bidirectional way.

Therefore, this module has to manage incoming and outgoing communications in order to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined so as to provide this module as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.8.3 BLOB Storage Module

The *BLOB Storage Module* has to manage the persistent storage of the data basing on *BLOB Storage Management System*. The target data are unstructured, heterogeneous without any constraints relating to their format, dimension and type.

This module is particularly important because, as defined in the section 2.7 relatively to the primary role of the data in modern solutions context.

This part of the architecture has been modeled and designed as a module due to its complexity, so the internal composition can not be defined as a component because it has to provide many tools, technologies and capabilities so as to support this kind of data storage relatively to the extended pool of technologies and components as concern the *Relational* data.

This module has to communicate and interact with the *Data Storage Accessing Hub* in order to manage the communications, interactions and transmissions between this module and the other architecture modules so as to provide the contained data to them,

On the basis of this communication and interaction scenario, this module has to communicate and interact with other components in a bidirectional way.

Therefore, this module has to manage incoming and outgoing communications in order to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined so as to provide this module as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.8.4 Data Storage Accessing Hub

The *Data Storage Accessing Hub* has to support and manage the communications, interactions and transmissions between the other architecture modules and the other *Data Storage Module* internal modules (*Relational Database Storage Module*, *Not Relational Database Storage Module* and *BLOB Storage Module*).

Therefore, this is the unique access point for the flow related to the data storage and correlated phases and operations. In fact, it has been necessary to define this component with this specific scope, so it's possible to execute all the preliminary operations on the target data.

Therefore, it's possible to define these operations so as to improve the performances and capabilities of the *Data Storage Module*.

This component acts as broker for all the communications, interactions and transmissions related to the architecture modules data. In fact, it's involved also in the communications between other architecture internal modules that have to retrieve and storage the target data.

On the basis of this communication and interaction scenario, this component has to communicate and interact with other components in a bidirectional way.

Therefore, this component has to manage incoming and outgoing communications in order to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined so as to provide this component as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.9 Backend Processing Module

This section will provide the definition and description related to the *Backend Processing Module*. This is the module devoted to support and manage the backend processes required by the solutions needs and scopes.

This module will be supported by many tools, infrastructures and technologies to manage the backend processes basing on the target solution requirements.

This module has been designed and modeled in order to provide a specific pool of components to contain, manage and support the backend processes required by the *Cloud* solutions based on this architecture.

Therefore, it's possible to provide a standard whose allows the designers and developers to build solutions specific backend processes independently of the management system and workflow.

Thus, basing on this architecture, any backend process can be manage by this specific module if they will satisfy the constraint and standard provided by this module.

The *Backend Processing Module* is particularly important because many modern solutions need and require to build and provide backend processes to execute heavy load operations.

In this kind of scenarios, this module has to manage, support and maintain large amounts of communications and interactions related to the backend processes.

Figure 2.9 represents this module architectural model.

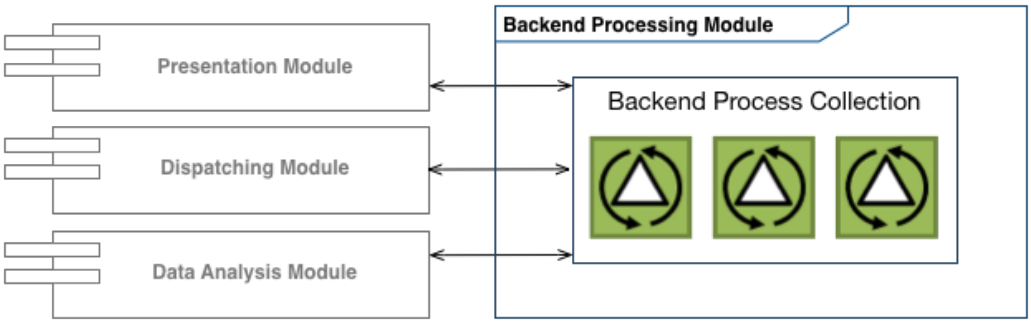

Figure 2.9: Backend Processing Module Architectural Model.

Basing on Figure 2.9, it's possible to infers that in this module architectural model there are many other external components so as to provide a better description of this module and related scopes.

In fact, in the target figure there are the following external components:

- **Presentation Module**: module to provide data, features and capabilities related to the architecture based solutions. It will provide tools, services and applications to manage the solutions capabilities and features.

- **Dispatching Module**: module to support the management, dispatching and routing of the data relating to the architecture internal modules (reference to section 2.6).

- **Data Analysis Module**: module to support and execute the data analysis based on many different approaches, technologies and methodologies selected basing on the solutions requirements (reference to section 2.7).

This module external components reported in Figure 2.9 are not interested by this module analysis because their definition and description have been provided in the others sections so as to focus on their independence.

Basing on this factor and modularization, this module will be more elastic and flexible in order to follow the guidelines related to the proposed architecture.

In the following paragraphs will be provided the definition and the description related to this module internal components so as to focus on their features and capabilities.

In details, in the *Backend Processing Module* there is only one internal component: the **Backend Process Collection**. This component has to contain and manage the backend processes collection related to the specific architecture based solution.

The defined component allows to model, design and manage this module independently from the other architecture internal modules because the design of its internal specific entity can not be treated as base entity in order to manage this module independently from its design details.

In the following section will be described the feature and capabilities of the *Backend Processing Module* internal component so as to provide its complete overview.

2.9.1 Backend Process Collection

The *Backend Process Collection* has to contain and manage the backend processes collection related to the specific architecture based solution.

This component has also to manage and support the communications, interactions and transmissions between the other architecture modules and the backend processes available.

Therefore, this is the unique access point for the flow related to the backend processes management and execution.

This component acts as broker for all the communications, interactions and transmissions related to the architecture modules requests and results relating to the backend processes execution, support and management. In fact, it's involved also in the communications between other architecture internal modules that have to execute a backend process or the internal modules that have to retrieve backend processes execution result.

The *Backend Process Collection* is particularly important because many modern solutions need and require to build and provide backend processes to execute heavy load operations.

In this kind of scenarios, this component has to manage, support and maintain large amounts of communications and interactions related to the backend processes so as to support the features and capabilities provided by the *Backend Processing Module*.

In details, this component has to communicate and interact with the *Presentation Module*, *Dispatching Module* and the *Data Analysis Module* in order to allow them to require and manage the backend processes execution and related results.

On the basis of this communication and interaction scenario, this component has to communicate and interact with other components in a bidirectional way.

Therefore, this component has to manage incoming and outgoing communications so as to provide a more flexible component that make this component suitable to any kind of interaction and communication mode.

All the design and development details and specifications won't be defined in order to provide this component as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.10 Presentation Module

In this section will be provided the definition and description related to the *Presentation Module*.

This is the module devoted to provide data, features and capabilities related to the architecture based solutions. It has to provide tools, services and applications to manage the solutions capabilities and features.

Therefore, the users can use this module tools, applications and services to obtain data informations and manage the built solution (eg the users can obtain data reports relating to the analyzed data). Therefore, this module has a primary role in the proposed architecture.

All the solutions data, informations, features and capabilities have to be managed, supported and provided by this module because there are many contexts and types of data, features and capabilities in terms of definition, structure, format and scope, so it has be necessary to model and design this module for this specific scope.

This modeling choice has been done basing on the need specific for the performances and characteristics that this module must have.

Thus, the definition of a specific module to provide these features and capabilities allows to obtain high performances relating to transmission and communication speed, maintainability and reliability.

These advantages couldn't be provided in case of multi-purpose module or multi-context module.

Figure 2.10 represents this module architectural model.

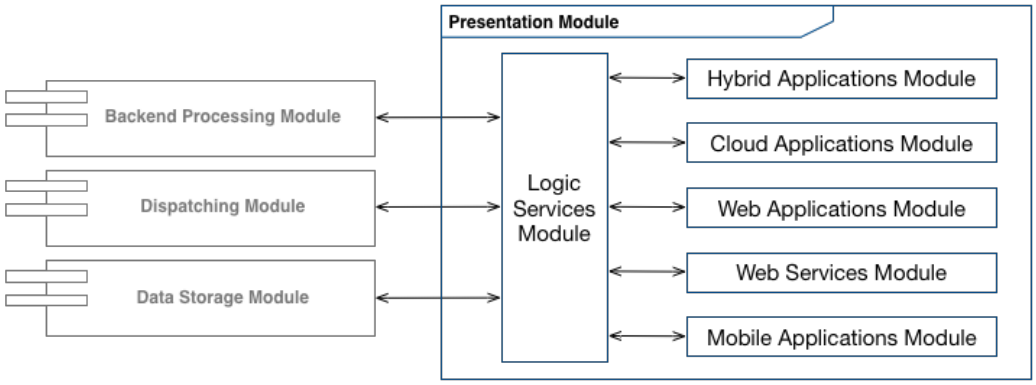

Figure 2.10: Presentation Module Architectural Model.

Basing on Figure 2.10, it's possible to infers that in this module' architectural model there are many other external components so as to provide a better description of this module and related scopes.

In fact, in the target figure there are the following external components:

- **Backend Processing Module**: module to support and manage the backend processes required by the solutions needs and scopes (reference to section 2.9).
- **Dispatching Module**: module to support the management, dispatching and routing of the data relating to the architecture internal modules (reference to section 2.6).
- **Data Storage Module**: module to manage the persistent data storage with high levels of reliability and consistency. This module will be supported by many tools, infrastructures and technologies to manage the data storage basing on the target solution requirements (reference to section 2.8).

This module external components reported in Figure 2.10 are not interested by this module analysis because their definition and description have been provided in the others sections so as to focus on their independence.

On the basis of this factor and modularization, this module will be more elastic and flexible in order to follow the guidelines related to the proposed architecture.

In the following paragraphs will be provided the definition and the description related to this module internal components so as to focus on their features and capabilities.

In details, in the *Presentation Module* module there are the following internal components:

- **Logic Services Module**: module to provide the architecture internal logical services in order to allow the *Presentation Module* modules to manage, interact and communicate to the other architecture internal modules.
- **Hybrid Applications Module**: module to provide, manage and support the architecture based solutions hybrid applications features and capabilities.
- **Cloud Applications Module**: module to provide, manage and support the architecture based solutions *Cloud* applications features and capabilities.
- **Web Applications Module**: module to provide, manage and support the architecture based solutions web applications features and capabilities.
- **Web Services Module**: module to provide, manage and support the architecture based solutions web services features and capabilities.
- **Mobile Applications Module**: module to provide, manage and support the architecture based solutions services specialized for the mobile context applications.

The defined modules allow to model, design and manage this module independently from the other architecture internal modules because the design of its internal specific entities can't be treated as base entities so as to manage this module independently from their design details.

In the following sections will be described the feature and capabilities of the *Presentation Module* internal components and modules in order to provide their complete overview.

2.10.1 Logic Services Module

The *Logic Services Module* has to provide the architecture internal logical services in order to allow the *Presentation Module* modules to manage, interact and communicate to the other architecture internal modules.

This part of the architecture has been modeled and designed as a module due to its complexity, so the internal composition can not be defined as a component because it has to provide many tools, informations, features and capabilities so as to support the *Presentation Module* modules.

Then, this module act as a broker and manager relating to the architecture internal modules in order to define an abstraction gap between the *Presentation Module* outbound modules (*Hybrid Applications Module*, *Cloud Applications Module*, *Web Applications Module*, *Web Services Module* and the *Mobile Applications Module*) and the other architecture internal modules (*Backend Processing Module*, *Dispatching Module* and *Data Storage Module*).

Therefore, it's the unique access point for the flow related to the informations, features and capabilities provided by the *Presentation Module*, so this is the logical bridge between the outbound and internal modules.

On the basis of this communication and interaction scenario, this module has to communicate and interact with other components and modules in a bidirectional way.

Therefore, this module has to manage incoming and outgoing communications so as to provide a more flexible component that make this module suitable to any kind of interaction and communication.

All the design and development details and specifications won't be defined in order to provide this module as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.10.2 Hybrid Applications Module

The *Hybrid Applications Module* has to provide, manage and support the architecture based solutions hybrid applications features and capabilities.

These applications will provide informations, data, features and capabilities to the user applications that will interact with the solutions using these kind of applications.

It has been necessary to design and model a specific module for the hybrid applications in order to allow the designers and developers to build specific services for this kind of applications.

This decision has been taken because in many scenarios require and need of custom and specific services in terms of data and informations as concern their type, format and structure.

It's often necessary to define also custom and specific services for features and capabilities as concern their management and support so as to obtain independence from other application types (eg hybrid applications often require different features and capabilities from the web applications).

Then, this outbound module has to support and manage all the hybrid applications that will be used and accessed by the user applications and the choice of this module design has been made basing on the high diffusion of this kind of applications in modern solutions.

Then, this module has to interact and communicate with the *Logic Services Module* in order to support and manage the hybrid applications features and capabilities.

On the basis of this communication and interaction scenario, this module has to communicate and interact with other components and modules in a bidirectional way.

Therefore, this module has to manage incoming and outgoing communications so as to provide a more flexible component that make this module suitable to any kind of interaction and communication.

All the design and development details and specifications won't be defined in order to provide this module as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.10.3 Cloud Applications Module

The *Cloud Applications Module* has to provide, manage and support the architecture based solutions *Cloud* applications features and capabilities.

These applications will provide informations, data, features and capabilities to the user applications that will interact with the solutions using these kind of applications.

It has been necessary to design and model a specific module for the *Cloud* applications in order to allow the designers and developers to build specific services for this kind of applications.

This decision has been taken because in many scenarios require and need of custom and specific services in terms of data and informations as concern their type, format and structure.

It's often necessary to define also custom and specific services for features and capabilities as concern their management and support so as to obtain independence from other application types (eg *Cloud* applications often require different features and capabilities from the web applications).

Then, this outbound module has to support and manage all the *Cloud* applications that will be used and accessed by the user applications and the choice of this module design has been made basing on the high diffusion of this kind of applications in modern solutions.

Thus, this module has to interact and communicate with the *Logic Services Module* in order to support and manage the *Cloud* applications features and capabilities.

On the basis of this communication and interaction scenario, this module has to communicate and interact with other components and modules in a bidirectional way.

Therefore, this module has to manage incoming and outgoing communications so as to provide a more flexible component that make this module suitable to any kind of interaction and communication.

All the design and development details and specifications won't be defined in order to provide this module as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.10.4 Web Applications Module

The *Web Applications Module* has to provide, manage and support the architecture based solutions web applications features and capabilities. These applications will provide informations, data, features and capabilities to the user applications that will interact with the solutions using these kind of applications.

It has been necessary to design and model a specific module for the web applications in order to allow the designers and developers to build specific services for this kind of applications.

This decision has been taken because in many scenarios require and need of custom and specific services in terms of data and informations as concern their type, format and structure.

It's often necessary to define also custom and specific services for features and capabilities as concern their management and support so as to obtain independence from other application types (eg web applications often require different features and capabilities from the hybrid applications).

Then, this outbound module has to support and manage all the web applications that will be used and accessed by the user applications and the choice of this module design has been made basing on the high diffusion of this kind of applications in modern solutions.

Thus, this module has to interact and communicate with the *Logic Services Module* in order to support and manage the web applications features and capabilities.

On the basis of this communication and interaction scenario, this module has to communicate and interact with other components and modules in a bidirectional way.

Therefore, this module has to manage incoming and outgoing communications so as to provide a more flexible component that make this module suitable to any kind of interaction and communication.

All the design and development details and specifications won't be defined in order to provide this module as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.10.5 Web Services Module

The *Web Services Module* has to provide, manage and support the architecture based solutions web services features and capabilities. These services will provide informations, data, features and capabilities to the external systems that will interact with the solutions using these services.

It has been necessary to design and model a specific module for the web services in order to allow the designers and developers to build specific support services for them so as to allow the development of architecture based solutions web services devoted to the interaction with external systems in-

dependently from other contexts services (eg web services for external *On-Premise* systems often require different features and capabilities from the *Cloud* applications).

Then, this outbound module has to support and manage all the web services that will be used and accessed by the external systems and the choice of this module design has been made basing on the requirements and needs of provide many kind of services to the external systems so as to support the informations and data retrieving.

Thus, this module has to interact and communicate with the *Logic Services Module* in order to support and manage the web services features and capabilities.

On the basis of this communication and interaction scenario, this module has to communicate and interact with other components and modules in a bidirectional way.

Therefore, this module has to manage incoming and outgoing communications in order to provide a more flexible component that make this module suitable to any kind of interaction and communication.

All the design and development details and specifications won't be defined so as to provide this module as independent from them allowing to design and develop they basing on the specific solutions requirements.

2.10.6 Mobile Applications Module

The *Mobile Applications Module* has to provide, manage and support the architecture based solutions services customized and specialized for the mobile context applications.

These applications will provide informations, data, features and capabilities to the user applications that will interact with the solutions using these kind of applications.

It has been necessary to design and model a specific module for the mobile applications so as to allow the designers and developers to build specific services for this kind of applications.

This decision has been taken because in many scenarios require and need of custom and specific services in terms of data and informations as concern their type, format and structure.

It's often necessary to define also custom and specific services for features and capabilities as concern their management and support in order to obtain

independence from other application types (eg mobile applications often require different features and capabilities from the web applications).

Then, this outbound module has to support and manage all the mobile applications that will be used and accessed by the user applications and the choice of this module design has been made basing on the high diffusion of this kind of applications in modern solutions.

Thus, this module has to interact and communicate with the *Logic Services Module* so as to support and manage the mobile applications features and capabilities.

On the basis of this communication and interaction scenario, this module has to communicate and interact with other components and modules in a bidirectional way.

Therefore, this module has to manage incoming and outgoing communications in order to provide a more flexible component that make this module suitable to any kind of interaction and communication.

All the design and development details and specifications won't be defined so as to provide this module as independent from them allowing to design and develop they basing on the specific solutions requirements.

Chapter 3

Cloud Platforms

In this chapter will be described, analyzed and evaluated the three major *Cloud* platforms, in order to provide an overview about them relating to the main factor of interest for *Cloud* platforms.

The *Cloud* platforms selected for this activity are the following ones:

- **Google Cloud Platform**: *Cloud* platform provided by *Google*.
- **Amazon Web Services**: *Cloud* platform provided by *Amazon*.
- **Microsoft Azure**: *Cloud* platform provided by *Microsoft*.

These platforms has been selected because they're the most interesting ones in terms of technology support, development and technology innovation due to their provider relevance and experience.

Furthermore, the Gartner [3] has evaluated the selected platforms as the best three public *Cloud* platforms focusing on their performances and most relevant aspects for customers (eg the focus has been set on key properties such as computing performances and pricing) [4][5][6].

The choice has been based on many factors related to the realization of *Cloud* solutions because the focus has to be put on the platforms that will receive the best upgrades, improvements and innovations.

A relevant aspect considered during the platforms selection process is the interest that the platforms have relating to the research and development context and enterprise context because these aspects are tightly coupled in the *Cloud* technologies.

In fact, each provider constantly works to apply innovations to its platform in order to offer more and more features, capabilities and support.

A very important factor affecting to the platform selection process is related to the tools, components and support technologies provided. In fact, all these platforms have high performance tools, components and support technologies that can help the designer and developers during all the *Cloud* solutions development and maintenance life-cycle.

Another relevant aspect in the selection of these platforms is related to their *enterprise development* interest, because these platforms are the most used in the *enterprise Cloud solutions development*.

An equivalent selection process has been done relating to the aspects with the aim of considering in the platforms introduction, description and evaluation, since it's not possible to treat all these platforms because many of the listed factors are tightly technology dependent, so they are less important in an high level analysis.

Furthermore, the *Cloud* platforms and technologies have an extremely high speed development, upgrading and improvement so it's not easy to find recent descriptions or evaluations about them. In fact, the most updated sources are the official documentations provided by the platforms vendors.

Another relevant aspect related to the platforms documentation is strictly couple with the target platforms treated in many documents and reports. In fact, most of the descriptions and evaluations are referred to open source platforms while the selected ones are proprietary [7][8][9][10].

In fact, the issues related to the limited documentations can be confirmed by the *EAI International Conference on Cloud Computing*, because it would seem to be no recent analysis and studies related to the commonly adopted platforms such as the selected ones in this analysis.

On the basis of the report *The Future of Cloud Computing Opportunities for European Cloud Computing beyond 2010* [11], the platforms analysis key factors has been selected. In fact, in that report have been defined many of the key aspects to analyze the *Cloud* platforms (eg many service categories have been selected in the platforms architecture evaluation basing on it).

More in details, the factors interested by these platforms description are the following ones:

1. **Main Properties and Characteristics**: it will be described the most important properties and characteristics of the selected platforms.

2. **Main Services Architecture**: it will be described the platforms architecture as concern the provided services, features and capabilities. The services will be grouped in order to avoid the technological and technical details that are not the target aspect and details to consider.

3. **Main Advantages**: it will be described the main advantages related to the target platform considering the main aspects strictly related to the *Cloud* technologies aspects and properties.

4. **Main Disadvantages**: it will be described the main disadvantages related to the target platform considering the main aspects strictly related to the *Cloud* technologies aspects and properties.

5. **Platform Evaluation**: it will be considered the main aspects and properties related to the platform services categories, in order to evaluate it avoiding the technological and technical details.

The platforms analysis will be provided in order to justify the usage of them to design and build *Cloud* solutions and will be based on the main properties and aspects of these technologies.

At the end, it will be provided and described a comparison between the selected platforms, in order to check and analyze if the selected platforms have different level of quality relating to the *Cloud* solutions design and build. In fact, focusing on the main aspects and properties related to the platforms services and components categories, it will be possible to compare considering all the common analyzed factors for each platform.

3.1 Amazon Web Services

Amazon Web Services is a *Cloud Platform* designed, developed and provided by *Amazon* that allows to design, build, distribute, manage and maintain applications and services through the *Internet* using the *Amazon* data centers. *Amazon* is a provider of this platform, but it also uses this platform as a customer for its products and services.

Amazon Web Services provides services as **PaaS** and **IaaS**, in order to provide an extended pool of services of different type to reply and satisfy the customers' needs and requests.

One of the most important characteristics of this platform is related to the elastic development capabilities that it provides, because it supports many different programming languages, tools and frameworks provided by many vendors (eg. it's possible to develop solutions using *.Net* and *Java* programming languages). This property allows a company to adopt this platform without any change about the used frameworks, languages and tools.

Amazon Web Services is a *Cloud Platform*, so it has all the properties related to the *Cloud* technologies previously shown and described, but it also has others specific properties that can be described using few keywords.

These keywords are the following ones:

- **Openness**: this platform allows to design, develop and distribute applications and solutions based on different programming languages, frameworks and tools provided by many vendors (eg. it's possible to develop solutions using a programming language related to the *.Net Framework* or *Java*).

- **Management by Amazon**: this platform is developed, upgraded, supported and maintained by *Amazon* and it hosts all the components, tools and infrastructure in its data centers located on different geographic areas. This factor is extremely important because it provides elevated levels of reliability, reachability and availability of the developed solutions distributed through this platform.

All the properties described for this *Cloud Platform* have an high importance, because thanks to them this platform can be used by companies to distribute their products with the assurance about the quality of the provided solutions.

3.1.1 Amazon Web Services Architecture

In this section will be described the architecture of *Amazon Web Services* in terms of kind and families of services, infrastructure and tools provided by this platform.

The architecture shown in Figure 3.1 contains all components, tools and services provided by the platform divided by context, scope and features.

Figure 3.1: Amazon Web Services Architecture.

Basing on the Figure 3.1, it can be seen that all services provided by the platform have been divided in many main categories and everyone of these has a specific context and scope, in terms of features and characteristics.

More in details, the main categories related to the platform provided services are the following ones:

- **Compute**: this category contains all services, tools and components provided to the designers and developers to support, manage and execute applications, services, machines and processes (eg. *Amazon EC2* allows to run and manage virtual machines on the *Cloud*).

- **Storage**: this category contains all services, tools and components provided to support the storage management of data on the *Cloud*, in order to allows the files storage and management on this platform (eg. *Amazon S3* provides a service of scalable storage on *Amazon Web Services Cloud Platform*).

- **Database**: this category contains all services, tools and infrastructures to contain, support, manage and query many kinds of data in terms of structure, nature, type and quantity (eg. using *Amazon RDS* it's possible to manage and query structured data in *Relational* databases).

- **Monitoring**: this category contains all services, tools and infrastructures to monitor all the resources, applications and solutions distributed through this platform (eg. using *CloudWatch* it's possible to monitor resources and applications).

- **Deployment and Management**: this category contains all services, tools and infrastructures to support the deployment and the management of *Cloud* solutions (eg. using *AWS IAM* it's possible to manage user authentication and cryptography keys and related features).

- **Application Services**: this category contains all services, tools and infrastructures to support the distributed solutions in terms of capabilities and features directly provided by the platform, in order to provide to the designers and developers a pool of common services ready to use (eg. *Amazon SNS* provides a *Push Notification* service ready to use by the distributed solutions).

- **Networking**: this category contains all services, tools and infrastructures to support the networking infrastructure and manage its

interactions and communications (eg. using the *Elastic Load Balancing* it's possible to balance the load of traffic of the interactions with the *Cloud* solutions).

