

POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Collaboration Interfacing Tools
via
Cloud Microservices



Relatori:

Politecnico

prof. MAURIZIO
MORISIO

Candidato

LUCA GIOMMONI

Intesa Sanpaolo

ALESSANDRO BRUNO

A.A. 2017 - 2018

Abstract

Nowadays almost all corporate applications integrate collaboration components that need to interface with the Enterprise Collaboration System through APIs. Concepts like *always on*, *high availability*, *resilience*, *automatic scalability* etc., have become part of the software programming paradigm in every company and with the advent of cloud computing, monolithic application architectures give way to new microservice architectures that, using container technology, realize the functions decoupling, enabling the continuous delivery / continuous integration paradigm of application development.

Introduction What is the *Enterprise Collaboration System* (ECS)? Why is it so important for companies today? This chapter explains the key role that the ECS plays in the business landscape and the challenges that are faced technologically and culturally during its introduction into the company. The ECS, born from the concept of Social Business, can be classified in the 4 internal zones of the *8C Model for Enterprise Information Management*, but there are characteristics that instead differentiate it a lot from *Social Media*. Finally, ECS is often associated with ERP, but there are at least four aspects in which the two systems differ: the application area, the content structure, the implementation processes and the intended use.

Main Problems The three major problems faced by the project are explained in this chapter. Until today, every application, that interfaces with the ECS, does so in a heterogeneous way compared to the rest of the existing applications, very often binding itself to APIs whose evolution is not under the control of Intesa Sanpaolo, but of the vendor of the product. This situation forces developers to rewrite – and often restructure – applications whenever the vendor forces to upgrade the version of an ECS component to a more recent one that implements new features and deprecates others. This makes the application ecosystem of the company rigid and difficult to manage and maintain. Finally, the security: the diversity of APIs and the long transition phases of an application from one version of the API component of collaboration to another always creates security gaps that are often difficult to manage, especially because vendor does not support the old version anymore.

Intesa Sanpaolo This chapter explains the actual ECS infrastructure of Intesa Sanpaolo and its Cloud and Microservices technology. The first topic is useful to understand the complexity, the numbers and

the issues met during this years. The second topic, instead, explains how it was possible to leverage the technology of Private Cloud to exploit the microservices infrastructure introduced in Intesa Sanpaolo via OpenShift platform. Some words on the adoption of OpenShift platform have also been spent. By creating new cloud service that set up the OpenShift microservices environment it was finally