The described categories provide an high level description of the *Amazon Web Services* in terms of tools, components, services and infrastructure, in order to give an holistic vision of the platform in an independent way from all the updates and upgrades made on the internal components. In fact, the scope of this section is to describe the architecture that can be used by the designers and developers to design, develop, distribute and maintain *Cloud* solutions.

The detailed focus on each component, tool and service hasn't been done because this platform evolves continually, so these parts are constantly in an rapid evolution cycle so it's not possible to keep the full description of the feature and capabilities of each component. In fact, the main target of this analysis is related to the main features and capabilities of the platform and its pool of services divided by scope and context.

3.1.2 Advantages of Amazon Web Services

In this section, the major advantages provided by *Amazon Web Services* and its usage in a *Cloud* solution will be described.

Being a *Cloud* platform, all advantages of the *Cloud* technologies are also valid for this platform so, basing on those advantages, it will be described the advantages tightly related to this specific platform, focusing on the aspects with great importance for designers and developers.

The main advantages provided by *Amazon Web Services* are the following ones:

- **Environments Integration**: *Amazon Web Services* allows to design and develop *Cloud* solutions using many programming languages (eg. *Java*), tools and frameworks without any constraint or limitation. This factor allows to integrate this platform with any technical environment of any company regarding the aspects affecting the technical and technology contexts. Basing on this feature, there are no constraints, limitations or changes required to the companies processes.

- **Data center Management and Distribution**: *Amazon Web Services* is completely supported by the *Amazon* data centers that guarantees a great support, performances, availability, reachability and scalability of the *Cloud* solutions. This factor represents a great advantage provided by this platform, because these characteristics and properties are the most important aspects related to this kind of solutions and related providers and costumers.

These advantages are the main factors that make *Amazon Web Services* as a great platform to design, develop, distribute and maintain *Cloud* solutions and it's one of most used platform by companies and software houses.

3.1.3 Disadvantages of Amazon Web Services

In this section will be described the major disadvantages provided by *Amazon Web Services* and its usage in a *Cloud* solution.

Being a *Cloud* platform, all disadvantages of the *Cloud* technologies are also valid for this platform so, basing on those disadvantages, it will be described the disadvantages tightly related to this specific platform, focusing on the aspects with great importance for designers and developers. It won't be treated the advantages related to legal context, because those issues depends on the countries laws and regulations, so they don't have much interest for this scope.

The main disadvantages provided by *Amazon Web Services* are the following ones:

- **Amazon Management**: this platform is supported, developed and maintained by *Amazon*, so all development guidelines and constraints will be provided by this provider. All constraints and guidelines have to be respected by the designers and developers of the solutions to be distributed through this platform.

- **Solutions Distribution Process Changing**: due to the external provider of this platform, the distribution process affecting the built *Cloud* solutions to distribute through *Amazon Web Services* has to fit to the distribution process provided and described by *Amazon*. If the companies who realize the solutions to distribute through this platform have a distribution process that differs from the one provided

by *Amazon*, they have to change their distribution process in order to adopt a process that fits with the distribution process provided by *Amazon*.

All the described advantages are related to the technical context, because this is the primary context related to the scope of this analysis.

The advantages related to the laws and regulations will be limited in next years, because the spread of this technologies will require better laws and regulations affecting the privacy and the other legal issues, in order to increase the diffusion of this technologies.

3.1.4 Amazon Web Services Evaluation

This section will present the evaluation related to *Amazon Web Services* introduced, analyzed and described in the previous sections.

In order to give a correct evaluation of this platform, the advantages and disadvantages previously described have been joined with many aspects related to the architecture, tools and components provided by the platform as concerns to the design, development, distribution and maintenance of the solutions realized using this platform.

Amazon Web Services has been evaluated as a great *Cloud* platform as concerns the solution production process, because it provides an extended pool of services, tools and components to support the solutions development process. This factor is extremely important because the this process is one of the most important aspects considered by companies.

More in details, the tools, components and services related to the solution design, development and maintenance (eg. *AWS CodeDeploy* to deploy automatically the developed *Cloud* applications) have been evaluated as striking solutions, because they provide a great support that allows to build *Cloud* solutions in reduced amount of time and complexity.

Analyzing the solutions distribution and management, the extended pool of data center and support tools provided by the platform allow to distribute and manage the developed solutions with an high level of quality as concerns the user experience and related major factors of interest (eg. the great performances provided by the *Amazon* data centers infrastructure).

At the end, the components and services to use to realize *Cloud* solutions. These components and services have been evaluated as interesting, because, as showed in Figure 3.1, *Amazon Web Services* provides an extended pool of components to realize *Cloud* solutions with an high level of quality.

This aspect derives from the components design, development and maintenance made by *Amazon*. Basing on this scenario the companies and software houses can focus and concentrate the resources and effort to the target scope of the solution to build, with no required time related to the components realization because they are completely managed by the provider.

3.2 Microsoft Azure

Microsoft Azure is a *Cloud Platform* designed, developed and provided by *Microsoft* that allows to design, build, distributed, manage and maintain applications and services through the *Internet* using *Microsoft* data centers.

Azure provides services as **PaaS** and **IaaS**, in order to provide an extended pool of services of different type to reply and satisfy the customers.

One of the most important characteristics of this platform affecting the elastic development capabilities that it provides, because it supports many different programming languages, tools and framework provided by many vendors (eg. it's possible to develop *.Net* and *JavaScript* solutions). This property allows a company to adopt this platform without any change about the used frameworks, languages and tools.

Azure is a *Cloud Platform* so it has all the properties related to the *Cloud* technologies previously shown and described, but it also has other specific properties that can be described using few keywords.

These keywords are the following ones:

- **Openness**: this platform allows to design, develop and distribute applications and solutions based on different programming languages, frameworks and tools provided by many vendors.
- **Management by Microsoft**: this platform is developed, upgraded, supported and maintained by *Microsoft* and it hosts all the components, tools and infrastructure in its data centers located on different geographic areas. This factor is extremely important because it provides elevated levels of reliability, reachability and availability of the developed solutions distributed through this platform.
- **Compatibility**: this platform allows to integrate all features and procedures with external systems realized using the *.Net* platform. The main scenario related to this property is affecting the *On-Premise* integration, so in this scenarios *Azure* can be integrated with external *On-Premise* solutions.

All the properties described for this *Cloud Platform* have an high importance because thanks to them this platform can be used by companies to distribute their products with the assurance about the quality of the provided solutions.

3.2.1 Microsoft Azure Architecture

In this section will be described the architecture of *Microsoft Azure* in terms of kind and families of services, infrastructure and tools provided by this platform.

The architecture shown in the Figure 3.2 contains all components, tools and services provided by the platform divided by context, scope and features.

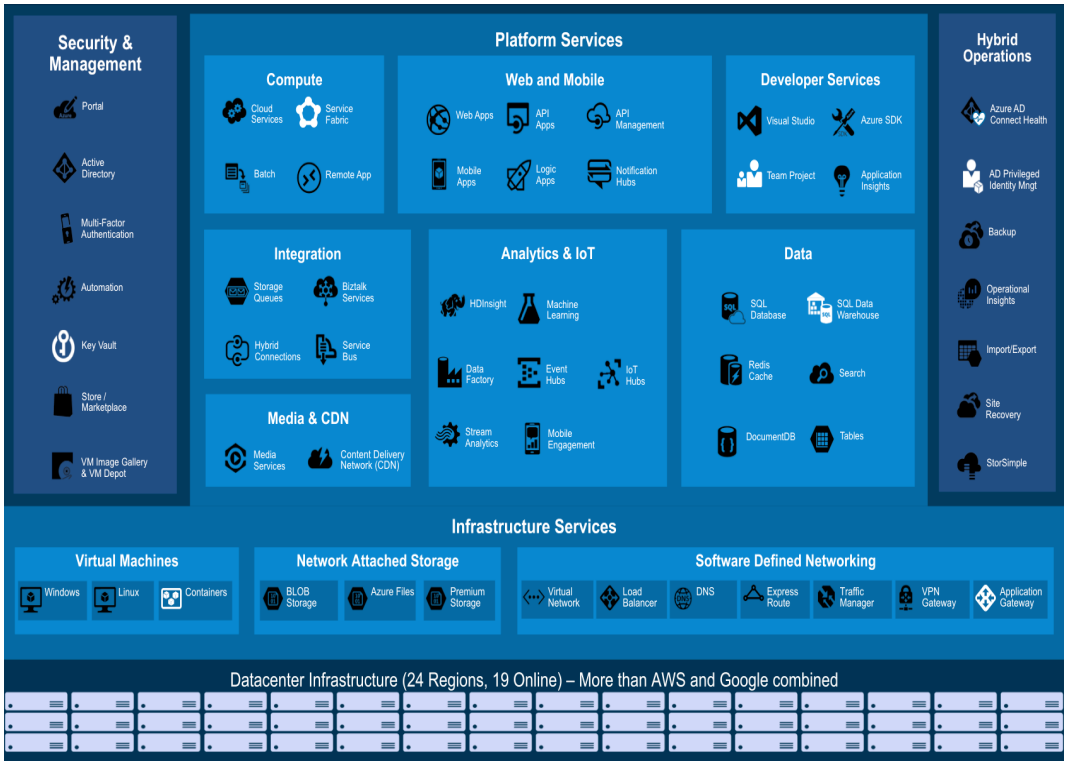

Figure 3.2: Microsoft Azure Services Architecture.

Basing on the Figure 3.2, it can be seen that all services provided by the platform have been divided in four main categories and everyone of these has a specific context and scope in terms of features and provided services.

More in details, the four main categories related to the platform provided services are the following ones:

- **Security and Management**: this category contains all services, tools and components provided to the designers and developers to build and manage the security policies and procedures affecting the solutions and the user who will use the realized solution.

- **Platform Services**: this category contains all services, tools and components provided to the designers and developers to design and develop *Cloud* solutions and applications. The components contained in this category can be used to support and provide features relating to the solutions to realize (eg. the components to manage data such as *SQL Database* and *DocumentDB*). The tools contained in this category have to be used to design and develop the solutions to realize, in order to give to designers and developers many tools to improve the development process quality (eg. the *Visual Studio* Service Tools and the *Azure SDK* to develop solutions to distribute through the *Azure Platform*).

- **Infrastructure Services**: this category contains all services, tools and components provided to the designers and developers to manage, develop and build all the services and components related to the infrastructure context closely related to the *IaaS* (*Infrastructure as a Service*), because it refers to the management of virtual machines, infrastructure to manage the data storage and the network infrastructure management.

- **Hybrid Operations**: this category contains all services, tools and components provided to the designers and developers to design, develop and distribute the products that will be distributed through the *Cloud*, in order to improve the distributed products support (eg. the backup and recovery management tools and services in order to give an easy management of these procedures).

- **Data center Infrastructure**: this category contains the references to the major data centers provided by *Microsoft* to manage and support the distribution of the *Cloud* solutions. This category it's not related to any components and services like the others previously described, but it has been reported as category, in order to give a reference to the data center because they are a focus property due their strategy related to distribution and management.

Some of the described categories are an high level grouping of many categories more detailed about services, components and tools provided by *Azure*.

Basing on this focus, the high level grouping categories are the *Platform Services* and the *Infrastructure Services* categories contains many specific categories related to many contexts and scopes. It will be described the categories contained in these two categories in order to give a more detailed description of the selected categories.

More in details, the *Platform Services* category is composed by the following ones:

- **Compute**: this category contains all services, tools and components provided to the designers and developers to support, manage and execute services, applications and processes (eg. the *Cloud Services* allows manage and build services related to the solution).
- **Web and Mobile**: this category contains all services, tools and components provided to the designers and developers to build web and mobile applications and services (eg. the *Web Apps* services allow to build web application solutions).
- **Developer Services**: this category contains all services, tools and components provided to the designers and developers to develop *Cloud* solutions to distribute and manage through the *Azure* platform (eg. *Azure SDK* supports the *Azure Cloud Solutions*).
- **Integration**: this category contains all services, tools and components provided to support the components, services and systems integration in order to provide a support to the integration process (eg. the *Service Bus* support the integration and interaction between *Cloud* components and solutions).
- **Analytics and IoT**: this category contains all services, tools and components provided to support the *Data Analysis* and *IoT* solutions (eg. *IoT Hub* allows to manage and interact with *IoT Devices* in order to support the *IoT Solutions* design and development).
- **Media and CDN**: this category contains all services, tools and components provided to support the building of *Media Services* solutions

(eg. using *Media Service*, it's possible to integrate *Media Services* in a *Cloud* solution).

- **Data**: this category contains all services, tools and components to contain, support, manage and query many kinds of data in terms of structure, nature, type and quantity (eg. using *SQL Database*, it's possible to manage and query structured data in *SQL* databases, while using *DocumentDB* it's possible to manage large quantities unstructured data).

The *Infrastructure Services* category instead is composed by the following ones:

- **Virtual Machines**: this category contains all services, tools and components provided to support and manage virtual machines and external system hosted and executed (eg. using *Containers* infrastructure, it's possible to host and execute *Docker* containers on *Azure*).

- **Network Attached Storage**: this category contains all services, tools and components provided to support the operations related to files management and storage (eg. *Azure Files* allows to support file system operations, storage and management hosting the target files on the *Cloud* platform).

- **Software Defined Networking**: this category contains all services, tools and components provided to support and manage the networking operations, communications and interactions, in order to support the interactions with the *Cloud* solutions distributed through *Azure* (eg. using the *Load Balancer*, it's possible to balance the traffic load related to the interaction with the *Cloud* solutions).

The described categories provide an high level description of *Microsoft Azure* in terms of tools, components, services and infrastructure, in order to give an holistic vision of the platform in an independent way from all the updates and upgrades made on the internal components. In fact, the scope of this section is to describe the architecture that can be used by the designers and developers to design, develop, distribute and maintain *Cloud* solutions.

The detailed focus on each component, tool and service hasn't been done because this platform evolves continually, so these parts are constantly in an rapid evolution cycle so it's not possible to keep the full description of the feature and capabilities of each component. In fact, the main target of this analysis is related to the main features and capabilities of the platform and its pool of services divided by scope and context.

3.2.2 Advantages of Microsoft Azure

In this section, the major advantages provided by *Microsoft Azure* and its usage in a *Cloud* solution will be described.

Being a *Cloud* platform, all advantages of the *Cloud* technologies are also valid for this platform so, basing on those advantages, it will be described the advantages tightly related to this specific platform, focusing on the aspects with great importance for designers and developers.

The main advantages provided by *Microsoft Azure* are the following ones:

- **Environments Integration**: *Microsoft Azure* allows to design and develop *Cloud* solutions using many programming languages (not only *.Net*, eg. *JavaScript*), tools and frameworks (eg. *.Net* and *Node.JS*) without any constraint or limitation. This factor allows to integrate this platform with any technical environment of any company regarding the aspects related to the technical and technology contexts. Basing on this feature, there are no constraints, limitations or changes required to the companies processes.

- **Solution Development Support Tools**: this platform provides a pool of components, frameworks and tools to support the design, development, maintenance and distribution of different types op *Cloud* solutions (eg. using the *IoT Hub* component make easy to manage and interact with the *Internet Of Things* and the *general purpose* devices, in order to give a good support to the interaction and the realization of infrastructures with less effort than other platforms).

- **Data center Management and Distribution**: *Microsoft Azure* is completely supported by the *Microsoft* data centers that guarantees a great support, performances, availability, reachability and scalability of the *Cloud* solutions. This factor represents a great advantage provided by this platform, because this characteristics and properties are the most important aspects related to this kind of solutions and related providers and costumers.

3.2.3 Disadvantages of Microsoft Azure

This section will present the major disadvantages provided by *Microsoft Azure* and its usage in a *Cloud* solution.

Being a *Cloud* platform, all disadvantages of the *Cloud* technologies are also valid for this platform so, basing on those disadvantages, it will be described the disadvantages tightly related to this specific platform, focusing on the aspects with great importance for designers and developers. It won't be treated the advantages related to legal context, because those issues depends on the countries laws and regulations so they don't have much interest for this scope.

The main disadvantages provided by *Azure* are the following ones:

- **Microsoft Management**: this platform is supported, developed and maintained by *Microsoft*, so all development guidelines and constraints will be provided by this provider. All constraints and guidelines have to be respected by the designers and developers of the solutions to be distributed through this platform.

- **Solutions Distribution Process Changing**: due to the external provider of this platform, the distribution process related to the built *Cloud* solutions to distribute through *Azure* has to fit to the distribution process provided and described by *Microsoft*. If the companies awho realize the solutions to distribute through this platform have a distribution process that differs from the one provided by *Microsoft*, they have to change their distribution process in order to adopt a process that fits with the distribution process provided by *Microsoft*.

All the described advantages are related to the technical context because this is the primary context related to the scope of this analysis.

The advantages related to the laws and regulations will be limited in next years, because the spread of this technologies will require better laws and regulations affecting the privacy and the other legal issues in order to increase the diffusion of this technologies.

3.2.4 Microsoft Azure Evaluation

In this section, the evaluation related to *Microsoft Azure* introduced, analyzed and described in the previous sections will be provided.

In order to give a correct evaluation of *Azure*, the advantages and disadvantages previously described have been joined with many aspects related to the architecture, tools and components provided by the platform as concerns to the solutions design, development, distribution and maintenance.

Microsoft Azure as been evaluated as interesting as concerns performances and management of the solutions design and development process, because the tools and frameworks provided to realize this type of solutions are more extended than the others and its openness is important as concerns to the companies *Technology Environment* integration of this platform.

More in details, the tools, components and services related to the solution design, development and maintenance (eg. *Azure SDK* to develop *Cloud* applications) have been evaluated as striking solutions because they provide a great support that allows to build *Cloud* products and solutions in reduced amount of time and complexity.

Analyzing the solutions distribution and management, the extended pool of data center and support tools provided by the platform allow to distribute and manage the developed solutions with an high level of quality as concerns the user experience and related major factors of interest (eg. the great performances provided by the *Microsoft* data centers infrastructure).

At the end, the components and services to use to realize *Cloud* solutions have been evaluated as impressive because, as showed in the Figure 3.2, *Microsoft* provides an extremely extended pool of services and components that can be used to realize *Cloud* solutions with an high level of quality.

This aspect derives from the components design, development and maintenance made by *Microsoft*. Basing on this scenario the companies can focus and concentrate the resources and effort to the target scope of the solution to build, with no required time related to the components realization because they are completely managed by the provider.

3.3 Google Cloud Platform

Google Cloud Platform is a *Cloud Platform* designed, developed and provided by *Google* that allows to design, build, distribute, manage and maintain applications and services through the *Internet* using the *Google* data centers. *Google* is the provider of this platform, but it also uses this platform as a customer for its products and services (eg. YouTube is provided using this platform).

Google Cloud Platform provides services as **PaaS** and **IaaS**, in order to provide an extended pool of services of different type to reply and satisfy the customers' needs and requests. This platform also includes many *Storage Platform*, in order to manage great quantities of data of different type and nature.

One of the most important characteristics of this platform is related to the elastic development capabilities that it provides, because it supports many different programming languages, tools and frameworks provided by many vendors (eg. it's possible to develop *.Net* and *Java* solutions). This property allows a company to adopt this platform without any change about the used frameworks, languages and tools.

Google Cloud Platform is a *Cloud Platform*, so it has all the properties related to the *Cloud* technologies, but it also has others specific properties that can be described using few keywords.

These keywords are the following ones:

- **Openness**: this platform allows to design, develop and distribute applications and solutions based on different programming languages, frameworks and tools provided by many vendors.
- **Management by Google**: this platform is developed, upgraded, supported and maintained by *Google* and it hosts all the components, tools and infrastructure in its data centers located on different geographic areas. This factor is extremely important because it provides elevated levels of reliability, reachability and availability of the developed solutions distributed through this platform.

All the properties described for this *Cloud Platform* have an high importance, because thanks to them this platform can be used by companies to distribute their products with the assurance about the quality of the provided solutions.

3.3.1 Google Cloud Platform Architecture

In this section will be described the architecture of *Google Cloud Platform* in terms of kind and families of services, infrastructure and tools provided by this platform.

The architecture shown in the Figure 3.3 contains all components, tools and services provided by the platform divided by context, scope and features.

Figure 3.3: Google Cloud Platform Services Architecture.

Basing on the Figure 3.3, it can be seen that all services provided by the platform have been divided in many main categories and everyone of them has a specific context and scope, in terms of features, characteristics and scope.

More in details, the main categories related to the platform provided services are the following ones:

- **Compute**: this category contains all services, tools and infrastruc-

tures provided to the designers and developers to build applications and related services, manage huge workloads on several virtual machines and support external systems hosting and execution (eg. using *Google Container Engine* it's possible to host and execute *Docker* containers in the *Google Cloud Platform* and related data centers).

- **Storage and Database**: this category contains all services, tools and infrastructures to contain, support, manage and query many kinds of data in terms of structure, nature, type and quantity (eg. using *Cloud SQL* it's possible to manage and query structured data in *MySQL* databases, while using *Google BigTable* it's possible to manage large quantities unstructured data).

- **Big Data**: this category contains all services, tools and infrastructures to support all the activities, operations and processes related to the *Data Management* and *Data Analysis* through mechanisms based on the type, nature and structure of data.

- **Machine Learning**: this category contains all services, tools and infrastructures that allow to realize solutions related to the *Machine Learning* context.

- **Identity and Security**: this category contains all services, tools and infrastructures to support all aspects related to identification of users, devices and the related security management affecting the policies and mechanisms (eg. the *Cloud Platform Security* allows to manage the platform security issues and policies).

- **Management Tools**: this category contains all services, tools and infrastructures to support the *Cloud* solutions management (eg. the logging support to track the solutions work flow and execution and the error reporting service to manage the report about solutions errors and related reporting).

- **Developer Tools**: this category contains all services, tools and infrastructures to support the *Cloud* solutions development (eg. the services and tools to integrate the *Google Cloud Platform* on several development tools like *Visual Studio*, *Android Studio* and *Eclipse*).

- **Networking**: this category contains all services, tools and infrastructures to support the networking infrastructure and manage its interactions and communications (eg. using the *Cloud Load Balancing* allows to balance the load of traffic of the interactions with the *Cloud* solutions).

The described categories provide an high level description of the *Google Cloud Platform* in terms of tools, components, services and infrastructure, in order to give an holistic vision of the platform in an independent way from all the updates and upgrades made on the internal components. In fact, the scope of this section is to describe the services architecture that can be used by the designers and developers to design, develop, distribute and maintain *Cloud* solutions.

The detailed focus on each component, tool and service hasn't been done because this platform evolves continually, so these parts are constantly in an rapid evolution cycle and it's not possible to keep the full description of the feature and capabilities of each component. In fact, the main target of this analysis is related to the main features and capabilities of the platform and its pool of services divided by scope and context.

3.3.2 Advantages of Google Cloud Platform

This section will present the major advantages provided by *Google Cloud Platform* and its usage in a *Cloud* solution.

Being a *Cloud* platform, all advantages of the *Cloud* technologies are also valid for this platform so, basing on those advantages, it will be described the advantages tightly related to this specific platform, focusing on the aspects with great importance for designers and developers.

The main advantages provided by *Google Cloud Platform* are the following ones:

- **Environments Integration**: *Google Cloud Platform* allows to design and develop *Cloud* solutions using many programming languages (eg. *Java*), tools and frameworks without any constraint or limitation. This factor allows to integrate this platform with any technical environment of any company regarding the aspects affecting the technical and technology contexts. Basing on this feature, there are no constraints, limitations or changes required to the companies processes.

- **Data center Management and Distribution**: *Google Cloud Platform* is completely supported by the *Google* data centers that guarantees a great support, performances, availability, reachability and scalability of the *Cloud* solutions. This factor represents a great advantage provided by this platform, because these characteristics and properties are the most important aspects related to this kind of solutions.

These advantages are the main factors that make *Google Cloud Platform* as a great platform to design, develop, distribute and maintain *Cloud* solutions and it's one of most used platform by companies and software houses.

3.3.3 Disadvantages of Google Cloud Platform

In this section will be described the major disadvantages provided by *Google Cloud Platform* and its usage in a *Cloud* solution.

Being a *Cloud* platform, all disadvantages of the *Cloud* technologies are also valid for this platform so, basing on those disadvantages, it will be described the disadvantages tightly related to this specific platform, focusing on the aspects with great importance for designers and developers. It won't be treated the advantages related to legal context, because those issues depends on the countries laws and regulations, so they don't have much interest for this analysis scope.

The main disadvantages provided by *Google Cloud Platform* are the following ones:

- **Google Management**: this platform is supported, developed and maintained by *Google*, so all development guidelines and constraints will be provided by this provider. All constraints and guidelines have to be respected by the designers and developers of the solutions to be distributed through this platform.

- **Solutions Distribution Process Changing**: due to the external provider of this platform, the distribution process affecting the built *Cloud* solutions to distribute through *Google Cloud Platform* has to fit to the distribution process provided and described by *Google*. If the companies who realize the solutions to distribute through this

platform have a distribution process that differs from the one provided by *Google*, they have to change their distribution process in order to adopt a process that fits with the distribution process provided by *Google*.

All the described advantages are related to the technical context, because this is the primary context related to the scope of this analysis.

The advantages related to the laws and regulations will be limited in next years, because the spread of this technologies will require better laws and regulations affecting the privacy and the other legal issues, in order to increase the diffusion of this technologies.

3.3.4 Google Cloud Platform Evaluation

In this section will be provided the evaluation related to *Google Cloud Platform* introduced, analyzed and described in the previous sections.

In order to give a correct evaluation of this platform, the advantages and disadvantages previously described have been joined with many aspects related to the architecture, tools and components provided by the platform as concerns to the design, development, distribution and maintenance of the solutions realized using this platform.

Google Cloud Platform has been evaluated as a great *Cloud* platform as concerns the solution production process, because it provides an extended pool of services, tools and components to support the solutions development process. This factor is extremely important because this process is one of the most important aspects considered by companies.

More in details, the tools, components and services affecting the solution design, development and maintenance (eg. *Google App Engine* to develop *Cloud* applications) have been evaluated as interesting solutions, because they provide a great support that allows to build *Cloud* solutions in reduced amount of time and complexity.

Analyzing the solutions distribution and management, the extended pool of data center and support tools provided by the platform allow to distribute and manage the developed solutions with an high level of quality as concerns the user experience and related major factors of interest (eg. the great performances provided by the *Google* data centers infrastructure).

Finally, the components and services to use to realize *Cloud* solutions have been evaluated as striking, because, as showed in the Figure 3.3, *Google Cloud Platform* provides an extended pool of components that can be used to realize *Cloud* solutions with an high level of quality.

This aspect derives from the components design, development and maintenance made by *Google*. Basing on this scenario the companies and software houses can focus and concentrate the resources and effort to the target scope of the solution to build, with no required time related to the components realization, because they are completely managed by the provider.

3.4 Cloud Platforms Comparison

This section will present a comparison between the selected and described *Cloud Platforms*, in order to provide an holistic vision of a comparison of these platforms.

This comparison will consider many aspects relating to the *Cloud* technologies in order to consider and analyze many points of view of these.

The first aspects that will be considered affects the platforms provided services in order to compare their fundamental aspects. After this primary comparison, it will be provided a second one, where it will be considered many other aspects not focused on the design and development of different kind of solutions related to the *Cloud*.

The considered aspects in the primary comparison are the following ones:

- **Computing**: this comparison will be focused on the platforms *Computing* services capabilities, performances and support.
- **Database**: this comparison will be focused on the platforms *Database* services capabilities and support.
- **Storage**: this comparison will be focused on the platforms *Storage* services capabilities and support.
- **Data Analysis**: this comparison will be focused on the platforms *Data Analysis* services performances, capabilities and support.
- **Identity**: this comparison will be focused on the platforms *Identity* services capabilities and support.
- **Security**: this comparison will be focused on the platforms *Security* services capabilities, performances and support.
- **Networking**: this comparison will be focused on the platforms *Networking* services capabilities, performances and support.

These aspects will be treated focusing on the provided main capabilities and features, in order to evaluate and compare the selected platforms with an high abstraction level to give a description of the common services and capabilities. This choice has been based on the most focused capabilities considered in the main *Cloud* platforms evaluation process.

The considered aspects in the solutions design and development comparison are the following ones:

- **Internet Of Things**: in this scenario, the platforms will be compared considering their support to the development of IoT solutions.
- **Big Data**: in this scenario, the platforms will be compared considering their support to the development of *Big Data* solutions.

The unique aspect that it won't be considered in these comparisons are the scope affecting the pricing of the selected platforms provided services. This aspect has not been chosen because the pricing of the *Cloud* services is an aspect not strictly associated to the quality evaluation about them. The platforms pricing plan and SLA agreement and related conditions are strictly coupled within the law and economic contexts and, basing on the targets of this analysis, these contexts are not in the main targets pool of this analysis.

In the following sections, it will be described the comparison done considering the services capabilities, features and scenarios previously defined, in order to provide an holistic comparison of these platforms.

3.4.1 Computing Services Comparison

In the *Computing Services* context, the selected platforms are at the same level as concerned to performances, customization and distribution over regions and geographic areas.

In order to provide a valid comparison about this context, it has been done considering the main success factors related to the *Cloud Computing* because this analysis has to be focused on the fundamental and most important services of this category.

The considered aspects on this context comparison are following ones:

1. **Scaling Capabilities**: this factor represents the quality of the services related to the scaling of the platforms *Computing* basing on the needs and requirements.
2. **Virtual Machines**: this factor represents the quality, performances and capabilities related to the operating systems hosting on virtual machines provided by the platforms.

3. **External Systems Hosting**: this factor represents the quality, performances and capabilities related to the external systems hosting on dedicated components (named as *Containers*) provided by the platforms.

In the following paragraphs will be provided a comparison of the platforms basing on the main factors considered as concerns the *Cloud Computing* context.

Focusing on the factor 1, the evaluation of the platforms *scaling* capabilities define that the selected platforms have the same capabilities and quality level, because all of them have an *autoscaling* support, able to scale the resources dedicated to the solutions basing on the needs and requirements.

Basing on the factor 2, all the selected platforms support the virtual machines hosting and the dedicated resources in order to provide a full support to this capabilities and services. The only differences are related to the accepted operating systems that can be hosted on these virtual machines. This difference is not so relevant because the most used operating systems are accepted from all (eg. only some *Linux* distributions are accepted from one platform and rejected from the others).

At the end, focusing on the factor 3, all the treated platforms have a particular engine categorized as *Container* that can host external systems. The unique external system that can be hosted, executed and managed on these platforms *Containers* is *Docker* so, as concerns this context, there are no differences between the platforms. The performances related to these *Containers* are very similar, because the resources are allocated to these components depending on the needs and requirements.

3.4.2 Database Management Services Comparison

In the *Database Management Services* context, the selected platforms are at the same level as concerned to the pool of *Database Management System* technologies, supported types and pool of *Data* that the platforms can manage in terms of structure, type, nature and quantity.

This context is particularly relevant because the *Data* has a key-role in the modern solutions.

In order to provide a valid comparison about this context, it has been done considering the main success factors related to the *Cloud Database*

services, because this analysis has to be focused on the fundamental and most important services of this category.

The considered aspects on this context comparison are following ones:

1. **Relational Database**: this factor represents the quality, performances and capabilities affecting the platforms services support to the *Relational Databases*.

2. **Non Relational Database**: this factor represents the quality, performances and capabilities related to the platforms services support to the *Non Relational Databases*.

3. **BLOB Storage**: this factor represents the quality, performances and capabilities affecting the platforms services support to the storage of *BLOB Data*, a collection of large binary data.

In the following paragraphs will be provided a comparison of the platforms basing on the main factors considered as concerns the *Database Management System (DBMS)* context.

Focusing on the factor 1, the evaluation of the *Relational DBMS Services* provided by the platforms is similar between all of them, because all of them can manage *Relational DBMS* with an high performance and high quality levels. The only differences affect the specific *DBMS* that the platforms can manage. In fact, the allowed *Relational DBMS* for the platforms are the following ones:

- **Amazon Web Services**: this platform can principally support and manage *Amazon Aurora*, *Microsoft SQL Server*, *Oracle DB* and *MySQL*.

- **Google Cloud Platform**: this platform can principally support and manage only *Cloud SQL*.

- **Microsoft Azure**: this platform can principally support and manage only *Microsoft SQL Server*.

Basing on the provided lists of supported *Databases*, *Amazon Web Services* can be evaluated as the best provider as concerns this context due to the most extended pool of accepted *DBMS*, because the other selected platforms can manage only one *Relational DBMS*.

Basing on the factor 2, all the treated platforms can manage and support systems related to the *Non Relational Databases Management* and the performances are very similar, because the dedicated resources can be managed and allocated, in order to provide a full support to this capabilities and services. In details, the supported technologies related to the *Non Relational Database Management* for the platforms are the following ones:

- **Amazon Web Services**: this platform can principally support and manage *Amazon DynamoDB*.
- **Google Cloud Platform**: this platform can principally support and manage only *Google BigTable and Datastore*.
- **Microsoft Azure**: this platform can principally support and manage only *Microsoft DocumentDB*.

The comparison between the treated platforms provided services affecting the *Non Relational Database Management* can be defined as similar in terms of quality, performances and capabilities, because this technologies are tightly coupled with the main standard concepts related to this context.

At the end, focusing on the factor 3, the evaluation of the *BLOB Storage Services* provided by the platforms is similar between all of them because all of them can manage *BLOB Data Storage* with an high performance and high quality levels without any restrictions about them. Due to this evaluation and the standard abstractions and management related to this context of data, the comparison between the platforms as concerns this category of services can be defined as similar in terms of quality, performances and capabilities.

3.4.3 Storage Services Comparison

The *Storage Services* context is one of the first *Cloud Services* born, because this was one of primary needs of users and customers so this aspect of the treated platform doesn't notify differences between the selected *Cloud* platforms.

All the analyzed platforms provide *Storage Services* in order to allow the customers and users to storage and locate files on on the *Cloud* platform basing on their needs and requirements. Each platform can allocate space

dynamically basing on request of the users without technological constraints and limitation.

Basing on this analysis about *Storage Services*, there are no differences between the selected platforms because this type of services is one of the first services provided by *Cloud Platforms* so it's fully provided by all the considered platforms.

3.4.4 Data Analysis Services Comparison

In the *Data Analysis Services* context, the selected platforms are at the same level as concerned to the pool of *Data Analysis* methodologies supported and types and pool of *Data* that the platforms can manage in terms of structure, type, nature and quantity. As defined about the *Database Management Services*, this context is particularly relevant because the *Data* has a key-role in the modern solutions.

In order to provide a valid comparison about this context, it has been done considering the main success factors related to the *Data Analysis* services, because this analysis has to be focused on the fundamental and most important services of this category.

The considered aspects on this context comparison are following ones:

1. **Data Batch Analysis**: it's a type of *Data Analysis* related to the analysis powered by a batch processing of incoming data, in order to analyze the data by grouping of data and each group will be analyzed to provide a batch view of the analysis results.

2. **Data Stream Analysis**: it's a type of *Data Analysis* related to the analysis powered by the data stream of incoming data, in order to analyze the data at real-time and provide real-time analysis results.

3. **Business Intelligence Analysis**: it's a type of *Data Analysis* related to the analysis powered by the data contained and managed through *DataWareHousing*.

The described type of analysis are related to the data processing type, in fact, the semantic analysis of the target data (eg. the *Predictive Data Analysis* affecting the analysis to do predictions about future events) are not directly related to the *Cloud Platforms*.

In the following paragraphs it will be provided a comparison of the platforms basing on the main factors considered as concerns the *Data Analysis*.

Basing on the factor 1, all the considered platforms provide full support and management to this *Data Processing*, in fact, all the operations affecting the *Batch Data Processing* are allowed and supported.

In details, the support to the *Batch Data Analysis* is provided by the following services and components:

- **Amazon Web Services**: this platform can principally support and manage *Batch Data Analysis* through *Amazon EMR*.
- **Google Cloud Platform**: this platform can principally support and manage *Batch Data Analysis* through *Cloud DataFlow*.
- **Microsoft Azure**: this platform can principally support and manage *Batch Data Analysis* through *HDInsight*.

The comparison between the treated platforms provided services affecting the *Batch Data Analysis* can be defined as similar in terms of quality, performances and capabilities, because this technologies are tightly coupled with the main standard concepts related to this context (eg. all the cited services and components are fully based on *Hadoop*).

Focusing on the factor 2, all the considered platforms provide full support and management to this *Data Processing*, in fact, all the operations related to the *Stream Data Processing* are allowed and supported.

In details, the support to the *Stream Data Processing* is provided by the following services and components:

- **Amazon Web Services**: this platform can principally support and manage *Stream Data Analysis* through *Amazon Kinesis*.
- **Google Cloud Platform**: this platform can principally support and manage *Stream Data Analysis* through *Cloud DataFlow*.
- **Microsoft Azure**: this platform can principally support and manage *Stream Data Analysis* through *Stream Analytics*.

The comparison between the selected platforms provided services affecting the *Stream Data Analysis* can be defined as similar in terms of quality, performances and capabilities, because this technologies are tightly coupled with the main standard concepts related to this context.

At the end, basing on the factor 3, all the considered platforms provide full support and management to this *Business Intelligence*, in fact, all the operations related to the *Business Intelligence Analysis* are allowed and supported.

In this paragraph, it will be treated the services and components affecting the *Business Intelligence Data* support and management focusing on the *Data WareHousing*, because this is the core of this type of analysis.

In details, the support to the *Data WareHousing* is provided by the following services and components:

- **Amazon Web Services**: this platform can principally support and manage *Data WareHousing* through *Amazon Redshift*.
- **Google Cloud Platform**: this platform can principally support and manage *Data WareHousing* through *BigQuery*.
- **Microsoft Azure**: this platform can principally support and manage *Data WareHousing* through *SQL Data Warehouse*.

The comparison between the selected platforms provided services affecting the *Business Intelligence Analysis and Data WareHousing* can be defined as similar in terms of quality, performances and capabilities, because this technologies are tightly coupled with the main standard concepts related to this context (eg. the main *Data WareHousing* concepts, strategies and methodologies are adopted by all the selected platforms).

In the common scenario related to a *Data Analysis* solution, anyone of these platforms can be chosen without particular problems affecting the performances, quality and capabilities provided.

This evaluation does not refers to the type of *Data Analysis*, because in this analysis the target was to evaluate and compare the components and services provided to support and manage the *Data Analysis*.

Finally, basing on the provided comparison and evaluation affecting the described factors of interest, we can define the selected platforms as similar as concerns to the properties, capabilities and performances related to the *Data Analysis* support and management.

3.4.5 Identity Services Comparison

The *Identity Services* context is one of the most important *Cloud Services*, because they allow to manage all aspects related to the user identification and management.

All the analyzed platforms provide this type of service with a particular focus on the security aspects. Due to them it's possible to increase the level of quality and capabilities affecting the security functionalities and mechanisms.

Basing on this evaluation about *Identity Services*, there are no relevant differences between the selected platforms, because all these services provide the same features and capabilities so, focusing on these specific aspects the user identification, authentication and management features are fully supported and managed.

3.4.6 Security Services Comparison

The *Security Services* context is one of the most important *Cloud Services*, because they allow to manage and support all policies, mechanisms and capabilities affecting the *Cloud Security* context.

All the analyzed platforms provide this type service because nowadays this context is one of the most important. In fact, the security context is the target one for all kind of solutions (eg. *On Premise Solutions*), specially in the *Cloud* context, because this type of platform is considered less secure than others so, in order to improve the reliability of this platforms and related solutions.

The selected platforms provides many services and components to support and manage the security of the solutions deployed on the *Cloud*. Another important aspect affecting these services is related to the continuous improvement of the mechanisms, capabilities and features to support the security. In fact, using these platforms allows to use the latest and best security features developed by the providers, with no development needed to the customer. In fact, the only task the user must complete is to configure the security constraints, policies and mechanisms to adopt.

Basing on this evaluation about *Security Services*, there are no relevant differences between the selected platforms, because all these services provide the same features and capabilities so, focusing on these specific aspects the *Cloud Security* policies and mechanisms are fully supported and managed.

3.4.7 Networking Services Comparison

The *Networking Services* context is a very important *Cloud Services* because they allow to manage and support all features, mechanisms and capabilities affecting the *Cloud Networking* context.

All the analyzed platforms provide this type service in that this context is one of the most relevant, because many solutions require network capabilities and features (eg. many solution require VPN connections in order to allow the interaction between companies offices and the *Cloud* solution).

Another important aspect related to these services affect the continuous improvement of the mechanisms, capabilities and features to support the networking features and capabilities. Using these platforms allows to use the latest and best networking features developed by the providers, with no development needed to the customer. In fact, the only task the user must complete is to configure these type of features and mechanisms to adopt.

Basing on this evaluation about *Networking Services*, there are no relevant differences between the selected platforms, because all these services provide the same features and capabilities so, focusing on these specific aspects, the *Networking* features and mechanisms are fully supported.

3.4.8 IoT Solutions Support Comparison

In this section will be provided a comparison based on the quality of the development, support and management of *Internet Of Thins* solutions, in order to provide a quality comparison between the platforms provided services affecting the *IoT* solutions context.

In this context of solutions, an important factor of success is the supplying of specific systems and services *IoT* oriented. In fact, all these providers provide specific systems and services specific for the *Internet Of Things* solutions, in order to improve and simplify the development.

In details, the systems provided by these providers are the following ones:

1. **Microsoft** provides *Windows 10 IoT Core*[22]. It's a version of Windows 10 that is optimized for smaller devices with or without a display (eg. Raspberry Pi) that utilizes the rich, extensible *Universal Windows Platform* (*UWP*) API for building *Internet Of Things* solutions.

2. **Amazon** provides *AWS IoT*[23]. It's a system of services and components that provide secure, bi-directional communication between Internet-connected things (such as sensors, actuators, embedded devices, or smart appliances) and the AWS cloud, in order to support the building process of *Internet Of Things* solutions.

3. **Google** provides *Android Things*[24]. It's a version of Android that is optimized for smaller devices with or without a display (eg. Raspberry Pi) that utilizes the Android Platform API for building *Internet Of Things* solutions.

The major difference between *Windows 10 IoT Core*, *Android Things* and *AWS IoT* is that the first and the second one are operating system, while the last one is a pool of services and components to support the interaction between the *IoT* devices and the *Cloud Platforms*.

An important factor affecting the *openness* of the integration and interaction between the *IoT* devices and *Cloud Platforms*, because none of the selected platforms set constraints about the interaction. In fact, all the treated platforms provides services, infrastructures and components to support the building of *Internet Of Things* solutions in a independently from the operating system of the *IoT* devices.

The advantages given by the use of operating system and *Cloud Platform* belonging to the same provider affect the better and easier integration between the *Cloud Platform* and the devices when they belong to the same technical ecosystem.

These advantages are not fundamental aspects, because in the heterogeneous scenario (devices operating system and *Cloud Platform* provided by different vendors) the required effort to build an *Internet Of Things* solution is not too higher compared to the homogeneous scenario (devices operating system and *Cloud Platform* provided by the same vendor).

Basing on this analysis, we can define the treated platforms as similar as concerns to the *Internet Of Things* solution building process, because all the major aspects and factors affecting this process are completely supported and managed as concerns.

3.4.9 Big Data Solutions Support Comparison

In this section will be provided a comparison based on the quality of the development, support and management of *Big Data* solutions, in order to provide a quality comparison between the platforms provided services affecting the *Big Data* solutions context.

In order to analyze the comparison related to this context, the considered aspects in this analysis are related to the data processing type. In fact, the semantic analysis of the target data (eg. the *Predictive Data Analysis* affecting the analysis to do predictions about future events) are not directly related to the *Cloud Platforms*.

Basing on the selected platforms provided services described and compared in the section 3.4.4 (section dedicated to the *Data Analysis Services Comparison*), there are no relevant differences between these platforms as concerns the building process of *Big Data* solutions.

All these platforms services and components affecting the *Big Data* context provide complete support and capabilities related to the data analysis through this type of solutions with an high level of quality of service in terms of reliability, performances and results quality.

Typically, the *Big Data* and the *Internet Of Things* solutions are strictly coupled, because nowadays the concept of data is a pivot of the companies core business while the most diffused context to produce and collect data to analyze is the *Internet Of Things* (eg. in the industrial context the *IoT* solutions allow to collect all interested data and *Big Data* solutions allow to analyze the collected data about industrial devices and environment properties and characteristics).

The collected data by the *Internet Of Things* solutions can be managed and analyzed by *Big Data* solutions, in order to support and provide results to improve the company core business.

Basing on this analysis, we can define the treated platforms as similar as concerns to the *Big Data* solution building process, because all the major aspects and factors affecting this process are completely supported and managed as concerns.

Chapter 4

Technological Evaluation

In this chapter a technological evaluation of the proposed architecture will be provided. In details, this evaluation has been focused on the compatibility and congruency between the analyzed and considered *Cloud* platforms and the proposed architecture.

Basing on the descriptions, analysis and evaluations provided in the *Cloud Platforms* chapter (reference to Chapter 3), it's possible to describe how this architecture can be used to design and build *Cloud* solutions using those platforms.

This evaluation will take into account many aspects related to the proposed architecture focusing on the capabilities and features that can be completely provided by the analyzed *Cloud* platforms because the main scope is to demonstrate and verify that those platforms can support the solutions based on the proposed architecture.

In details, all the architecture modules will be treated and considered in order to describe and analyze how they can be supported and build using the chosen *Cloud* platforms. In each section, it will be verified if a module and its internal components and modules can be developed using the *Cloud* platforms services considering their main features and capabilities.

Furthermore, this evaluation will also provide a main overview relating to the guidelines, so as to design and build solutions based on the proposed architecture by using the chosen *Cloud* platforms.

In the following sections, all the features and capabilities required by the proposed architecture that can be completely provided and supported by the chosen *Cloud* platforms will be analyzed and described.

4.1 Data Hub Module

This section will provide a technological analysis related to design and development of the *Data Hub Module* in order to provide an high level overview relating to the compatibility of the analyzed *Cloud* platforms and the design and implementation of this module.

This module can be designed and developed by using any analyzed *Cloud* platforms; in details, the components of the platforms that enable and support this module are the following:

1. **Amazon Web Services**: this platform can support and enable this module basing on the *Cloud* services and components related to the *AWS IoT* (eg through the components *Device Gateway* and *Message broker* enable the data communication)[23] as concerns the *Cloud* side.

2. **Microsoft Azure**: this platform can support and enable this module basing on the *Cloud* services and components related to the *IoT Hub*[25] as concerns the *Cloud* side and *IoT Core*[22] as concerns the remote device side.

3. **Google Cloud Platform**: this platform can support and enable this module basing on the *Cloud* services and components related to the *Cloud Pub/Sub*[26] as concerns the *Cloud* side and *Android Things*[24] as concerns the remote device side.

Focusing on the remote device side of the computation and interaction related to this module, all the platforms can interact with any kind and type of device and the unique constraint is related to the device capabilities; In fact, a device can interact with these platforms and related services and components only if it implements and uses the required protocols (generally the required protocol is *MQTT*[27]).

While *Azure* and *Google Cloud Platform* have a remote device side corresponding system, *Amazon Web Services* doesn't have it but this is not a limit because in the *Internet of Things* context, this factor doesn't limit any kind of interactions or communication.

Basing on this analysis, the *Data Hub Module* can be designed, developed and supported using anyone of the analyzed *Cloud* platforms, so this part of the proposed architecture can be realized without any constraint or limitation related to the scope of this module.

All the *Data Hub Module* internal components can be designed and developed using the introduced components and services as concerns the *Cloud* side in order to build the treated module in this section.

The strategy to follow to design and develop this module internal components is strictly coupled with the context specialization of the *Cloud* platforms components and services. In fact, following this strategy it's possible to use more instances of the *Cloud* platforms components and services in order to separate and specialize them basing on the *IoT Devices* and the *General Purpose Device* contexts in order to specialize their feature to support the *IoT Data Hub* and the *Device Data Hub*.

At the end, the *Data Hub Module* external components can be designed, developed and supported defining specific components to locate on the remote devices basing on the devices systems and technologies. In fact, basing on this specific software components development strategy it's possible to build these external components in order to interact with the *Cloud* side components contained in the *Data Hub Module*.

The defined strategies and methodologies to design, develop and support the *Data Hub Module* and its external and internal components allow to define this module of the proposed architecture as fully achievable using anyone of the selected *Cloud* platforms.

4.2 Events Hub Module

This section will provide a technological analysis related to design and development of the *Events Hub Module* in order to provide an high level overview relating to the compatibility of the analyzed *Cloud* platforms and the design and implementation of this module.

This module can be designed and developed by using any analyzed *Cloud* platforms; in details, the components of the platforms that enable and support this module are the following:

- **Amazon Web Services**: this platform can support and enable this module basing on the *Cloud* services and components related to the *AWS IoT* (eg through the component *Message broker* enable the events data communication)[23] as concerns the *Cloud* side.
- **Microsoft Azure**: this platform can support and enable this module basing on the *Cloud* services and components related to the *IoT*

Hub[25] and *Event Hub*[28] as concerns the *Cloud* side and *IoT Core*[22] as concerns the remote device side.

- **Google Cloud Platform**: this platform can support and enable this module basing on the *Cloud* services and components related to the *Cloud Pub/Sub*[26] as concerns the *Cloud* side and *Android Things*[24] as concerns the remote device side.

Focusing on the remote device side of the computation and interaction related to this module, all the platforms can interact with any kind and type of device and the unique constraint is related to the device capabilities; In fact, a device can interact with these platforms and related services and components only if it implements and uses the required protocols (generally the required protocol is *MQTT*[27]).

While *Azure* and *Google Cloud Platform* have a remote device side corresponding system, *Amazon Web Services* doesn't have it but this is not a limit because in the *Internet of Things* context, this factor doesn't limit any kind of interactions or communication.

Basing on this analysis, the *Events Hub Module* can be designed, developed and supported using anyone of the analyzed *Cloud* platforms, so this part of the proposed architecture can be realized without any constraint or limitation related to the scope of this module.

All the *Events Hub Module* internal components can be designed and developed using the introduced components and services as concerns the *Cloud* side in order to build the treated module in this section.

The strategy to follow to design and develop this module internal components is strictly coupled with the context specialization of the *Cloud* platforms components and services. In fact, following this strategy it's possible to use more instances of the *Cloud* platforms components and services in order to separate and specialize them basing on the *IoT Devices* and the *General Purpose Device* contexts in order to specialize their feature to support the *IoT Event Hub* and the *Device Event Hub*.

At the end, as concerns the *Events Hub Module* external components, they can be designed, developed and supported defining specific components to locate on the remote devices basing on the devices systems and technologies. In fact, basing on this specific software components development strategy it's possible to build these external components in order

to interact with the *Cloud* side components contained in the *Events Hub Module*.

The defined strategies and methodologies to design, develop and support the *Events Hub Module* and its external and internal components allow to define this module of the proposed architecture as fully achievable using anyone of the selected *Cloud* platforms.

4.3 Messages Hub Module

This section will analyze technological aspects related to design and development of the *Messages Hub Module* in order to provide an high level overview relating to the compatibility of the analyzed *Cloud* platforms and the design and implementation of this module.

This module can be designed and developed by using any analyzed *Cloud* platforms; in details, the components of the platforms that enable and support this module are the following:

- **Amazon Web Services**: this platform can support and enable this module basing on the *Cloud* services and components related to the *Amazon Simple Queue Service*[29] and the *Message broker*, that enable the message communication as concerns the *Cloud* side.
- **Microsoft Azure**: this platform can support and enable this module basing on the *Cloud* services and components related to the *IoT Hub*[25] and *Service Bus Messaging*[30] as concerns the *Cloud* side and *IoT Core*[22] as concerns the remote device side.
- **Google Cloud Platform**: this platform can support and enable this module basing on the *Cloud* services and components related to the *Google Cloud Messaging*[31] as concerns the *Cloud* side and *Android Things*[24] as concerns the remote device side.

Focusing on the remote device side of the computation and interaction related to this module, all the platforms can interact with any kind and type of device and the unique constraint is related to the device capabilities; In fact, a device can interact with these platforms and related services and components only if it implements and uses the required protocols (generally the required protocol is *MQTT*[27]).

While *Azure* and *Google Cloud Platform* have a remote device side corresponding system, *Amazon Web Services* doesn't have it but this is not a limit because in the *Internet of Things* context, this factor doesn't limit any kind of interactions or communication.

Basing on this analysis, the *Messages Hub Module* can be designed, developed and supported using anyone of the analyzed *Cloud* platforms, so this part of the proposed architecture can be realized without any constraint or limitation related to the scope of this module.

All the *Messages Hub Module* internal components can be designed and developed using the introduced components and services as concerns the *Cloud* side in order to build the treated module in this section.

The strategy to follow to design and develop this module internal components is strictly coupled with the context specialization of the *Cloud* platforms components and services. In fact, following this strategy it's possible to use more instances of the *Cloud* platforms components and services in order to separate and specialize them basing on the *IoT Devices* and the *General Purpose Device* contexts in order to specialize their feature to support the *IoT Message Hub* and the *Device Message Hub*.

At the end, as concerns the *Messages Hub Module* external components, they can be designed, developed and supported defining specific components to locate on the remote devices basing on the devices systems and technologies. In fact, basing on this specific software components development strategy it's possible to build these external components in order to interact with the *Cloud* side components contained in the *Messages Hub Module*.

The defined strategies and methodologies to design, develop and support the *Messages Hub Module* and its external and internal components allow to define this module of the proposed architecture as fully achievable using anyone of the selected *Cloud* platforms.

4.4 Dispatching Module

This section will provide a technological analysis related to design and development of the *Dispatching Module* in order to provide an high level overview relating to the compatibility of the analyzed *Cloud* platforms and the design and implementation of this module.

This module can be designed and developed by using any analyzed *Cloud* platforms; in details, the components of the platforms that enable and support this module are the following:

- **Amazon Web Services**: this platform can support and enable this module basing on the *Cloud* services and components related to *Amazon Simple Queue Service*[29] and the *Message broker*, in order to support the internal components relating to the informations flow in *Cloud* solutions.

- **Microsoft Azure**: this platform can support and enable this module basing on the *Cloud* services and components related to the *Service Bus*[32], in order to support the internal components relating to the informations flow in *Cloud* solutions.

- **Google Cloud Platform**: this platform can support and enable this module basing on the *Cloud* services and components related to the *Cloud Pub/Sub*[26], in order to support the internal components relating to the informations flow in *Cloud* solutions.

The introduced components and services of the selected *Cloud* platforms can support all the *Dispatching Module* internal components. In fact, the internal components can be developed and built with less complexity because the informations flow at low level is completely managed by the platforms services. Therefore, the dispatcher components and the related hub have to be designed and built in order to allow them to interact and manage these *Cloud* platforms services to manage the informations flow at an high level of abstraction.

Basing on this analysis, the *Dispatching Module* can be designed, developed and supported using anyone of the analyzed *Cloud* platforms, so this part of the proposed architecture can be realized without any constraint or limitation related to the scope of this module.

All the *Dispatching Module* internal components can be designed and developed using the introduced components and services in order to build the treated module in this section.

The strategy to follow to design and develop this module internal components is strictly coupled with the context specialization of the *Cloud* platforms components and services. In fact, following this strategy it's possible

to use more instances of the *Cloud* platforms components and services in order to separate and specialize them basing on the contexts of the informations to manage and dispatch in order to specialize their feature to support the *Data Dispatcher*, *Events Dispatcher*, *Messages Dispatcher* and the *Dispatching Hub*.

The defined strategies and methodologies to design, develop and support the *Dispatching Module* and its internal components allow to define this module of the proposed architecture as fully achievable using anyone of the selected *Cloud* platforms.

4.5 Data Analysis Module

This section will be based on a technological analysis related to design and development of the *Data Analysis Module* in order to provide an high level overview relating to the compatibility of the analyzed *Cloud* platforms and the design and implementation of this module.

This module can be designed and developed by using any analyzed *Cloud* platforms; in details, the components of the platforms that enable and support this module are the following:

- **Amazon Web Services**: this platform can support and enable this module basing on the data analysis and data analytics categories of *Cloud* services and components; In details, *Amazon QuickSight*[33] as concerns the *Business Intelligence* module, *Amazon EMR*[34] as concerns the *Data Batch Processing Module* and *Amazon Kinesis*[35] as concerns the *Data Stream Processing Module*.

- **Microsoft Azure**: this platform can support and enable this module basing on the data analysis and data analytics categories of *Cloud* services and components; In details, *Azure Analysis Services*[36] as concerns the *Business Intelligence Module*, *Azure HDInsight*[37] as concerns the *Data Batch Processing Module* and *Azure Stream Analytics*[38] as concerns the *Data Stream Processing Module*.

- **Google Cloud Platform**: this platform can support and enable this module basing on the data analysis and data analytics categories of *Cloud* services and components; In details, *Google BigQuery*[39]

as concerns the *Business Intelligence* module, *Cloud Dataproc*[40] as concerns the *Data Batch Processing Module* and *Cloud Dataflow*[41] as concerns the *Data Stream Processing Module*.

The introduced components and services of the selected *Cloud* platforms can support all the *Data Analysis Module* internal components. In fact, the internal modules and components can be developed and built with less complexity because the low level complexity of the data analysis is completely managed and supported by the platforms services. Therefore, the internal modules and components have to be designed and built in order to allow them to interact and manage these *Cloud* platforms services to manage the data analysis at an high level of abstraction.

The previously described *Cloud* services are the main platforms services to use in this data analysis context and, in order to build the proposed architecture *Data Analysis Module*, it will be necessary to use other platforms services that will act as supports and helpers for the main services previously introduced (eg in *Cloud* solutions built for *Google Cloud Platform* relating to the *Data Batch Processing*, it's necessary to use the informations flow services to communicate the target data to *Cloud Dataproc*).

Basing on this analysis, the *Data Analysis Module* can be designed, developed and supported using anyone of the analyzed *Cloud* platforms, so this part of the proposed architecture can be realized without any constraint or limitation related to the scope of this module.

All the *Data Analysis Module* internal components and modules can be designed and developed using the introduced components and services in order to build the treated module in this section.

The strategy to follow to design and develop this module internal components is strictly coupled with the context specialization of the *Cloud* platforms components and services. In fact, following this strategy it's possible to use many instances of the *Cloud* platforms components and services in order to separate and specialize them basing on the contexts of the informations to manage and related analysis to execute.

The defined strategies and methodologies to design, develop and support the *Data Analysis Module* and its internal components and modules allow to define this module of the proposed architecture as fully achievable using anyone of the selected *Cloud* platforms.

4.6 Data Storage Module

This section will analyze technological issues related to design and development of the *Data Storage Module* in order to provide an high level overview relating to the compatibility of the analyzed *Cloud* platforms and the design and implementation of this module.

In particular, in this section will the focus will be set on the *Database Management Systems* of the analyzed *Cloud* platforms because the proposed architecture *Data Storage Module* is strictly coupled and its scope relating to the *Databases*. In fact, first of all will be described how the platforms *Database Management System Services* can be used to design and build the *Database and BLOB Storage* of the *Data Storage Module*. After that, the focus will be set on the *Data Storage Accessing Hub* in order to provide a description of how the platforms services can be used to design and build this component.

This module can be designed and developed by using any analyzed *Cloud* platforms; in details, the components of the platforms that enable and support this module are the following:

- **Amazon Web Services**: this platform can support and enable this module basing on the data storage categories of *Cloud* services and components; In details, *Amazon RDS*[42] as concerns the *Relational Database Storage Module*, *Amazon DynamoDB*[43] as concerns the *Not Relational Database Storage Module* and *Amazon S3*[44] as concerns the *BLOB Storage Module*.

- **Microsoft Azure**: this platform can support and enable this module basing on the data storage categories of *Cloud* services and components; In details, *Azure SQL Database*[45] as concerns the *Relational Database Storage Module*, *DocumentDB*[46] as concerns the *Not Relational Database Storage Module* and *BLOB Storage*[47] as concerns the *BLOB Storage Module*.

- **Google Cloud Platform**: this platform can support and enable this module basing on the data storage categories of *Cloud* services and components; In details, *Cloud SQL*[48] as concerns the *Relational Database Storage Module*, *Cloud Bigtable*[49] as concerns the *Not Relational Database Storage Module* and *Cloud Storage*[50] as concerns the *BLOB Storage Module*.

The introduced components and services of the selected *Cloud* platforms can support all the *Data Storage Module* internal modules relating to the *Database Management* (*Relational Database Module*, *Not Relational Database Module* and *BLOB Storage Module*). In fact, the internal modules can be developed and built with less complexity because the low level complexity of the data storage is completely managed and supported by the platforms services. Therefore, the internal modules have to be designed and built in order to allow them to interact and manage these *Cloud* platforms services to manage the data storage at an high level of abstraction.

In the previous paragraphs, the *Data Storage Module* internal modules related to the *Database Storage* and *BLOB Storage* have been introduced in order to describe how they can be designed and built using the selected *Cloud* platforms components and services.

In the following paragraphs will be described how the *Data Storage Accessing Hub* can be designed and built using the selected *Cloud* platforms components and services.

The *Data Storage Accessing Hub* can be designed and developed by using any analyzed *Cloud* platforms; in details, the platforms services that enable and support this component are the following:

- **Amazon Web Services**: this platform can support and enable this component basing on the *Cloud* services and components related to *Amazon Simple Queue Service*[29] and the *Message broker*, in order to support the informations data flow in *Cloud* solutions.

- **Microsoft Azure**: this platform can support and enable this component basing on the *Cloud* services and components related to the *Service Bus*[32], in order to support the informations data flow in *Cloud* solutions.

- **Google Cloud Platform**: this platform can support and enable this component basing on the *Cloud* services and components related to the *Cloud Pub/Sub*[26], in order to support the informations data flow in *Cloud* solutions.

The introduced components and services of the selected *Cloud* platforms can support the *Data Storage Accessing Hub*. In fact, it can be developed and built with less complexity because the low level complexity of the informations data flow is completely managed and supported by the platforms

services. Therefore, the *Data Storage Accessing Hub* has to be designed and built in order to allow them to interact and manage these *Cloud* platforms services to manage the data flow at an high level of abstraction.

Basing on this analysis, the *Data Storage Module* can be designed, developed and supported using anyone of the analyzed *Cloud* platforms, so this part of the proposed architecture can be realized without any constraint or limitation related to the scope of this module.

All the *Data Storage Module* internal components and modules can be designed and developed using the introduced components and services in order to build the treated module in this section.

The defined strategies and methodologies to design, develop and support the *Data Storage Module* and its internal components and modules allow to define this module of the proposed architecture as fully achievable using anyone of the selected *Cloud* platforms.

4.7 Backend Processing Module

This section will provide a technological analysis related to design and development of the *Backend Processing Module* in order to provide an high level overview relating to the compatibility of the analyzed *Cloud* platforms and the design and implementation of this module, particularly focusing on the platforms batch process execution *Cloud* services because this is the main scope of this module.

This module can be designed and developed by using any analyzed *Cloud* platforms; in details, the components of the platforms that enable and support this module are the following:

- **Amazon Web Services**: this platform can support and enable this module basing on the *Cloud* services and components related to the *AWS Batch*[51], in order to support the execution of batch processes in *Cloud* solutions.

- **Microsoft Azure**: this platform can support and enable this module basing on the *Cloud* services and components related to the *Azure Batch*[52], in order to support the execution of batch processes in *Cloud* solutions.

- **Google Cloud Platform**: this platform can support and enable this module basing on the *Cloud* services and components related to the *Google Compute Engine*[53], in order to support the execution of batch processes in *Cloud* solutions.

The introduced components and services of the selected *Cloud* platforms can support all the *Backend Processing Module* internal components. In fact, the internal components can be developed and built with less complexity because the low level complexity of the batch processes execution is completely managed and supported by the platforms services. Therefore, the internal components have to be designed and built in order to allow them to interact and manage these *Cloud* platforms services to manage the batch processes execution at an high level of abstraction.

Basing on this analysis, the *Backend Processing Module* can be designed, developed and supported using anyone of the analyzed *Cloud* platforms, so this part of the proposed architecture can be realized without any constraint or limitation related to the scope of this module.

The *Backend Processing Module* internal components and modules can be designed and developed using the introduced components and services in order to build the treated module in this section.

The defined strategies and methodologies to design, develop and support the *Backend Processing Module* and its internal components allow to define this module of the proposed architecture as fully achievable using anyone of the selected *Cloud* platforms.

4.8 Presentation Module

This section will provide a technological analysis related to design and development of the *Presentation Module* in order to provide an high level overview relating to the compatibility of the analyzed *Cloud* platforms and the design and implementation of this module.

In the following paragraphs will be described and treated many contexts relating to the many *Cloud* platforms services related to the internal modules and components of the *Presentation Module*, in order to specify the target platforms services for each module and component contained in this module because it has an extended composition so it's complex to describe and detect all the platforms services to use without this partition.

The first module to analyze is the *Logic Services Module* because it is the core component of the *Presentation Module* as concerns the business logic and work flow of this module.

This module can be designed and developed by using any analyzed *Cloud* platforms; in details, the components of the platforms that enable and support this module are the following:

- **Amazon Web Services**: this platform can support and enable this component basing on the *Cloud* services and components related to *Amazon Simple Queue Service*[29] and the *Message broker*, in order to support the features and informations data flow in order to provide all the capabilities and features related to this module.

- **Microsoft Azure**: this platform can support and enable this component basing on the *Cloud* services and components related to the *Service Bus*[32], in order to support the features and informations data flow in order to provide all the capabilities and features related to this module.

- **Google Cloud Platform**: this platform can support and enable this component basing on the *Cloud* services and components related to the *Cloud Pub/Sub*[26], in order to support the features and informations data flow in order to provide all the capabilities and features related to this module.

After the analysis and description related to the design and development of the *Logic Services Module*, it's necessary to analyze the other modules contained in the proposed architecture *Presentation Module*.

In this phase, a good strategy to provide guidelines and key-practices relating to the design and development of the *Application-Oriented Modules* (*Web Applications Module*, *Hybrid Applications Module* and *Cloud Applications Module*) and the *Service-Oriented Modules* (*Web Services Module* and *Mobile Applications Module*) is based on subdivision of these modules in these defined groups in order to treat separately the related modules main aspects, properties and capabilities.

This grouping has been done focusing on the targets, scopes and features of the interested modules, in fact, the *Application-Oriented Modules* are strictly related to the applications support and management (eg the *Web*

Applications Module has to support and manage the proposed architecture solutions Web Applications); the *Service-Oriented Modules*, instead, are strictly related to the services support and management (eg the *Mobile Applications Module* has to support and manage the services dedicated to the proposed architecture solutions Mobile Applications).

The *Application-Oriented Modules* can be designed and developed by using any analyzed *Cloud* platforms; in details, the components of the platforms that enable and support this module are the following:

- **Amazon Web Services**: this platform can support and enable these modules basing on the *Cloud* services and components related to the *Amazon EC2 Container Service*[54], *Amazon AppStream 2.0*[55] and *AWS Step Functions*[56], in order to support the management and execution of applications in *Cloud* solutions.

- **Microsoft Azure**: this platform can support and enable these modules basing on the *Cloud* services and components related to the *Azure App Service*[57], *Azure Container Service*[58] and *Azure Fabri Service*[59], in order to support the management and execution of applications in *Cloud* solutions.

- **Google Cloud Platform**: this platform can support and enable these modules basing on the *Cloud* services and components related to the *Google Container Engine*[60] and *Google App Engine*[61], in order to support the management and execution of applications in *Cloud* solutions.

All the *Cloud* platforms services and components required to design and build the *Application-Oriented Modules* can manage all the aspects and properties related to the low level complexity of these modules context. Therefore, all these internal modules have to be designed and built in order to allow them to interact and manage these *Cloud* platforms services to support the management and execution of applications at an high level of abstraction.

After the analysis related to the *Application-Oriented Modules*, the *Service-Oriented Modules* will be treated in order to analyze and describe how they can be designed and built using the *Cloud* platforms services.

The *Service-Oriented Modules* can be designed and developed by using any analyzed *Cloud* platforms; in details, the components of the platforms that enable and support this module are the following:

- **Amazon Web Services**: this platform can support and enable these modules basing on the *Cloud* services and components related to the *Amazon API Gateway*[62], *AWS Mobile*[63] and *Amazon Cognito*[64], in order to support the management and execution of services and API in *Cloud* solutions.
- **Microsoft Azure**: this platform can support and enable these modules basing on the *Cloud* services and components related to the *Azure Cloud Services*[65] and *Azure App Service*[57], in order to support the management and execution of services and API in *Cloud* solutions.
- **Google Cloud Platform**: this platform can support and enable these modules basing on the *Cloud* services and components related to the *Cloud Endpoints*[66], in order to support the management and execution of services and API in *Cloud* solutions.

All the *Cloud* platforms services and components required to design and build the *Service-Oriented Modules* can manage all the aspects and properties related to the low level complexity of these modules context. Therefore, all these internal modules have to be designed and built in order to allow them to interact and manage these *Cloud* platforms services to support the management and execution of services and API at an high level of abstraction.

Basing on the analysis related to *Application-Oriented Modules* and *Service-Oriented Modules* internal modules, the *Presentation Module* can be designed, developed and supported using anyone of the analyzed *Cloud* platforms, so this part of the proposed architecture can be realized without any constraint or limitation related to the scope of this module.

All the *Presentation Module* internal components and modules can be designed and developed using the introduced components and services in order to build the treated module in this section.

The defined strategies and methodologies to design, develop and support the *Presentation Module* and its internal components and modules allow to define this module of the proposed architecture as fully achievable using anyone of the selected *Cloud* platforms.

Chapter 5

Case Study

In this chapter will be described a particular case study related to the proposed architecture, with the aim of providing an overview and a description of a scenario where the proposed architecture is exploited in the *Cloud* solutions building process.

This case study is particularly important in order to describe how the solution can be structured, designed and modeled to get all the advantages provided by the target architecture.

First of all, the main overview of the target *Cloud* solution context will be described, so as to provide an introduction based on the target context where the architecture will be adopted. Basing on this overview, it's possible to cover the most extended context affecting a *Cloud* solution in terms of scopes, features and capabilities.

After the overview about the *Cloud* solution context, it will be described the strategy affecting the model and design of the target solution to build so as to provide a detailed description and definition about the adoption of the proposed architecture in the selected context. This phase will describe the high level modeling and the design of the target solution in order to analyze all the affecting aspects and concepts.

At the end, the model and design processes of the target solution and the strategy adopted to build it basing on the proposed architecture will be evaluated. This will let us to provide an evaluation about the feasibility of a complete *Cloud* solution adopting the target architecture.

5.1 Solution Overview

The target solution to model and design has been based on the extended scenario affecting the *Cloud* solutions context, in order to provide the model and design of a case study. Thus, it's possible to describe the most extended scenario where the architecture can be adopted.

The solution that will be modeled and designed has to be related to the following contexts:

- **Internet Of Things**: this solution has to be an *IoT* solution, so it has to manage, interact and communicate to *IoT* devices, in order to collect interesting data and informations.
- **Big Data**: this solution has to be a *Big Data* solution, so it has to manage, analyze, extract and infer results, in order to provide interesting informations derived from the collected data.

The details about the contexts related to the target solution that will be treated in this case study have not been defined and described, because the main scope of this study is to model and design a particularly extended solution, adopting an high level of abstraction.

Focusing on this scope, it's not necessary to manage and treat the low level details about the target solution, such as technological details and specifications, thus only the main high level aspects will be treated.

Furthermore, the target solution has to provide capabilities and features to the end users through the following channels:

- **Cloud Applications**: this solution has to provide a pool of *Cloud* applications, in order to allow the end-users to access, retrieve and manage the all the data, features and capabilities using them.
- **Cloud Services**: this solution has to support a pool of *Cloud* services, in order to allow external systems to access, retrieve and manage the all the data, features and capabilities using them.
- **Cloud Mobile Applications**: this solution has to support a pool of mobile applications, in order to allow the end-users to access, retrieve and manage all the data, features and capabilities using them through specific *Cloud* services.

The requirements about the channels to interact and provide features, data and capabilities to the end users are strictly coupled with the main scopes of the modern solutions. In fact, in the modern solutions building process, the users provided interaction channels have primary roles.

In this case study analysis won't be treated the technological and functional details about the applications that have to be provided and supported by the target solution. The main reason related to this kind of analysis affects the main scope of this case study, so that to model and design their aspects as concern the solution design and model adopting an high level of abstraction.

The target solution to model and design has to be distributed on many *Cloud* platforms, because the *Hybrid Cloud* solutions are the main type to build affecting the modern and interesting scenarios as concerns the main one coupled with the *Cloud* context.

A primary requirement of this case study solution is to describe which *Cloud* platforms can be used to distribute the target solution so as to provide a description about the technological context related to the infrastructure, services and technologies.

Therefore, the details about the architecture modules split up and grouping will be described, in order to provide an holistic vision affecting the distribution of the target solution on many *Cloud* platforms. Thus, it's possible to prove the flexibility and the modularity of the proposed architecture applied to the case study solution.

The requirements of this case study solution has been described by an high level of abstraction and details with the aim of making it more complete and flexible as possible, so that all the details about the strategies and decisions affecting the design and model will be defined in the main scenario.

In the next sections, all the phases and strategies related to the modeling and design of the target solution will be defined and described. The main roles and steps to follow will be focused in order to provide the complete modeling and design of the target solution.

5.2 Design of the Solution

This section will present all the decisions, steps and strategies affecting the design and modeling of the target solution in terms of architectural design and related distribution in *Cloud* platforms.

All the details related to the solution design will be described with an high level of abstraction, in order to focus on the main scope of this case study. Furthermore, the modeling and design of the target solution will be split up in many steps, so as to focus on the specific aspects related to each part singularly.

The main steps of this phase will be the following ones:

1. **Solution Architecture Design**: this step affects the design and modeling of the target *Cloud* solution architecture using the proposed one, basing on the provided requirements.

2. **Solution Architecture Distribution**: this step affects the distribution of the architecture modules basing on the selected *Cloud* platforms analysis and the target solution architecture.

These steps can describe how the target solution can be modeled, designed and distributed. Thus, the main scope is to provide an holistic overview about it, basing on the primary and fundamentals aspects and concepts strictly coupled with the target context.

The distribution of the target solution through a single platform won't be treated because, being the selected *Cloud* platforms equivalent basing on the related analysis previously described (the chapter 3 presents the details about it), the distribution through a single platform can be done using anyone of the selected one.

5.2.1 Solution Architecture Design

This section will describe the design of the solution architecture, so as to provide its complete overview affecting the proposed architecture as concerns the main aspects and details.

The main scope is to show how the proposed architecture can be used as the target solution abstracting by the low level technological details, focusing on the main requirements and specifications.

First of all, the complete architecture has to be shown in order to provide an overview of it. Basing on it, the case study target solution has to be mapped on the proposed architecture, analyzing the target requirements.

Figure 5.1 shows the complete proposed architecture previously defined and described in chapter 2.

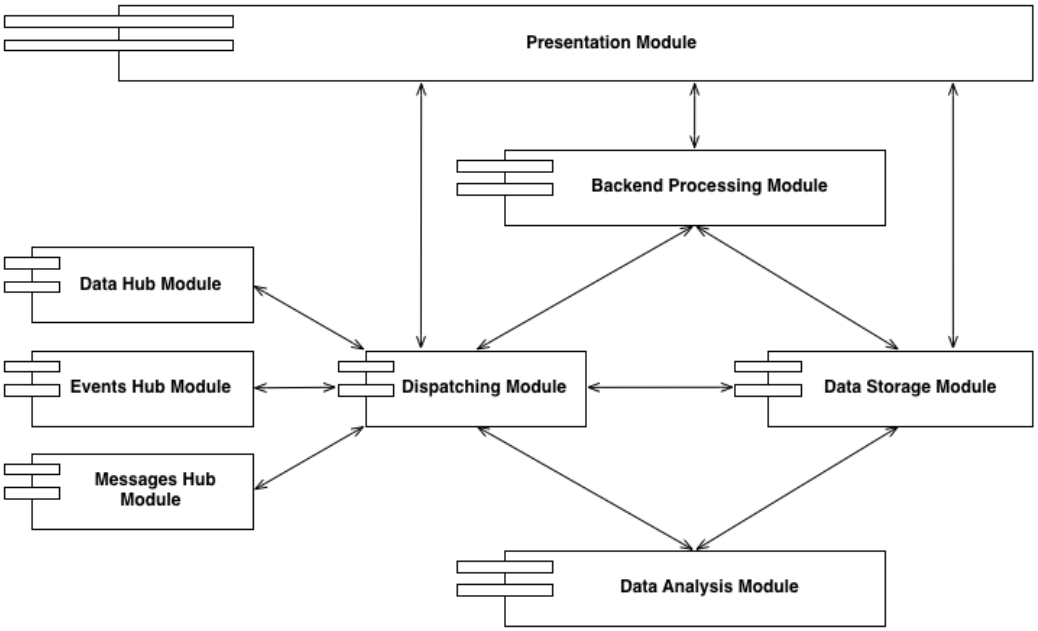

Figure 5.1: Architecture Main Overview.

Basing on the target solution requirements and overview, it will be described how the proposed architecture (in Figure 5.1) can be adopted to model and design it.

This architecture can be adopted because it can support and satisfy the target solution requirements in terms of features, context and scope. In fact, it will be described how the proposed architecture can be adopted to model and design the target solution.

Internet Of Things

It will be focused on the proposed architecture modules that can be used to build *IoT* solutions. In fact, one requirement is to support the *IoT* solutions design and development.

Figure 5.2 shows the proposed architecture where the interested modules for this context has been highlighted, thus, the focus will be put on those ones.

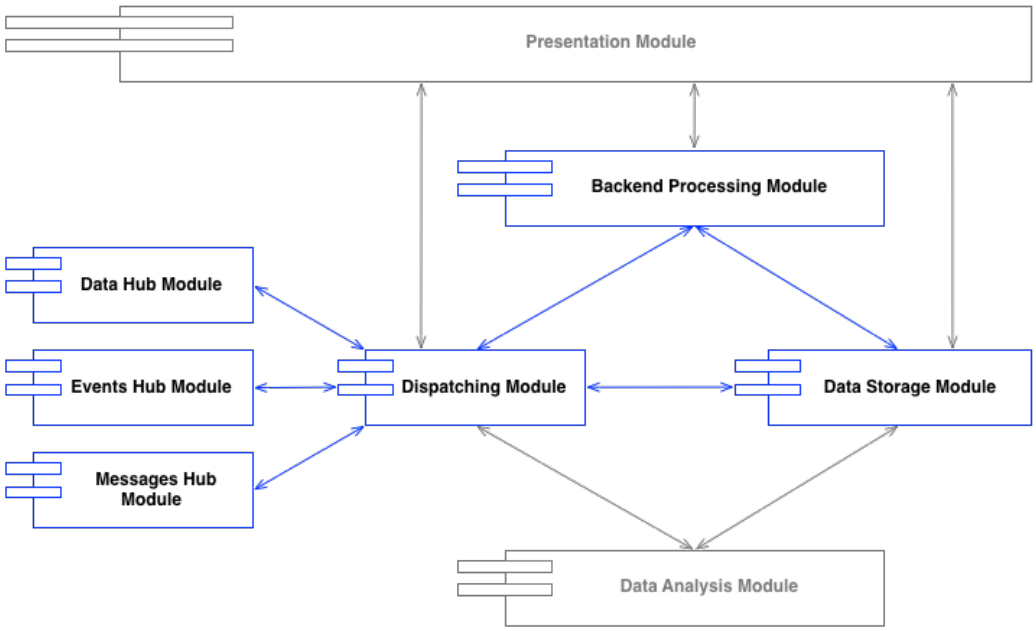

Figure 5.2: IoT Context Architecture Modules.

The proposed architecture can be adopted to build *Internet of Things* solutions, because it contains modules that can completely support this context solutions.

The selection of the highlighted modules shown in the picture 5.2 has been based on the main requirements of an *Internet of Things* solutions. In fact, it has been based on the more general and complete scenario about this context solutions.

The architecture modules that have to be used to build *IoT* solutions are the following ones:

- **Data Hub Module**: to support the interactions between the solution and the external devices and systems as concern the data. Thus, this module has been selected in order to support the data transmission with *IoT* devices.
- **Events Hub Module**: to support the interactions between the solution and the external devices and systems as concern the events occurrence. Thus, this module has been selected in order to support the events occurrence informations transmission with *IoT* devices.
- **Messages Hub Module**: to support the interactions between the solution and the external devices and systems as concern the messages. Thus, this module has been selected in order to support the messages transmission with *IoT* devices.
- **Dispatching Module**: to support the management, dispatching and routing of the data relating to the architecture internal modules. Thus, this module has been selected in order to support the communication between the architecture target modules.
- **Data Storage Module**: to manage the persistent data storage. Thus, this module has been selected in order to support the storage of the collected data from the *IoT* devices.
- **Backend Processing Module**: to support and manage the backend processes. Thus, this module has been selected in order to support the processes about the *IoT* devices and data continuous processing.

Therefore, the selected modules can completely satisfy the requirements and the needs of *Internet of Things* solutions, because all the aspects and concepts affecting this context are supported by these modules.

It has been considered the interaction with the end users and external systems through custom applications and services, because it will be detailed and described in the related part of this case study.

Big Data

It will be focused on the proposed architecture modules that can be used to build *Big Data* solutions. In fact, one requirement is to support the *Big Data* solutions design and development.

Figure 5.3 shows the proposed architecture where the interested modules for this context has been highlighted, thus, the focus will be put on those ones.

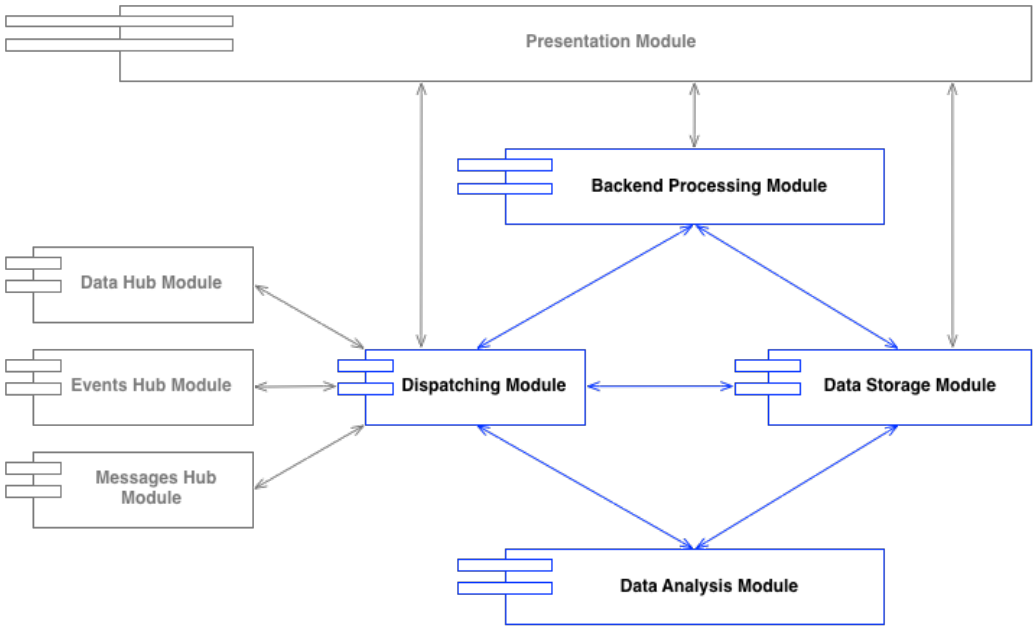

Figure 5.3: Big Data Context Architecture Modules.

Basing on Figure 5.3, the proposed architecture can be adopted to build *Big Data* solutions. In fact, the highlighted modules can completely support this context solutions.

The selection of the highlighted modules shown in Figure 5.3 has been based on the main requirements of an *Big Data* solutions. In fact, it has been based on the more general and extended scenario about this context solutions.

The architecture modules that have to be used to build *Big Data* solutions are the following ones:

- **Dispatching Module**: to support the management, dispatching and routing of the data relating to the architecture internal modules. Thus, this module has been selected in order to support the communication between the architecture target modules.
- **Data Storage Module**: to manage the persistent data storage. Thus, this module has been selected in order to support the storage of the interesting data for the target solution.
- **Data Analysis Module**: to support and execute the data analysis. Thus, this module has been selected in order to support the data analysis, primary scope of this context solutions.
- **Backend Processing Module**: to support and manage the backend processes. Thus, this module has been selected in order to support the processes that can help the *Big Data* processes and data continuous processing.

Therefore, the selected modules can completely satisfy the requirements and the needs of *Big Data* solutions, because all the aspects and concepts affecting this context are supported by these modules.

It has been considered the interaction with the end users and external systems through custom applications and services, because it will be detailed and described in the related part of this case study.

Applications & Services

It will be focused on the proposed architecture modules that can be used to build *Applications and Services* solutions. In fact, one requirement is to support the *Applications and Services* solution design and development.

Figure 5.4 shows the proposed architecture where the interested modules for this context has been highlighted, thus, the focus will be put on those ones.

Basing on Figure 5.4, the proposed architecture can be adopted to build *Applications and Services* solutions. In fact, the highlighted modules can completely support this context solutions.

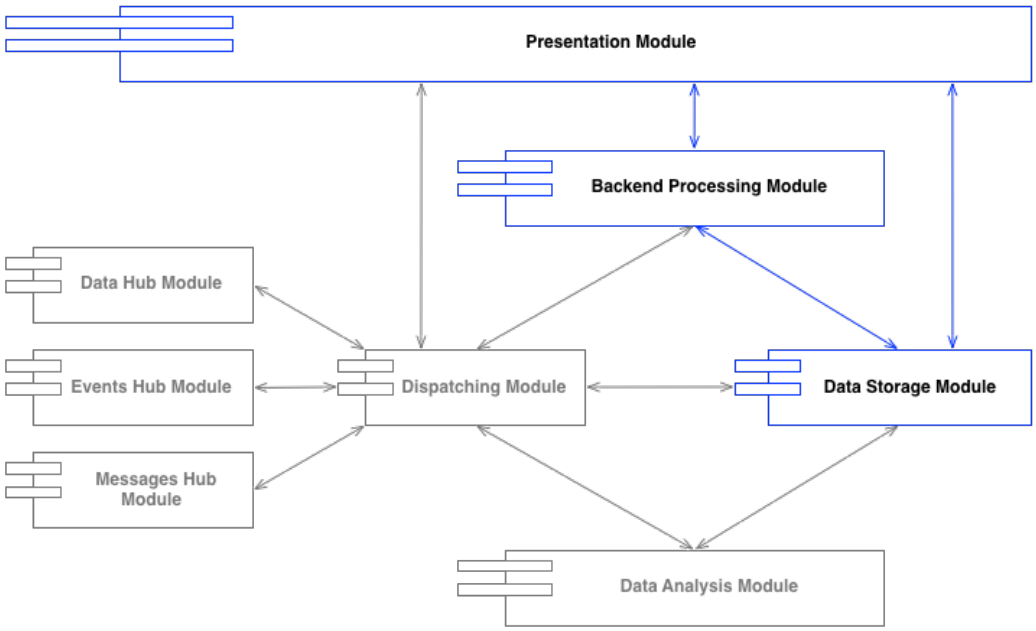

Figure 5.4: Applications & Services Context Architecture Modules.

The selection of the highlighted modules shown in Figure 5.4 has been based on the main requirements of an *Applications and Services* solutions. In fact, it has been based on the more general and extended scenario about this context solutions.

The architecture modules that have to be used to build *Applications and Services* solutions are the following ones:

- **Presentation Module**: to provide data, features and capabilities related to the solution. It will provide tools, services and applications to manage the solution capabilities and features. Thus, this modules has been selected so as it has the primary role for the target solution.
- **Data Storage Module**: to manage the persistent data storage. Thus, this module has been selected in order to support the storage of the interesting data for the target applications and services.
- **Backend Processing Module**: to support and manage the backend

processes. Thus, this module has been selected in order to support the processes that can help the *Applications and Services* processes and data continuous processing.

Therefore, the selected modules can completely satisfy the requirements and the needs of *Applications and Services* solutions, because all the aspects and concepts affecting this context are supported by these modules.

The design and model of this solution part could be a part of other contexts but it has been decided to manage it in a specific section, in order to abstract from the solution scope in terms of context and features.

5.2.2 Solution Architecture Distribution

This section will describe how the solution architecture can be distributed on the selected *Cloud* platforms, in order to provide all the details about the strategies and the decisions about the distribution of the architecture modules.

This analysis derives from the main requirements of this solution, because one of them is related to the distribution of the solution architecture on many *Cloud* platforms. Adopting many strategies and methodologies, an holistic overview of the architectural distribution of the target solution will be provided.

The solution architecture distribution will be described basing on the *Cloud Platforms Analysis* (done in the chapter 3). The selected and analyzed *Cloud* platforms will be filtered to recommend which ones can be adopted to distribute and manage the target modules, basing on the pool of platforms provided services and the related quality.

The main strategy about the grouping of the architecture modules has been based on the modules selection done in the previous section (the section 5.2), focusing on the context treated as concern *Internet of Things*, *Big Data* and *Applications and Services*.

Furthermore, the decision to design and model the distribution of the architecture has been based on the main scope of this case study, because one of the advantages of the proposed architecture affects the modularization of the solutions. In fact, analyzing this factor, the architecture distribution and the related scenario, it's possible to prove this advantage.

Figure 5.5 represents the distribution of the architecture modules where

the modules has been highlighted basing on the grouping; in fact, the modules highlighted using the same color will be grouped together.

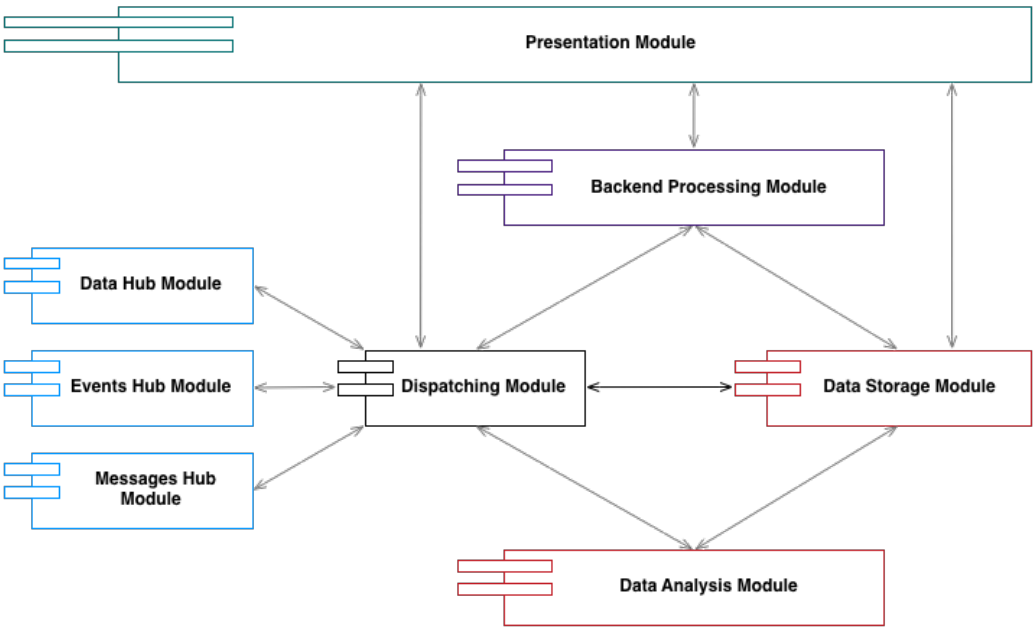

Figure 5.5: Main Distribution of the Architecture Modules.

Basing on Figure 5.5, the architecture modules can be grouped basing on their objectives and context. In fact, the grouping strategies have been defined focusing on the modules features and capabilities.

The groups defined are the following ones:

- **Hubs Group**: it's the group which contains the modules that act as hubs to interact and communicate with external devices and systems. The modules contained in this group are the *Data Hub Modules*, *Events Hub Modules* and the *Messages Hub Modules*.

- **Data Group**: it's the group which contains the modules that act to support, execute and manage all the aspects and operations affecting the data. The modules contained in this group are the *Data Storage Module* and the *Data Analysis Module*.

- **Dispatching Group**: it's the group which contains the modules that act as dispatcher, broker and manager for the data flow. The unique module contained in this group is the *Dispatching Module*, because it's the only one dedicated to these aims.

- **Backend Group**: it's the group which contains the modules that act as dispatcher, broker and manager for the backend processes, tasks and jobs. The unique module contained in this group is the *Backend Processing Module*, because it's the only one dedicated to these goals.

- **Presentation Group**: it's the group which contains the modules to provide data, features and capabilities related to the solution. The unique module contained in this group is the *Presentation Module*, because it's the only one dedicated to these objectives.

These groups have been defined basing on the scopes and contexts affecting the architecture modules. In fact, this approach allows to group together the modules affecting the same contexts (eg. the modules affecting the data context have been grouped in the *Data Group*), so as to locate them on the same *Cloud* platform.

When the solution has to be distributed through *Cloud* platforms, the groups can be located on the one which provides the best support to the context affecting the target group (eg. the *Data Group* should be located on *Azure*, because it provides the best support to the data context through *Azure SQL Database Service* [45]).

The distribution strategy affecting the selection of the target platforms has to be based on the modules scopes, features and capabilities. In fact, the selection can be based focusing on the quality of the platforms as concerns the groups and related modules.

Following these strategies and approaches, it's possible to build solutions about the main contexts such as *Internet of Things*, *Big Data* and *Applications & Services*. Thus, it's possible to build the solutions affecting these contexts, because the target groups can be distributed using anyone of the selected *Cloud* platforms previously analyzed (reference to the chapter 3).

Finally, the defined and described strategy in this analysis can be defined as the best one in the solutions building processes. In fact, applying this best practice can support designers and developers to select the platforms to use to distribute the target modules groups.

5.3 Evaluation

In this section will be provided an evaluation about the adoption of the proposed architecture to design the target solution and the distribution strategy affecting it. Thus, the main aim of this evaluation is to provide an holistic evaluation of the aspects affecting this case study and the proposed architecture appliance.

The proposed architecture can be used to design and model any kind of *Cloud* solutions as described in the specific section affecting the solution architecture design. In fact, the proposed architecture modules can be used to design solutions affecting the *Internet of Things*, *Big Data* and *Applications & Services*.

A relevant property related to the proposed architecture is the modularization affecting it. Analyzing this case study and the related solution, it's possible to highlight and prove that it's highly modularized. In fact, in this case study the architecture has been completely split up in many groups, thanks to the modularization of it, treating all the modules independently.

The main advantages given by this modularized architecture are the following ones:

- **Modularization**: the solution can be split up in many modules (the proposed architecture ones) and they can be managed in an easier way than the unique modules group approach.
- **Independence**: each solution module can be managed independently from the others, so there is no direct influence between them. Thus, the modules can be managed in a completely independent way.
- **Maintenance**: each solution module can be developed and maintained independently from the others, so the developments and maintenance actions can't influence more modules but only the target one. Thus, the modules can be developed and maintained in a completely independent way, except for the direct and required interactions.

All these advantages are strictly coupled with the proposed architecture advantages, so the described ones affecting it in the dedicated chapter (the chapter 2, providing the complete definition and the description of the proposed architecture) have been verified in this analysis.

The architecture distribution is particularly interesting as concerns the location of the defined groups on many *Cloud* platforms. In many solutions contexts, the distribution is a charge of a single platform and this approach could be critical in terms of support and improvement.

Following the *Hybrid Cloud* solutions building approach can provide many advantages affecting the diversification of the platforms selection. In fact, this approach can help and improve the quality and the cross-strategy affecting the solution distribution on many *Cloud* platforms.

The main advantages provided by this distribution strategy are the following ones:

- **Best Features for Platforms**: it's possible to distribute the architecture modules on the platforms that provide the best features in terms of quality, performances, improvement, support and costs. Thus, when the module groups have to be distributed, the target platforms selection has to be based on the target features required and the related costs (eg.the *Hub Group* can be distributed on *Azure*, because it provides many advanced services like *IoT Hub* that fits better than the other platforms with the target group features [25]).

- **Multi-Platform Compatibility**: the modules have to be developed in order to make them cross-platforms compatible, because this distribution strategy requires this approach. Thus, it's possible to develop the target modules, in order to make them as cross-platforms.

- **Lock-In Avoidance**: basing on this distribution strategy, the modules development has to be based on a cross-platforms technology as concerns selected ones. Thus, the groups location can be changed when the conditions are not suitable for the builder of the target solution, the modules group can be moved to another platform, because the design and implementation are equivalent as concerns the main capabilities and features (eg. when the current platform billing is became too high, the modules can be moved to another less expensive one).

All these advantages derive from the proposed architecture advantages, so the described ones affecting it in the dedicated chapter (the chapter 2, where there is the complete definition and description of the proposed architecture).

This case study can be evaluated as successful, because the main scenario treated, the related strategies, approaches and methodologies verify many advantages, features and capabilities of the provided architecture.

Furthermore, in the treated solution in this case study is particularly extended in terms of scopes, features and capabilities. Thus, the success of this case study is extremely important, because it verifies that the target architecture can be used to design and model the common solutions affecting the *Cloud* context.

Finally, this case study verifies that the proposed architecture can be adopted to model and design many kind of *Cloud* solutions. In fact, in the scenarios described in this chapter there are cases affecting *Internet of Things*, *Big Data* and *Applications & Services*, so all the main *Cloud* contexts have been involved.

Therefore, the proposed architecture can be adopted in many contexts and scenarios, thanks to its modularization, flexibility and adaptivity basing on the requirements and needs specific for the target solution to model and design.

5.4 Implemented Case Study

This section will describe how a Web based system can be redesigned by using the proposed architecture so as to prove how this architecture can be exploited to model and design a real world solution.

The selected solution which will be redesigned by using the proposed architecture is *UniSAS* (*Universal Structure Access System*), a system designed and developed following the best practices, patterns and methodologies related to the Web context.

First of all, the *UniSAS* system will be described focusing on its structure, feature and capabilities. In this phase, the system will be described mainly focusing on its components and interactions.

Then, the main steps and issues related to the redesign of the system adopting the proposed architecture will be described. So, it will be provided the complete work flow related to the redesign of the system adopting the treated architecture.

Finally, the redesigned system will be described focusing on its characteristics, features and capabilities. In this phase the focus will be set on the main differences between the Web based solution and the *Cloud* based on designed adopting the proposed architecture.

The *UniSAS* solution has been selected as the best one to prove and validate the proposed architecture thanks to its properties.

In fact, the selection has been based on the following characteristics:

- **Generated Data Issues**: it generates an large amount of data due to its objectives and capabilities. Furthermore, these data could have many types, structures and formats, so the related complexity will be greater than other contexts. This factor can be linked to the *Big Data* context.

- **Scalability and Performance Issues**: it requires an high scalable platform due to its distribution level. It has to be also extremely performing because the data and interactions management have to be done in a reduced amount of time (eg like a real time system). These factors can be connected to the *Cloud* context.

- **Devices Interaction Issues**: it requires an extended pool of devices that have to interact and communicate with other system entities. This factor can be linked to the *Internet of Things* context.

This section will present the scenario related to the design of a real world implemented solution in the *Cloud* context adopting the treated architecture. Thus, it can prove and validate all the analysis presented in this thesis.

5.4.1 UniSAS Overview

UniSAS is a system thought to improve the accessibility level of any building. In fact, this system acts as an helper to automatically open, close and manage any kind of doors and other structure that could limit the accessibility and mobility of a person.

The main objective is to help all the people situated in a building but, focusing on the accessibility and mobility issues, the main targets of this system are the disabled people (eg wheelchair users) or those people with physical impairments.

Furthermore, this system can act also as an access controller so as to control and manage access privileges. In fact, all the rules related to places and rooms accessibility can be managed by using a dedicated tools.

Focusing on the building management processes and analysis, *UniSAS* can be used to collect and manage data so as to improve all the maintenance actions and processes. This capability has a primary role in modern solutions because all the processes are managed through specific data analysis.

Therefore, *UniSAS* can:

1. **Control and Manage Accesses**: it acts as access controller in order to allow and guarantee the access only to the authorized people.

2. **Automatic Move of Access Structures**: when an authorized user has been detected near an access structure (eg a door), it will be opened in order to allow the access to the target user. When the user has moved into the target place, the access structure will be automatically closed.

3. **Monitor the Accesses**: all the accesses can be registered and logged with the aim to collect statistics and data in order to support the building management processes.

All the *UniSAS* main characteristics and capabilities are not related to specific access structures because they can be fitted for any structure.

The Figure 5.6 shows the main scenario related to the *UniSAS* system. In this scenario, when a user has to access through a structure (eg a door), it has to go close to the door in order to enter in the sensor device range (eg a low energy device such as an *iBeacon* [67]).

When the user mobile device detect the wireless one, it send an access request to the *UniSAS Core Services*. Then, if the user has the needed privileges to access to the locale, the access accorded response will be sent to structure the micro-controller (eg the *Raspberry Pi Micro-Controller* [68]).

Finally, the micro-controller will open the target access structure.

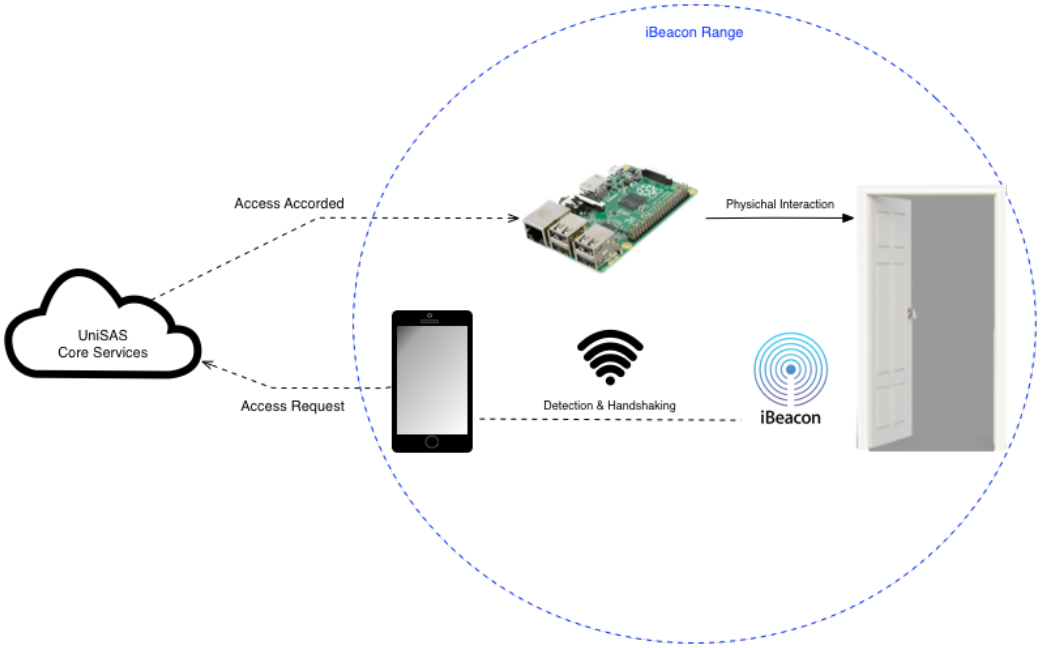

Figure 5.6: UniSAS System Main Overview.

In the following paragraphs, all the design and implementation details will be presented and described in order to provide a complete overview about the following factors:

1. the current state of the system and the related details about structure, interactions and characteristics, including issues and limitations;

2. the redesign work flow steps required to port the system in a solution built adopting the proposed architecture;

3. the final state of the system built adopting the treated architecture and the related details about structure, interactions and characteristics, including issues and limitations.

All the details related to this implemented case study will be presented basing on the major scopes and characteristics of the proposed architecture because the primary objective of this section is to validate and verify it.

In this case study, the details about the external devices (eg the microcontrollers to manage and move the access structures like doors) have been left out because the main target is to analyze the system components that could be moved to the *Cloud* adopting the proposed architecture.

5.4.2 UniSAS Web Solution

The initial implementation of the *UniSAS* system has been realized as *Web based*, so it's completely provided and managed through the Web. In fact, all the tools, applications and platforms of the system has been developed using Web technologies, patterns and best practices.

The Web implementation of *UniSAS* is composed by the following parts:

- **Web Application**: to provides capabilities and features to the administration users (eg it allow to manage user profiles privileges and configurations) and system end users (eg it's able to manage the opening and closing of the access structures).

- **Mobile Applications**: to provide all the *Web Application* features and capabilities as concerns the administration and system end users profiles through *Mobile Devices* (eg they have been implemented for *Android* [69] and *iOS* devices [70]).

- **Web Services**: these services provides all the capabilities and features required by the *Web* and *Mobile Applications* in order to centralize the system business logic in this limited pool of entities.

The system parts defined as concerns the *Web* solution will be detailed and analyzed in order to outline their *Cloud* porting work flow adopting the proposed architecture.

UniSAS Web Solution Architecture

All the entities composing the *UniSAS* has been modeled, designed and implemented by adopting the Web context best practices and patterns.

The Figure 5.7 shows the *UniSAS Web Solution* main architecture.

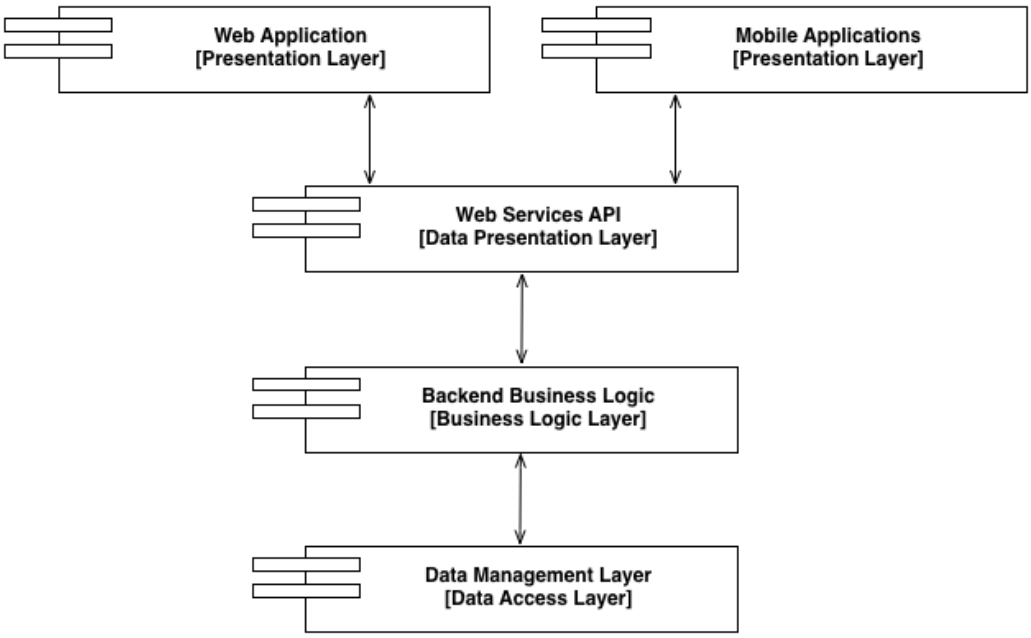

Figure 5.7: UniSAS Web Solution Architecture Main Overview.

The architecture shown in the Figure 5.7 has been designed focusing on the *Multi-Tier Architecture* (or *Multi-Layered Architecture*) [71].

This architecture choice has been based on the system design best practices, because it improves the solution maintainability and upgradability.

1. **Data Management Layer**: tier to manage the data storage, related components and entities so as to control the data flow in the treated system. This level has to manage all the features and capabilities related to the data storage, information flow and store access and management.

2. **Backend Business Logic Layer**: tier to manage and support the system business logic in order to provide its features and capabilities. This level has to manage all the system logic complexity and work flow in order to centralize it and manage in an better way.

3. **Web Services API Layer**: tier to provide system capabilities, features and properties as Web services in order to provide them to the target layers. Furthermore, this layer has to manage the data flow between the system tiers.

4. **Web Application Layer**: tier to support and manage the system capabilities as concerns the end users interactions and management through the target Web application. In fact, this level will be used by the end users to manage the system and the provided features using a Web browser.

5. **Mobile Applications Layer**: tier to support and manage the system capabilities as concerns the end users interactions and management through the target mobile devices applications. In fact, this level will be used by the end users to manage the system and the provided features using a the target application on the mobile devices.

Furthermore, the internal architecture of the *Web Application*, *Web Services* and *Mobile Applications* has been designed basing on the *Multi-Tier Architecture* in order to achieve the related advantages.

All the architecture components have been designed and developed adopting the main patterns related to the target technologies software engineering (eg the *Web Services API* have been designed adopting the *Model-View-Controller Pattern* [72]).

The components internal structure won't be treated in this section, because the main objective is to describe the solution architecture, so all the internal details won't be specifically treated.

All the solution parts have been developed using many frameworks and technologies to avoid vendor lock-in and obtain an heterogeneous solution. Adopting this strategy, it has been possible to evaluate and select the best technologies to develop each component so as to improve the overall quality.

UniSAS Web Solution Evaluation

The evaluation about the *UniSAS* Web solution will be provided so as to describe its properties, characteristics and capabilities.

Basing on the main scope of this analysis, this evaluation will be focused on the system critical aspects and limitations in order to correlate the system properties within its greats issues.

The system main issues have previously introduced in the solution overview (in the section 5.4.1) will be analyzed focusing on the system details and components.

1. **Scalability Issue**

   The system has been designed and developed basing on the main scenario where a system instance has to manage the access structures of a single or a limited pool of buildings.

   This limitation has been defined so as to guarantee high system performances and reliability. In fact, the huge communications amount could destabilize or compromise the system capabilities and features.

   This limitation is strictly coupled withing the scalability because this is a fundamental characteristic to manage extended buildings pools.

   Improving the system scalability will ensure its reliability and availability regardless of the number of target buildings and access structures to manage.

2. **Performances Issue**

   The limitation regarding the system managed access structures and buildings could be coupled within the performances issue because the computational resources have to be commensurate with the target structures pools.

   The scalability issues are also valid as concerns the performances because these properties are strictly coupled. In fact, the huge communications amount could restrict and limit both of those properties.

   Thus, increasing the system scalability will improve the system performances as concerns their main aspects such as services responses time and reactivity.

3. **Generated Data Issues**

   This system has to manage an extended pool of building and access structures so it will generate large data stores.

   The generated data will have different types, structures and formats because they are collected and generated by an heterogeneous ecosystem of devices and technologies.

   Therefore, the system data management and related components have to be enhanced so as to improve its data management capabilities and features.

4. **Device Interaction Issues**

   This system requires an extended pool of devices that have to interact and communicate with other system entities. In fact, these devices have to manage the access structures and collect related data.

   The huge amount of communications between the system devices due to the needed interactions could limit the system overall performances.

   In fact, the consequences related to these communications will impact on the system devices performances and characteristics.

   Thus, improving the communications and interactions between the system devices will enhance the overall performances and the related responses time and reactivity.

The described issues are strictly coupled with the *Cloud* because all of them could be referred to this context technologies and platforms.

Therefore, each introduced issue could be related to one *Cloud* context. More in details, focusing on the main references, there are the following relations:

- **Cloud**: can improve and increase the solutions performances and scalability. This context can be directly referred to the issues described in the scalability 1 and performances 2 issues. Thus, it can solve those issues, enhance the overall quality, reliability and reactivity.

- **Big Data**: can increase and improve the quality and performances of the data management systems. Thus, it can enhance the main quality related to the data store and management, while solving the referred issue 3 and the derived limitations.

- **Internet Of Things**: can improve the quality and the performances related to the device management and referred communications and interactions. Thus, it can enhance the system communications performances as concerns the devices, while solving the referred issue 4 and the derived limitations.

The main *Cloud* contexts have been referred directly to the *UniSAS* Web solution in order to check if it could be improved through the its redesign and porting to the target environment, platforms and technologies.

Hence, focusing on this case study main objective, the treated Web solution won't be ported directly to the *Cloud* because the main objective is to improve its overall quality, features and capabilities.

In fact, when a system is brought to the *Cloud*, the only benefits and advantages are coupled with the performances and scalability due to exploiting of the *Cloud* features related to those aspects.

The *UniSAS* Web solution will be completely redesigned so as to fit it to the *Cloud* context. More in details, the redesign will be based on the proposed architecture in order to verify its overall quality, advantages and benefits through its adoption in this Web solution porting case study.

5.4.3 UniSAS Redesign Work Flow

The *UniSAS* system porting to the *Cloud* adopting the proposed architecture will be described in this section, focusing on this work flow steps.

The porting will be presented by describing each layer porting work flow in order to define them singularly so as to describe the tiers focusing on the main aspects, features and characteristics.

The tier sequence will be the following one:

1. **Data Management Layer**: it's the first tier which has to be redesigned because the data management and the information flown are primary assets for the system. Thus, this layer has to manage all aspects, features and complexity related to the system data.

2. **Backend Business Logic Layer**: it's the second tier which has to be redesigned because it centralize the system main features, capabilities and characteristics. So, this layer has to manage the business logic related to the system features, characteristics and capabilities.

3. **Web Services API Layer**: it's the third tier which has to be redesigned because one of the main *Cloud* key aspects is the capabilities providing through services. Thus, this layer has to provide and support features and capabilities to the applications tiers.

4. **Web Application Layer**: it's the fourth tier which has to be redesigned because the *Cloud* applications have to be provided in the same way as Web applications and they have to be served and supported by services.

5. **Mobile Applications Layer**: it's the last tier which has to be redesigned because the mobile applications will be supported by the *Cloud* applications and services. In fact, the mobile applications are partially hybrid, so these parts are completely presented through Web components (eg through Web views are provided the administration environments); other parts, instead, are developed as completely native (eg the login and credentials management).

All the *UniSAS* Web solution tiers can be redesigned so as to be implemented adopting the proposed architecture. All of them can be developed adopting many modules, focusing on the main characteristics, features and advantages obtained through this porting, as it will be described in the following.

Data Management Layer Porting Work Flow

The *Data Management Layer* porting has to be done basing on many factors and characteristics about the *Database* type. In fact, this tier can be ported adopting the *Data Storage Module*.

The Figure 5.8 shows the proposed architecture target module to support the *UniSAS* system *Data Management Layer*.

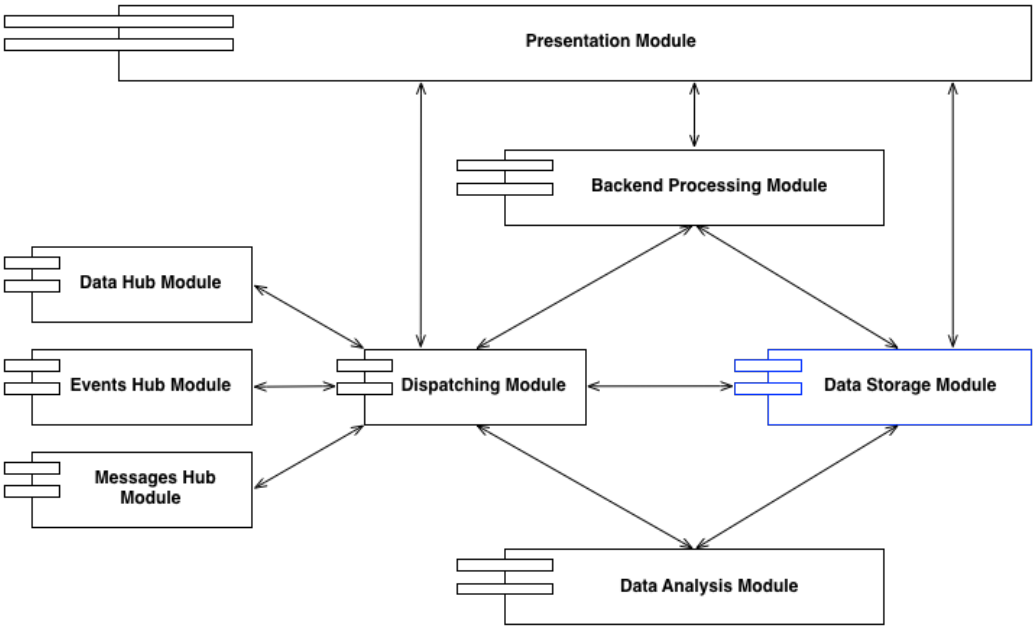

Figure 5.8: UniSAS Cloud - Data Management Layer.

The *Database Storage Module* can completely support all the data storage features because its requirements are a *Data Storage Module* capabilities restricted pool.

In fact, the architecture module can extends the *UniSAS* storage features because the target module has been designed basing on the main *Cloud* requirements (eg the *UniSAS* data storage Web solution didn't have any support to the *BLOB Storage*).

Finally, the proposed architecture *Data Storage Module* adoption can support and improve the *UniSAS* system *Data Management Layer*.

Backend Business Logic Layer Porting Work Flow

The *Backend Business Logic Layer* porting has to be done basing on many factors and characteristics about the target features and capabilities that

have to be supported and provided.

This tier provides many types of capabilities and features as concerns their contexts and scopes. In fact, it's not possible to port this layer adopting a single architecture module.

Focusing on this tier main features, it's necessary to adopt many proposed architecture modules because each architecture module has specific contexts, capabilities and scopes.

Therefore, this layer can be ported adopting a composition of many modules that can provide the target features and satisfy the main requirements.

The Figure 5.9 shows the target modules to adopt so as to support the *UniSAS* system *Backend Business Logic Layer*.

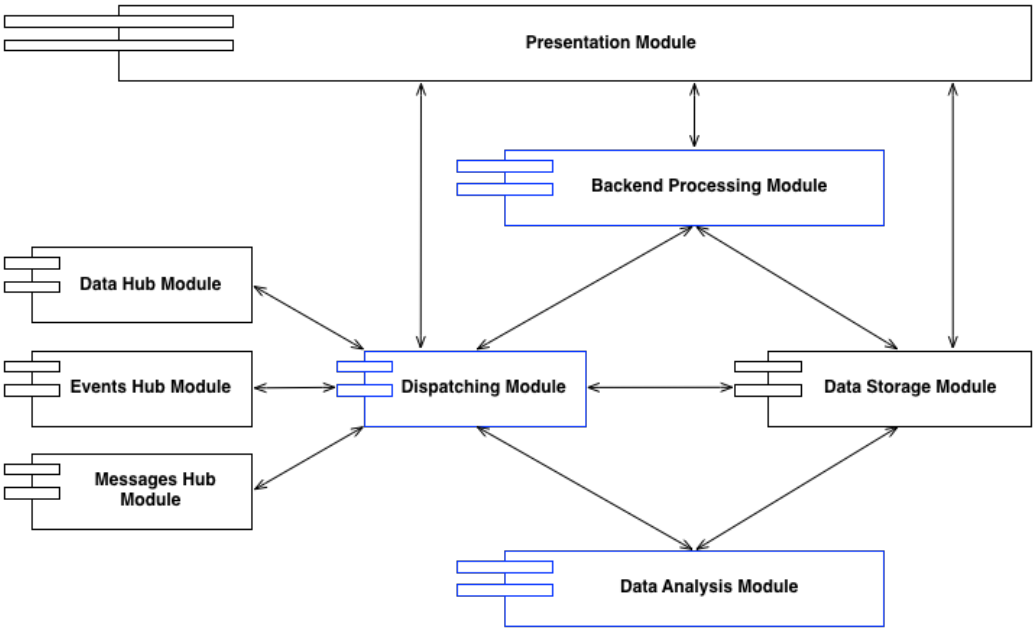

Figure 5.9: UniSAS Cloud - Backend Business Logic Layer.

The architecture modules that have to be adopted to port this tier are the following ones:

- **Data Analysis Module**: to support all the analysis related to the system. In fact, *UniSAS* provide statistics, reports and many kind of informations derived from the collected data through end users requests and monitoring actions (eg the reporting activities referred to the access structures actions required by end users).

- **Backend Processing Module**: to support all the back end processes and entities to allow the execution of target work flows. Adopting this module allow to cover all the tier capabilities that require specific procedures execution (eg the *UniSAS* background execution related to the devices firmware consistency check referred to its features).

- **Dispatching Module**: to support all the interactions between the architecture modules. In fact, this tier is the focus of the system features so it has to manage, control and tune all the communications. Thus, this module is part of the core as concern this layer and the entire system.

This *Backend Business Logic Layer* has to interact with the *Web Services API Layer*, *Web Application Layer* and the *Mobile Application Layer*.

In details, the *Dispatching Module* adopted in this tier has to be part of the other layers because this module is responsible to all the core interactions between any architecture module.

Finally, this tier represents the *UniSAS* core layer because all system tiers features and capabilities are supported by it.

Web Services API Layer Porting Work Flow

The *Web Services API Layer* porting has to be done basing on many factors and characteristics about the target features and capabilities that have to be supported and provided.

This layer has to provide all the features related to the device management and external systems communications (eg it has to manage the access structures devices communications to require their opening and the mobile applications information supplying).

It's not possible to port this tier by adopting a single architecture module due to its features extended pool. In fact, it's necessary to adopt many proposed architecture modules, because each architecture module has specific contexts, capabilities and scopes.

Therefore, this layer can be ported adopting a composition of many modules that can provide the target features and satisfy the main requirements.

The Figure 5.10 shows the target modules to adopt so as to support the *UniSAS* system *Web Services API Layer*.

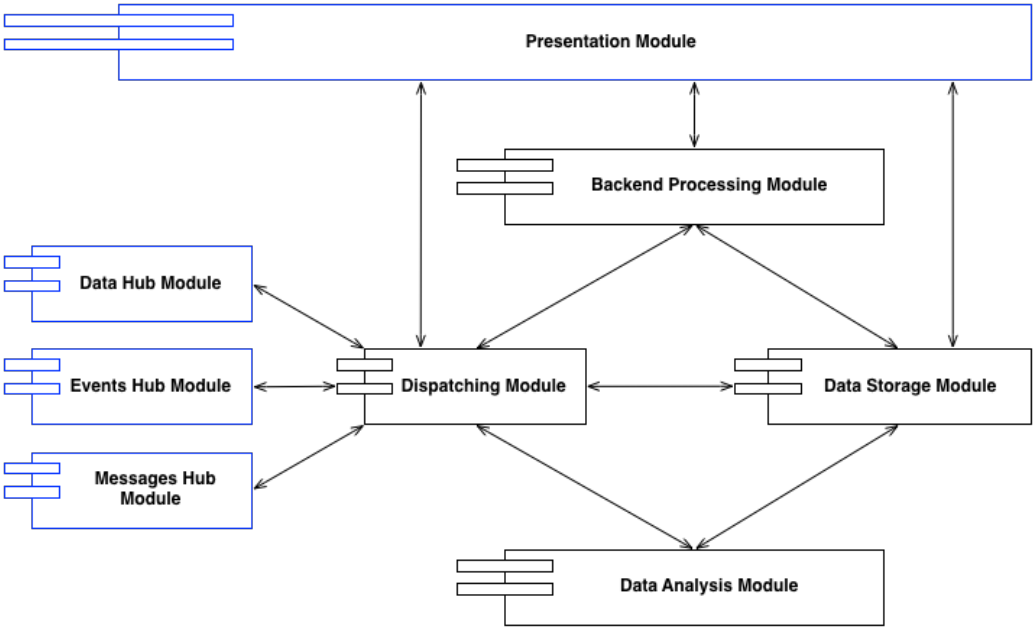

Figure 5.10: UniSAS Cloud - Web Services API Layer.

The architecture modules that have to be adopted to port this tier are the following ones:

- **Events Hub Module**: to support the communications between the platform and the system devices related to the events occurrence informations (eg the informations related to the failure about a device operation or action).

- **Data Hub Module**: to support the data communication between the platform and the system devices (eg the informations referred to the firmware upgrade process).

- **Messages Hub Module**: to support the communications between the platform and the devices as concerns the messages referred to interesting informations about system working (eg the collected informations about the device activities).

- **Presentation Module**: to support the system features and data presentation that can be usable and accessible through *Web Services* (eg the services to provide informations about the system working).

Focusing on the *Presentation Module* showed in the Figure 5.10, it won't be ported completely because only few modules are required to provide this layer features.

The Figure 5.11 presents the *Presentation Module* required components that have to be adopted so as to support the *Web Services* provided by the *UniSAS* system *Web Services API Layer*.

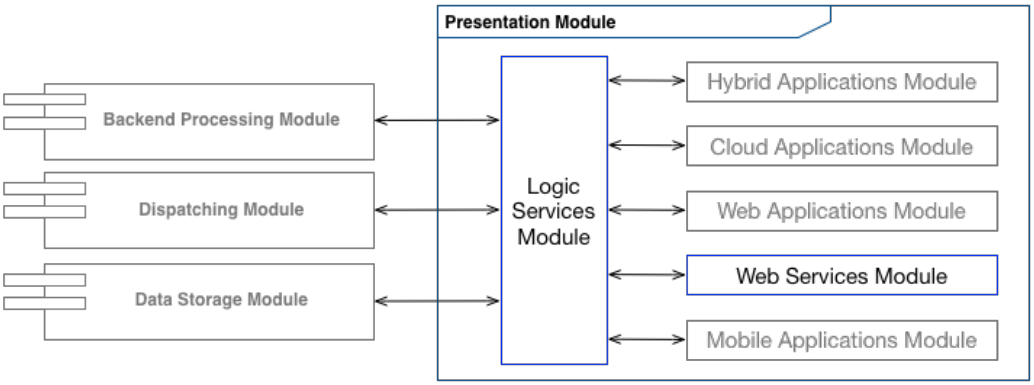

Figure 5.11: UniSAS Cloud - Presentation Module Web Services Layer.

It has been decided to adopt only the essential components related to the *Presentation Module*, because the complete porting could interfere with other layers and increase the system complexity.

Therefore, the essential components adoption would reduce the porting complexity because there are fewer entities to manage.

The *Presentation Module* components that have to be adopted so as to support the *Web Services* provided by this tier are the following ones:

- **Web Services Module**: to support and expose all the *Web Services* that this layer has to provide (eg the services to provide informations about the system working).
- **Logic Services Module**: to support the interactions between the *Web Services* and the other modules and components. It has to act as work flow and information broker so as to reduce the components complexity (eg the *Web Services* have to interact with the *Backend Business Logic Modules* through this component).

The *Data Hub Module*, *Events Hub Module*, *Messages Hub Module* and *Presentation Module* have to communicate with the *Dispatching Module*, because it's the system core dispatcher.

This module has been ported relating to the *Backend Business Logic Layer*, so it has to communicate with this tier modules. So, its complexity will be increased but it's necessary because it's the communications core.

The *Logic Services Module*, instead, will be ported as specific for this layer features. In fact, this strategy allow to reduce this component overall complexity.

Finally, this tier represents the *Web Services* provider of *UniSAS* system, in order to support all the devices and external systems needs as concerns this platform informations and features.

Web Application Layer Porting Work Flow

The *Web Application Layer* porting has to be done basing on many factors and characteristics about the *Web Application* features.

This tier can be ported adopting the *Presentation Module* because this tier refers to the *UniSAS* system *Web Application*.

Basing on the *Web Application* and its characteristics, all its requirements and needs can be satisfied by the target module because they are the main objectives of this specific module.

The Figure 5.12 shows the proposed architecture target module to support the *UniSAS* system *Web Application Layer*.

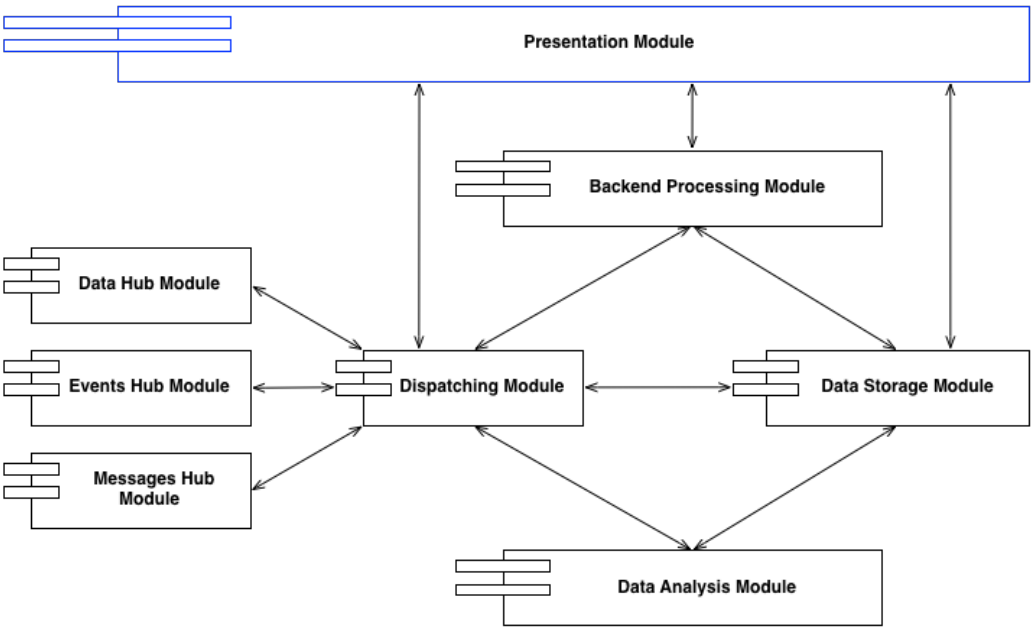

Figure 5.12: UniSAS Cloud - Web Application Layer.

The *Presentation Module* is the only required module to port the *Web Application Layer* to the *Cloud* adopting the proposed architecture. In fact, one of this module main objectives is to support all features of this kind of applications.

Furthermore, the *UniSAS Web Application* doesn't require all the *Presentation Module* capabilities because it has an extended pool of features to support also other applicative context (eg it has a specific module dedicated to the *Web Services* supplying and support).

In fact, the main scope of this tier is to provide and support all the features related to the *Web Application* that represents the interaction endpoint with the *UniSAS* end users.

Therefore, focusing on this aspect and the *Presentation Module* showed in the Figure 5.12, it won't be ported completely because only few modules are required to provide this layer features.

The Figure 5.13 presents the *Presentation Module* required components that have to be adopted so as to supply the *Web Application* provided and supported by the *UniSAS* system *Web Application Layer*.

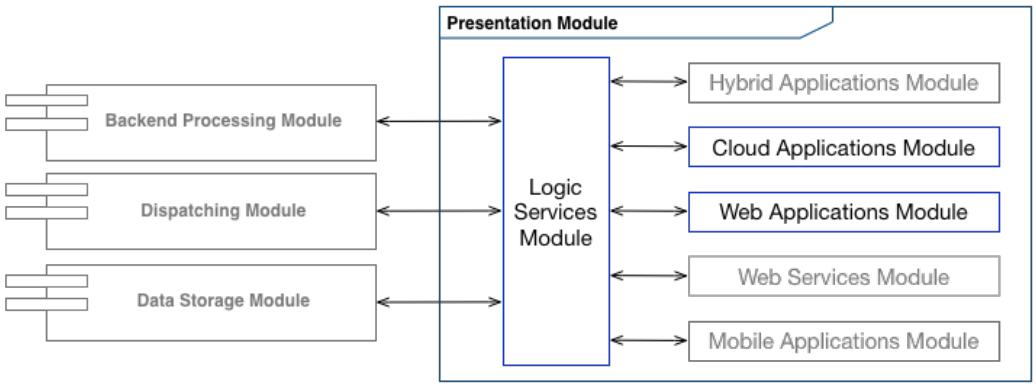

Figure 5.13: UniSAS Cloud - Web Application - Presentation Module.

It has been decided to adopt only the essential components related to the *Presentation Module* because the complete porting could interfere with other layers and increase the system complexity.

Therefore, the essential components adoption would reduce the porting complexity because there are fewer entities to manage.

The *Presentation Module* components that have to be adopted so as to support the *Web Application* provided by this tier are the following ones:

- **Web Applications Module**: to support the *Web Application* that this layer has to provide (eg the application to manage the system access privileges and roles through a Web browser).
- **Cloud Applications Module**: to support the *Cloud Application* that this layer has to provide (eg the application to manage the system directly through *Cloud* platforms tools).
- **Logic Services Module**: to support the interactions between the target *Applications* and the other modules and components. It has to act as work flow and information broker so as to reduce the components complexity (eg the *Applications* have to interact with the *Backend Business Logic Modules* through this component).

Finally, this tier represents the *Web* and *Cloud Applications* provider of *UniSAS* system, in order to support all the end users needs as concerns this platform informations and features.

Mobile Applications Layer Porting Work Flow

The *Mobile Application Layer* porting has to be done basing on many factors and characteristics about the *Mobile Applications* features.

This tier can be ported adopting the *Presentation Module* because this tier refers to the *UniSAS* system *Mobile Applications*.

Basing on the *Mobile Applications* and their characteristics, all its requirements and needs can be satisfied by the target module because they are the main objectives of this specific module.

The Figure 5.14 shows the proposed architecture target module to support the *UniSAS* system *Mobile Applications Layer*.

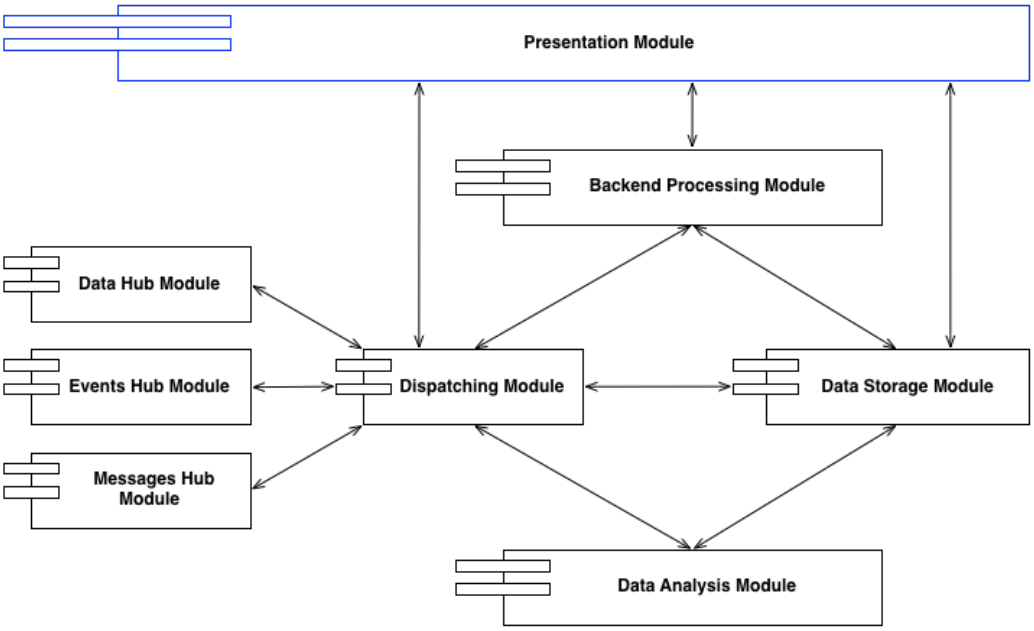

Figure 5.14: UniSAS Cloud - Mobile Applications Layer.

The *Presentation Module* is the only required module to port the *Mobile Applications Layer* to the *Cloud* adopting the proposed architecture. In fact, one of this module main objectives is to support all features of this kind of applications.

Furthermore, the *UniSAS Mobile Applications* doesn't require all the *Presentation Module* capabilities because it has an extended pool of features to support also other applicative context (eg it has a specific module dedicated to the *Cloud Applications* supplying and support).

In fact, the main scope of this tier is to provide and support all the features related to the *Mobile Applications* that represents the interaction endpoint with the *UniSAS* end users that use the mobile devices.

Therefore, focusing on this aspect and the *Presentation Module* showed in the Figure 5.14, it won't be ported completely because only few modules are required to provide this layer features.

The Figure 5.15 presents the *Presentation Module* required components that have to be adopted so as to supply the *Mobile Applications* provided and supported by the *UniSAS* system *Mobile Applications Layer*.

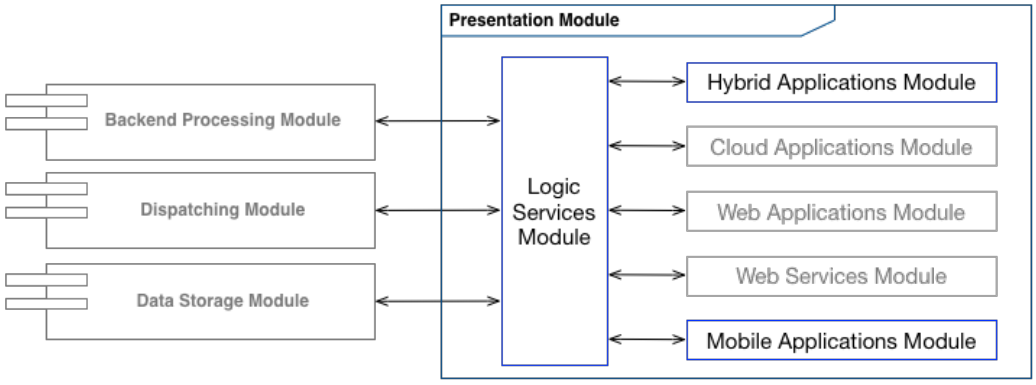

Figure 5.15: UniSAS Cloud - Mobile Applications - Presentation Module.

It has been decided to adopt only the essential components related to the *Presentation Module* because the complete porting could interfere with other layers and increase the system complexity.

Therefore, the essential components adoption would reduce the porting complexity, because there are fewer entities to manage.

The *Presentation Module* components that have to be adopted so as to support the *Mobile Applications* provided by this tier are the following ones:

- **Hybrid Applications Module**: to support the *Hybrid Application* that this layer has to provide and support (eg the application to manage and interact with the system through the Web components contained in the native application installed on the end users mobile devices).
- **Mobile Applications Module**: to support the *Native Mobile Application* that this layer has to support through specific services (eg it has to manage all the services related to the user authentication through the native app installed on the users mobile devices).
- **Logic Services Module**: to support the interactions between the target *Applications* and the other modules and components. It has to act as work flow and information broker so as to reduce the components complexity (eg the *Applications* have to interact with the *Backend Business Logic Modules* through this component).

Logically, the *Native Mobile Applications* can't be provided through a *Cloud* solution module but they can be supported and managed through specific services.

Thus, all the *Mobile Applications Module* features refer to *Web Services* dedicated to the mobile devices platforms (eg specific services basing on the mobile devices operating systems to improve performances and user experience).

Another aspect regards the *Hybrid Mobile Applications* because their diffusion is constantly growing because they allow to build cross-platform applications that have many common parts, reducing the required effort.

The common parts between the *Hybrid Applications* are *Web Applications* that can be rendered into specific sections of a native application by using specific Web components (eg a details Web page can be rendered into a native application section which contains a Web view).

Basing on these aspects, the *Hybrid Applications Module* has been ported to built the *UniSAS Mobile Applications*.

This strategy has made possible to implement these applications developing common parts provided by the *Hybrid Applications Module* coupled with other native parts supported by the *Mobile Applications Module*.

5.4.4 UniSAS Cloud AMBER Solution

The main objective of the porting process described in the section 5.4.3 was to define all the necessary steps that have to be done to port a Web based solution to the *Cloud* adopting the proposed architecture.

Thus, focusing on the obtained solution, the defined process can be adopted to port any Web base solution because the required steps can be adopted independently from the type and context of the target solution.

Basing on the porting process, the final *Cloud* solution obtained adopting the proposed architecture and the related work flows will be presented.

Finally, an evaluation of the *UniSAS Cloud Solution* will be presented to provide a comparison between the Web based solution and the *Cloud* one.

UniSAS Cloud Solution Architecture

This section will present the *UniSAS Cloud Solution* architecture obtained by execution of the porting process.

The Figure 5.16 presents the definition of the *UniSAS Cloud Solution*.

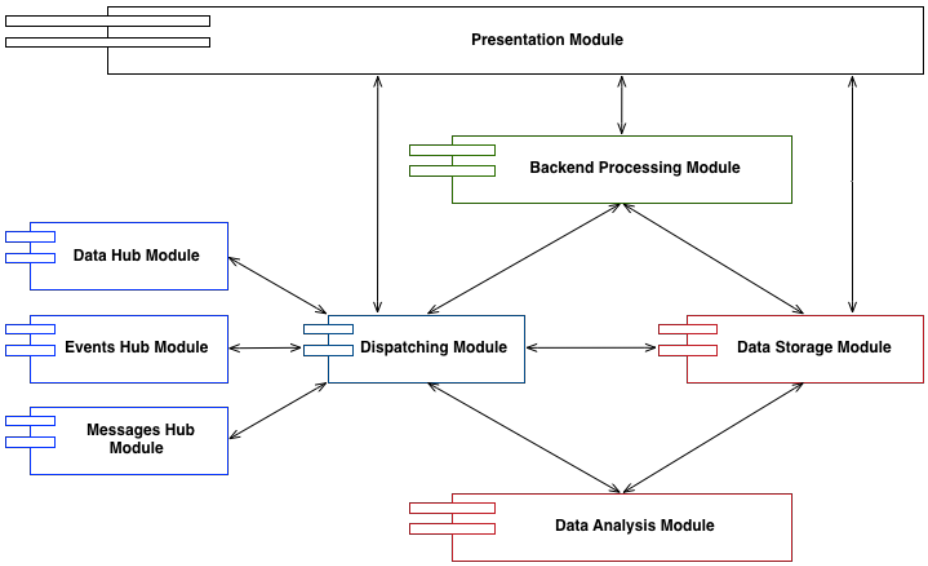

Figure 5.16: UniSAS Cloud Solution.

Figure 5.16 verifies that the proposed architecture can be adopted to port a Web based solution to the *Cloud* without reducing or limit solution features and capabilities as described in the section 5.4.3.

Furthermore, focusing on the *Solution Architecture Distribution* and the related aspects described in the section 5.2 (section focused on the *Case Study Solution Design*), the defined *UniSAS Cloud Solution* will be completed defining the modules distribution.

Thus, basing on this aspect, the Figure 5.16 shows how the *UniSAS Cloud Solution* modules can grouped by color so as to describe the distribution strategy. In fact, the module distribution has a key role in the *Cloud* solution obtained when the porting has been completed.

Finally, basing on the described porting process and the distribution strategy, it's possible to verify them through this case study. In fact, all the *UniSAS* features, characteristics and capabilities can be maintained adopting the proposed architecture and the related distribution strategy.

UniSAS Cloud Solution Evaluation

The evaluation about the *UniSAS Cloud Solution* will be provided so as to describe its properties, characteristics and capabilities.

Basing on the main scope of this analysis, this evaluation will be focused on the system critical aspects and limitations in order to correlate the system properties within its greats issues.

All the system main issues have previously introduced and described in the Web solution evaluation (see section 5.4.2).

Thus, they will be compared with the *Cloud* solution focusing on the system details and components so as to verify the proposed architecture advantages and benefits.

The main issues comparisons between Web solution and the *Cloud* one are the following ones:

1. **Scalability Issue**

   The *UniSAS Web Solution* has been designed and developed basing on the main scenario where a system instance has to manage the access structures of a single or a limited pool of buildings.

   This limitation has been defined so as to guarantee high system performances and reliability.

In fact, the huge communications amount could destabilize or compromise the Web bases solution capabilities and features.

The *UniSAS Cloud Solution*, instead, can support and manage an extended pool of buildings and access structures thanks to the properties and capabilities of the *Cloud* technologies and platforms.

Thus, the porting to the *Cloud* has improved the solution scalability because it has no issues and limitations related to this context. In fact, the scalability improving is one of the main advantages of the *Cloud* technologies adoption.

2. **Performances Issue**

   The *UniSAS Web Solution* limitation regarding the system managed access structures and buildings could be coupled within the performances issue, because the computational resources have to be commensurate with the target structures pools.

   Thus, Web solution scalability issues are also valid as concerns the performances, because these properties are strictly coupled. In fact, the huge communications amount could restrict and limit both of those properties.

   The *UniSAS Cloud Solution* scalability, instead, has been improved thanks to the properties and capabilities of the *Cloud* technologies and platforms.

   Thus, the porting to the *Cloud* has improved the solution performances, because it has no issues and limitations related to this context. In fact, the performances dynamic assignment and improving is one of the main advantages of the *Cloud* technologies adoption.

3. **Generated Data Issues**

   The *UniSAS Web Solution* limitations and issues regarding the generated data is due to the large amount of system generated data because this system has to manage an extended pool of access structures.

   Furthermore, the generated data will have different types, structures and formats because they are collected and generated by an heterogeneous ecosystem of devices and technologies.

The *UniSAS Cloud Solution* data management, instead, has been improved thanks to the capabilities of the *Cloud* technologies and platforms as concerns the data storage and related aspects.

Thus, the porting to the *Cloud* has improved the solution data management, because it has no issues and limitations related to this context.

In fact, the data store services and components improving are many of the main advantages of the *Cloud* technologies adoption (eg many data store managers and services such as *SQL Database* [**?**] can be used).

4. **Device Interaction Issues**

   The *UniSAS Web Solution* limitations and issues regarding the devices management is due to the extended pool of devices that have to interact and communicate with other system entities.

   Furthermore, the huge amount of communications between the system devices due to the needed interactions could limit the system overall performances.

   The *UniSAS Cloud Solution* device management, instead, has been improved thanks to the capabilities of the *Cloud* technologies and platforms as concerns the device communications, interactions and management.

   Thus, the porting to the *Cloud* has improved the solution device management, because it has no issues and limitations related to this context.

   In fact, the device communications, interaction and management improving are many of the main advantages of the *Cloud* technologies adoption (eg many device managers such as *IoT Hub* [**?**] can be used).

The described issues are strictly coupled with the *Cloud* so the porting process to this platforms type has improved the *UniSAS* overall quality.

Main quality improvements has been achieved thanks to the *Cloud* technologies adoption as concerns the main issues. In fact, all the main advantages and benefits will be presented so as to demonstrate and verify them.

Basing on the analysis related to the *UniSAS Web Solution* presented in the section 5.4.2, each introduced issue has been coupled to the main *Cloud* context.

In details, the main contexts related to the identified issues are:

- *Cloud*
- *Big Data*
- *Internet Of Things*

The *UniSAS Cloud Solution* can be referred to all these contexts thanks to the proposed architecture adoption. In fact, it's an architecture to design and build *Cloud* solution; furthermore, this architecture can be adopted also to design and build *Big Data* and *Internet Of Things* solutions.

All the details about these architecture objectives have been described and verified in the architecture modeling chapter (the chapter 2 that contains all details about the proposed architecture).

Finally, all the main issues related to the Web solution can be solved by porting it to the *Cloud*. Furthermore, the porting can be based on the proposed architecture adoption, in order to achieve its main advantages and benefits.

Chapter 6

Future Developments

This chapter will describe the main future developments of the proposed architecture with the aim of providing the most important development that will improve it. Furthermore, the described developments will enhance the proposed architecture in terms of features, capabilities and performances.

This chapter is crucial because it defines how the proposed architecture can be extended and improved to increase the quality and the support to designers and developers. Basing on this scope, all the possible future developments represent a relevant improvement of the proposed architecture.

Therefore, the development and building of the developments here proposed can support in a better way the diffusion and the usage of the presented architecture in real world solutions, because the requested effort will be reduced thanks to them.

Basing on these aspects and scopes, all the future developments will here described are strictly coupled with the building of the proposed architecture and the extensions and improvements related to its features and capabilities.

The main future developments related to the proposed architecture are the following ones:

- **AMBER Framework**: this development concerns the definition, design and build of a framework to support the development of solutions specific components, modules and libraries to provide capabilities and features related to the proposed architecture modules.

- **AMBER Testing Framework**: this development concerns the definition, design and build of a framework to support the testing related to solutions specific modules and components in order to simplify testing solutions based specific components, modules and libraries related to the proposed architecture modules.

- **AMBER Cloud Platform**: this development concerns the definition, design and build of a *Cloud Platform* based on the proposed architecture in terms of structure, behaviour, interactions, features and capabilities.

- **AMBER Hub & Registry**: this development concerns the definition, design and build of an *Hub* and a *Registry* to provide the proposed architecture built modules in order to distribute them as isolated and independent modules that can be hosted by any *Cloud* platform (eg with an approach like *Docker*[73]).

- **Amber DSL**: this development concerns the definition, design and build of a *Domain Specific Language* (shortened to *DSL*), which could defines a *meta-model* that provides the code generation feature based on this custom language.

The introduced future developments has been defined basing on many factors specific for *Cloud* technological context in terms of key features and main aspects.

Therefore, the focus has been particularly set on the simplification of the development of the proposed architecture modules and components by the designers and developers.

This allows them to design and develop solutions in an easier way, thanks to the target complexity reduction provided by these possible developments.

The following sections will define and describe the introduced futures development in order to provide an overview of them with maintaining an high level of abstraction so as to set the focus on their scopes, characteristics and properties.

6.1 AMBER Framework

This future development concerns the definition, design and build of a custom framework to support the development of components, modules and libraries to provide capabilities and features related to the proposed architecture modules.

This framework should be based on many factor of interest relating to the development of the proposed architecture modules specialized for the target solutions to build. In this scenario, all the architecture modules has been previously implemented and they are ready to be used in order to build the target solutions.

The proposed framework aims to provide the features, capabilities and services to allow solutions specific customizations in order to fit most to the target solutions requirements and needs. Therefore, this framework has to provide the basic and generic implementation of the proposed architecture modules and components.

In this scenario, the target framework can be progressively improved and customized basing on the target solutions requirements and needs. Following this guidelines, when an interesting and useful requirement is identified, its solution design and development can be abstracted and generalized in order to add it to the provided features and capabilities to this framework with the aim of improving it.

Furthermore, this framework has to be designed and developed focusing also on the provision of the tools and technologies useful to the development of the customizations related to the architecture modules and components.

Therefore, it has to be designed and developed allowing to interact and provide the required technologies and tools in order to allow their usage by the designers and developers that need to build custom solutions based on the proposed architecture.

At the end, this framework will simplify the design and development of the solutions specific custom modules, so it will improve the usage of the proposed architecture and will reduce the required effort and resources to build custom modules based on the custom solutions requirements and needs.

6.2 AMBER Testing Framework

This future development concerns the definition, design and build of a framework to support the testing related to solutions specific modules and components in order to simplify the testing of solutions specific components, modules and libraries based on the proposed architecture modules.

Therefore, this framework should be based on the main factor and principles relating to the testing of software systems in order to provide a solid framework to automate and simplify the test of the solutions based on the proposed architecture.

In this scenario, all the architecture modules has been previously implemented and they are ready to use in order to build the target solutions.

The types of testing this framework has to support and manage are the following ones:

- **Unit Testing**: this testing is the validation of the smallest part of systems and applications (eg the smallest part in procedural languages is a function). Since the focus of the unit test is very small, these tests can cover the entire code base. Furthermore, they need the least amount of setup and teardown. They are based on combinations of fewest assertions in order to improve their performances such as the *Unit Testing Framework* used in several programming languages (eg in *Java* the commonly used ones are *JUnit*[74] and *TestNG*[75]).

- **Component Testing**: this testing is the validation of the components of systems in order to verify and validate these components singularly (eg in a three-tire application, all the components related to the tires have to be verified and validated singularly). In fact, these components are composed by many smaller internal entities that has to be verified and validated by the *Unit Testing*. So, the systems components has to be verified and validated through the *Component Testing* in order to test them in complete isolation from the other components. In this type of testing, it's necessary to build mock models and simulators in order to simulate and provide input and data that has to be consumed in the isolated components testing, in order to validate the behaviours of the systems components in real world simulated scenarios.

- **Functional Testing**: this testing is the validation of the entire system in order to verify and validate the structure, interactions and behaviours of a system from the complete prospective related to the system functionalities and features (eg in case of application to report database data to a user, the features and capabilities to interact and retrieve data from the database have to be verified and validated considering the complete execution flow). These test are the most important and most complex because they have to cover all the systems execution flows in order to guarantee and test the reliability of the complete systems. Therefore, this type of tests are particularly slow and their required costs and effort are particularly high.

All the described types of test have to be supported by the considered framework in order to give them all the main standards characteristics and properties that provide an high reliability and efficiency to the tests.

The most important properties that have to be respected and considered in tests and related frameworks design and development are the following ones:

- **Keep tests short**: the tests have to be most short as possible in order to obtain a test suite for a solution where there are many tests extremely reduced in terms of length and complexity (eg in *Unit Testing*, a single test has to be a single assertion). In this scenario the complexity of the tests is reduced and the parts covered by a single test are extremely reduced and it's easier to manage them.

- **Keep tests independent**: in a solution test suite, no test should be dependent on another in order to keep them and their execution most isolated as possible. In this scenario, the result of a test can't influence the execution and result of the other tests. Therefore, the right solution is to ensure that each test is independent and validates one single step.

- **Tests should be idempotent**: if the tests can be executed over and over again and with the same results, they are considered idempotent. If the tests aren't designed to be idempotent, it's necessary to take an extended pool of measures to ensure they are executed just

once and then the test environment is rebuilt. This makes it cumbersome to reproduce failures while debugging and can lead to undesired inefficiency and inflexibility in the way tests are executed.

- **Tests should be deterministic**: having indeterminacy in tests undermines any test automation effort because no body trusts the results. Furthermore, if the tests are proved and verified as deterministic, whenever a feature or capability has to be called or required, it has to be tested in order to verify and validate its behaviour and interactions. This situation is highly instable and the tests performances are significantly and the results are unstable and untrusted. Therefore, all the tests has to be deterministic.

- **Minimize incidental test coverage**: the tests should cover the maximum quantity of code, features and capabilities in order to test, verify and validate the systems in the best way possible. For this reason, the test coverage must be dependent by the definition provided by the tests designers and developers because the incidental test coverage could influence the execution and results of the tests.

The introduced and described types of testing can't be completely automated because the effort and resources required to the testers are necessary to design and build complex tests.

Basing on this consideration, the target testing framework has to be designed and developed in order to provide tools and components to create the solutions required tests.

In fact, the tests design and development is extremely complex and for this reason it's impossible to completely automate them so the main scope of this framework is to support and simplify the testers' tasks as concerns the tests design and development.

All this framework components and helpers have to be designed and developed basing on the most important properties and characteristics defined by the main standards previously introduced and described in order to provide a stable and reliable testing framework.

Furthermore, this framework has to be designed and developed focusing on the interaction of with the main testing tools and framework, so as to provide all the components to support the design and development of the modules of the solutions based on the proposed architecture.

At the end, this framework will simplify all the types of testing to verify and validate the solutions specific custom modules, so it will improve the usage of the proposed architecture and will reduce the required effort and resources to test the custom modules and components of the specific solutions requirements and needs.

6.3 AMBER Cloud Platform

This future development concerns the definition, design and build of a *Cloud Platform* based on the proposed architecture in terms of structure, behaviour, interactions, features and capabilities.

This platform has to represent the base level on which the designers and developers will design, develop and manage the target solutions to build. In this scenario, all the required features and capabilities will be supported and managed by the considered *Cloud Platform* so, the target solutions features and capabilities can be designed and developed in an easier way because the low level complexity will be managed by this *Cloud Platform*.

Basing on these considerations, the development of the solutions to build will be easier then the traditional approaches and platforms because this particular *Cloud Platform* will provide specific management and support to the solutions designed and modeled basing on the proposed architecture.

Another fundamental aspect related to this future development is strictly coupled with the *Cross-Platform* development considering the other *Cloud Platform* provided by the main vendors (eg *Microsoft Azure* provided by *Microsoft*, *Amazon Web Services* provided by *Amazon* and *Google Cloud Platform* provided by *Google*). In fact, this *Cloud Platform* can be designed and developed in such a way to act as a wrapper between the architecture specific features and the services provided by the other platforms (eg the *Data Storage Module* can be wrapped in order to use the *Database Cloud Services* of the other platforms).

This *Cloud Platform* would not be only a wrapper but it has to extend the features and capabilities of the other platforms services in order to provide the specific features and capabilities of the proposed architecture modules.

At the end, this specific *Cloud Platform* can simplify the design and the development of the solutions to build basing on the proposed architecture. Furthermore, it could also improve and simplify all the solutions building processes, because it has to be fully compatible with the development and testing frameworks described in the previous sections (see sections 6.1 and 6.2) in order to reduce the required effort to build the architecture based solutions and improve the related building processes.

6.4 AMBER Hub & Registry

This future development concerns the definition, design and build of an *Hub* and a *Registry* to provide the proposed architecture built modules in order to distribute them as isolated and independent modules that can be hosted by any *Cloud* platform based on an approach and management similar to *Docker*[73].

This development is particularly interesting as concerns the main *Cloud* scenarios related to the modern interesting *Cloud* solutions and platforms. In fact, all the most used *Cloud* platforms provide services to host external systems by their engines and they have to contain, manage and execute the target external systems. The hosted and executed systems are provided as custom images and they can be executed in any engine (eg an image built to be executed using *Docker* can be executed on *Amazon Web Services*, *Google Cloud Platform* and *Azure* without any changes).

Focusing on this strategy, the future development treated in this section is particularly important because, following this way, it's possible to build the modules of the proposed architecture as images that can be executed in the target container engines of any *Cloud* platform which support and provide container engines.

This development has to be built basing on the container engine systems and all the related support and management components. In fact, the main strategy to realize this development starts from the building of the target images related to the proposed architecture modules and the components. This first step is quite simple because there are many systems that allows to build this images very fast in an easy way.

When the target images have been built, they have to be managed in order to provide them to the solutions that have to be supported by the proposed architecture.

Referencing to the main strategy to realize this development, this is the second step and it's extremely important because the management of the target modules and components images is one of the primary scopes of this development.

This *Hub* has to contain and manage the images to distribute in the target containers relating to the solutions based on the proposed architecture. This *Hub* can be distributed in order to improve its performances and can be designed and developed basing the most used hubs of the main *Container Engines* (eg the *Docker Hub*[76]).

The images contained and managed by the *Hub* have to be distributed to the platforms and systems in order to support the solutions based on the proposed architecture. Referencing to the main strategy to realize this development, this is the final step and it has a primary role because the distribution of the images can influence the performances of the entire system treated in this development.

The *Registry* has to distribute the images to distribute in the target containers relating to the solutions based on the proposed architecture. This *Registry* has to be distributed in order to improve its performances and can be designed and developed basing the most used registry of the main *Container Engines* (eg the *Docker Registry*[77]).

The *Hub* and *Registry* has to be designed and developed independently adopting the strategy related to the *Cross-Platform* focusing on the *Container Engines*.

In fact, both of these components have to be designed and developed so that they can interact with the architecture specific components and other platforms *Container Engines* components (eg the *Hub* has to be designed and developed so that it can interact with the *Docker Registry* and the architecture specific *Registry*).

Furthermore, the *Hub* and *Registry* have to be designed and developed in so that they have to be distributed in order to obtain components with high performances that can support and provide images to all the platforms and systems that require the treated and build images.

This future development can be built to support the future development related to the design and development of a custom *Cloud Platform* treated in the section 6.3. In fact, all the architecture modules can be developed as images that have to be executed and supported by the *Container Engines*. Following this strategy, these modules can be distributed and managed in an easier way.

At the end, these components will simplify the management, distribution and support of the components and modules to support, build and maintain the architecture based solutions. Therefore, it will improve the usage of the proposed architecture and will reduce the required effort and resources to develop, support and manage the custom modules and components of the specific solutions based on the proposed architecture.

6.5 AMBER DSL

This future development concerns the definition, design and build of a *Domain Specific Language* (shortened to *DSL*) which define a *meta-model* that provides the code generation feature based on this custom language.

This development is particularly important because the custom *DSL* to define can be used to simplify the tasks to do to the developers because it can make the low level implementation completely transparent as concern the technological and platforms specific features and tasks (eg it can automate the instance and the startup configurations of the *Cloud* services instances so that the developers can focus on the solutions scopes).

The *DSL* to has to be defined basing on the main scenario related to the *Cross-Platform* capability that it has to have so that it can be used to support the solution based on the proposed architecture that can be developed basing on the main *Cloud* platforms (eg *Microsoft Azure*, *Google Cloud Platform* or *Amazon Web Services*) or the specific *Cloud Platform* treated in the section 6.3.

A relevant benefit that can be provided using the specific *DSL* is the automatic code generation. In fact, basing on a *DSL*, it's possible to build a specific code generator that allows to automate the generation of specific platforms code starting from the *meta-model* written using the defined *DSL*, so designers and developers' tasks can be simplified and the focus can be set on the solutions scopes.

The code generation can be realized basing on many tools and libraries that can manage, validate and interpret the syntax and the semantics of the developers' produced code using the defined *DSL* (eg one of the most used frameworks to define a *DSL* and develop the related code generator is the *XText Framework*[78]).

At the end, the *DSL* to define and the related components to validate the code produced by the developers and generate the platforms specific code is particularly important, because it will simplify the development of the architecture based solutions. In fact, it will improve the usage of the proposed architecture and will reduce the required effort and resources to design and develop the custom modules and components of the specific solutions based on the proposed architecture.

Chapter 7

Conclusions

In this chapter will be presented the conclusions related to this thesis in order to provide a complete overview of the work that has been done as concern the proposed architecture.

The concept of *data* has a key role in modern scenario and this represents the main incentive for the constant growing up of the threated contexts.

In fact, in any context, company and business area, the value of this concept is growth up because it's possible to improve the qualitative and quantitative aspects related to the business.

The concept of *data* is strictly coupled with the *Internet Of Things* and *Big Data* concepts. In fact, these two concepts and related technologies are constantly growing up coupled with the *Cloud* and the *data* contexts.

Thus, the relation between these contexts will be considered as a milestone as concerns the solution design and building in order to focus on the solution future generations.

One of key aspects treated and analyzed in this thesis is the *Cloud* platforms architecture. The concept of *Architecture* has been used focusing on the nature of the *Cloud*, because it has been refined with the introduction of this context.

In fact, this term was used by considering the analyzed *Cloud* Platform services and components as the structural parts of the platforms themselves.

This choice was made because the *Architecture* of the software system represents the structural organization of the system, including its software components, the properties externally visible from each of them and the relationships between the parties [1].

Thus, in this case, it was decided to refer to the services as direct components of the platforms by abstracting from the individual internal ones.

On the basis of this platform services abstraction, it has been chosen to use the term *Architecture* as a shrinkage of *Cloud Service Architecture* [2].

Therefore, focusing on the treated main concepts, this thesis has proposed the definition of an architecture model that aims to support the design of different types of *Cloud* solutions to facilitate, simplify and improve modeling and design processes.

The proposed architecture is the result of an accurate study of the key issues with *Cloud* design and its related aspects of the necessary effort and resources, focusing on platforms and technologies related to real solutions and scenarios.

One of the goals of this model is transparency over technological aspects. For this reason, it has been described how this architecture model can act as a wrapper between the solution and the platform that hosts it, thus making the technology complex closely linked to the platform transparency.

Another aspect of particular interest is linked to the modules that make up the architecture, as they can be distributed and adopted on different *Cloud* platforms, without affecting their interactions, features and functionality, due to their independence with respect to the related aspects about the distribution and evolution. In fact, one of the key features of this architecture is being oriented towards multi-platform solutions, a key feature for *Cloud* solutions.

The proposed architecture aim also to be an helper to support the modeling and design of multi-contexts solutions. In fact, it aims to support the main *Cloud* solution types such as *Internet Of Things*, *Big Data* and *Cloud* generic applications.

On the basis of the main aspects defined about the target solutions, one of the key features of this architecture is being oriented towards multi-contexts solutions, another key feature for *Cloud* solutions.

The modeling and design processes related this kind of solutions is particularly complex and it requires an high effort due to their complexity and extended pool of coupled issues.

Furthermore, the architecture defined in this thesis allows to optimize the effort and resources needed for modeling and designing multi-platform *Cloud* solutions, since the complexity and the associated issues with this phase are greatly reduced in case of its adoption.

The *Cloud*, *Internet Of Things* and *Big Data* contexts are strictly coupled with many modern and innovative solutions contexts.

All these solutions have as fundamental the concept of *Smartness* associated with many environments and aspects which have as primary target the support and help to the people.

In fact, the *Smartness* has been referenced to main targets such as buildings, naming them as *Smart Buildings*, in order to improve the people living aspects related to the buildings (eg there are solutions to increase energy efficiency, reduce energy consumption so as to improve the environmental sustainability of the building).

The *Smartness* can also be related to the cities, in fact, the *Smart Cities* contexts is referenced to the solutions to improve the people living aspects related to the cities (eg there are solutions to improve navigation by high energy sustainability vehicles to reduce emissions of pollutants).

Another important context strictly coupled with the *Smart Buildings* and *Smart Cities* ones is the *Accessibility*.

Thus, this context has a primary role in modern solutions dedicated to the improvement of the quality related to the living aspects focusing on the special needs required by people whose have disabilities or physical impairments (eg people whose have to move with wheel chairs).

On the basis of the main concepts and issues related to the *Accessibility* contexts, the implemented case study which has been described and analyzed in the Chapter 6. The case study and its scopes have been selected focusing on the primary role of this context in modern solutions.

Therefore, *Cloud*, *Internet Of Things* and *Big Data* are the main concepts coupled with this thesis context because all of them have a key role in several modern solutions design and build processes.

On the basis of the provided case studies, the proposed architecture can be defined as feasible, effective and achievable, because all its modules and components can be implemented adopting any analyzed *Cloud* platforms.

In fact, the *Technological Evaluation* chapter (see Chapter 4) assess how the proposed architecture can be realized focusing on its features, capabilities and modules.

The feasibility is one of the most important aspects related to any solution in order to assess its improvements and capabilities, because one of this thesis scopes is to provide a feasible solution.

At the end, it's necessary to provide an overall evaluation focusing on many critical remarks. In fact, this architecture has many critical evaluations due to its technological contexts and scopes.

Being an architecture to model and design *Cloud* solutions, it could be necessary to analyze many platforms that could be adopted to implement and realize it.

In this thesis, only the main *Cloud* platforms have been considered in the analysis to assess if they can be used to realize the target architecture.

The selected platforms are proprietary and they can be analyzed basing on the official documentation (see Chapter 3), because the main articles and reports are focused on many open source platforms.

Thus, one important evolution could be the *Cloud* platforms analysis extension so as to include many other platforms. In fact, it could be interesting to include other proprietary *Cloud* platforms, such as *IBM BlueMix* [79], and open source ones, such as *OpenStack* [80].

Being a *Cloud* context architecture, the high speed changing of this context technologies could represents an issue, because it could cause problems relating to the solutions realization.

In case of solutions issues caused by platforms upgrading, this architecture could help thanks to its multi-platform property.

In fact, one of the main scopes of this thesis is to provide an architecture which could provide an high level abstraction to make the solutions based on it as independent from technological details referenced to the *Cloud* platform that host and distributed it.

Therefore, one of the architecture scopes is to provide an helper to avoid issues relating to the obsolescence of the *Cloud* contexts technologies.

Furthermore, in this thesis evaluation it has to be considered that its contexts dimensions are extremely dynamic and fast in evolution.

In fact, the future evolutions of *Cloud* contexts technologies could cause many new dimensions and issues due to new technical concepts, technologies and innovations.

Finally, the proposed architecture and all its modules and components have to be maintained, upgrade and improved so as to adapt it to the innovations.

Bibliography

[1] Wikipedia, "Software architecture." https://en.wikipedia.org/wiki/Software_architecture.

[2] Techopedia, "Cloud services architecture." https://www.techopedia.com/definition/26533/cloud-service-architecture.

[3] Gartner Inc. http://www.gartner.com.

[4] Dazeinfo, "Amazon aws vs google cloud platform vs microsoft azure: Which public cloud is best for you?." https://dazeinfo.com/2015/05/22/amazon-aws-google-cloud-microsoft-azure.

[5] Gartner, "Gartner cloud showdown: Amazon web services vs. microsoft azure." http://www.networkworld.com/article/2850114/cloud-computing/gartner-s-cloud-showdown-amazon-web-services-vs-microsoft-azure.html.

[6] M. Aldiab, "Public cloud war: Aws vs azure vs google." https://cloudacademy.com/blog/public-cloud-war-aws-vs-azure-vs-google.

[7] M. G. A. Donevski, S. Ristov, "Comparison of open source cloud platforms." 48th Int. Scientific Conf. on Information, Communication and Energy Systems and Technologies.

[8] A. M. R. B. H. M. J. Mahjoub, Meriam, "A comparative study of the current cloud computing technologies and offers." In Network Cloud Computing and Applications (NCCA), 2011 First International Symposium.

[9] G. K. Hoefer, Christina N., "Taxonomy of cloud computing services." In GLOBECOM Workshops (GC Wkshps).

[10] I. Nwobodo, "A comparison of cloud computing platforms." International Conference on Circuits and Systems.

[11] B. N.-L. S. R. Keith Jeff ery [ERCIM], "The future of cloud computing opportunities for european cloud computing beyond 2010." http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf.

[12] National Institute of Standards and Technology. https://www.nist.gov.

[13] ISO Organization, "Iso 7498-2:1989." http://www.iso.org/iso/catalogue_detail.htm?csnumber=14256.

[14] V. P. A. P. A. Botta, W. De Donato, "On the integration of cloud computing and internet of things," 2014.

[15] ITU, "Itu - committed to connecting the world." http://www.itu.int.

[16] S. M. M. R. C. Cecchinel, M. Jimenez, "An architecture to support the collection of big data in the internet of things," 2015.

[17] N. B. A. S. M. A. G. S. U. K. I. A. T. Hashem, I. Yaqoob, "The rise of big data on cloud computing: Review and open research issues," 2015.

[18] Wikipedia. https://en.wikipedia.org.

[19] X. Z. J. C. C. Liu, C. Yang, "External integrity verification for outsourced big data in cloud and iot: A big picture," 2015.

[20] Erik D. Brown, "What's the difference between business intelligence and big data?." http://ericbrown.com/whats-difference-business-intelligence-big-data.htm.

[21] NoSQL Database. http://nosql-database.org.

[22] Microsoft Corporation, "Windows 10 iot core."
`https://developer.microsoft.com/en-us/windows/iot`.

[23] Amazon, "Amazon aws iot."
`https://aws.amazon.com/en/iot-platform`.

[24] Google, "Android things."
`https://developer.android.com/things/index.html`.

[25] Microsoft Corporation, "Azure event hub."
`https://azure.microsoft.com/en-us/services/iot-hub`.

[26] Google, "Google cloud pub/sub."
`https://cloud.google.com/pubsub/docs`.

[27] MQTT.
`http://http://mqtt.org`.

[28] Microsoft Corporation, "Azure event hub."
`https://azure.microsoft.com/en-us/services/event-hubs`.

[29] Amazon, "Amazon simple queue service."
`https://aws.amazon.com/en/sqs`.

[30] Microsoft Corporation, "Azure service bus messaging."
`https://docs.microsoft.com/en-us/azure/service-bus-messaging`.

[31] Google, "Google cloud messaging."
`https://developers.google.com/cloud-messaging/gcm`.

[32] Microsoft Corporation, "Azure service bus."
`https://azure.microsoft.com/en-us/services/service-bus`.

[33] Amazon, "Amazon quicksight."
`https://quicksight.aws`.

[34] Amazon, "Amazon emr."
`https://aws.amazon.com/en/emr`.

[35] Amazon, "Amazon kinesis."
`https://aws.amazon.com/en/kinesis`.

[36] Microsoft Corporation, "Azure analysis services." https://azure.microsoft.com/en-us/services/analysis-services.

[37] Microsoft Corporation, "Azure hdinsight." https://azure.microsoft.com/en-us/services/hdinsight.

[38] Microsoft Corporation, "Azure stream analytics." https://azure.microsoft.com/en-us/services/stream-analytics.

[39] Google, "Google bigquery." https://cloud.google.com/bigquery.

[40] Google, "Cloud dataproc." https://cloud.google.com/dataproc.

[41] Google, "Cloud dataflow." https://cloud.google.com/dataflow.

[42] Amazon, "Amazon relational database service." https://aws.amazon.com/en/rds.

[43] Amazon, "Amazon dynamodb." https://aws.amazon.com/en/dynamodb.

[44] Amazon, "Amazon simple storage service." https://aws.amazon.com/en/s3.

[45] Microsoft Corporation, "Azure sql database." https://azure.microsoft.com/en-us/services/sql-database.

[46] Microsoft Corporation, "Azure documentdb." https://azure.microsoft.com/en-us/services/documentdb.

[47] Microsoft Corporation, "Azure blob storage." https://azure.microsoft.com/en-us/services/storage/blobs.

[48] Google, "Cloud sql." https://cloud.google.com/sql.

[49] Google, "Cloud bigtable."
`https://cloud.google.com/bigtable`.

[50] Google, "Cloud storage."
`https://cloud.google.com/storage`.

[51] Amazon, "Aws batch."
`https://aws.amazon.com/it/batch`.

[52] Microsoft Corporation, "Azure batch."
`https://azure.microsoft.com/en-us/services/batch`.

[53] Google, "Google compute engine."
`https://cloud.google.com/products/compute`.

[54] Amazon, "Amazon ec2 container service."
`https://aws.amazon.com/en/ecs`.

[55] Amazon, "Amazon appstream 2.0."
`https://aws.amazon.com/en/appstream2`.

[56] Amazon, "Aws step functions."
`https://aws.amazon.com/en/step-functions`.

[57] Microsoft Corporation, "Azure app service."
`https://azure.microsoft.com/en-us/services/app-service`.

[58] Microsoft Corporation, "Azure container service."
`https://azure.microsoft.com/en-us/services/container-service`.

[59] Microsoft Corporation, "Azure fabric service."
`https://azure.microsoft.com/en-us/services/service-fabric`.

[60] Google, "Google container engine."
`https://cloud.google.com/container-engine`.

[61] Google, "Google app engine."
`https://cloud.google.com/appengine`.

[62] Amazon, "Amazon api gateway."
`https://aws.amazon.com/en/api-gateway`.

[63] Amazon, "Aws mobile."
`https://aws.amazon.com/en/mobile`.

[64] Amazon, "Amazon cognito."
`https://aws.amazon.com/en/cognito`.

[65] Microsoft Corporation, "Azure cloud services."
`https://azure.microsoft.com/en-us/services/cloud-services`.

[66] Google, "Cloud endpoints."
`https://cloud.google.com/endpoints`.

[67] iBeacon Insider, "ibeacon."
`http://www.ibeacon.com`.

[68] R. Pi, "Raspberry pi."
`https://www.raspberrypi.org`.

[69] Google, "Android."
`https://www.android.com`.

[70] Apple, "ios."
`https://www.apple.com`.

[71] Wikipedia, "Multitier architecture."
`https://en.wikipedia.org/wiki/Multitier_architecture`.

[72] Wikipedia, "Model-view-controller pattern."
`https://en.wikipedia.org/wiki/Model-view-controller`.

[73] Docker.
`https://www.docker.com`.

[74] JUnit.
`http://junit.org/junit4`.

[75] TestNG.
`http://testng.org`.

[76] Docker Hub.
`https://hub.docker.com`.

[77] Docker Registry.
`https://docs.docker.com/registry`.

[78] X. Framework.
`https://en.wikipedia.org/wiki/Xtext`.

[79] IBM, "Ibm bluemix cloud." `https://www.ibm.com/cloud-computing/bluemix`.

[80] OpenStack, "Openstack cloud." `https://www.openstack.org`.

[81] Amazon, "Amazon web services."
`https://aws.amazon.com`.

[82] Amazon, "Amazon redshift."
`https://aws.amazon.com/en/redshift`.

[83] Microsoft Corporation, "Microsoft virtual accademy."
`https://www.mva.microsoft.com`.

[84] Google, "Google cloud platform."
`https://cloud.google.com`.

[85] Google, "Cloud datastore."
`https://cloud.google.com/datastore`.

[86] IBM.
`https://www.ibm.com`.

List of Figures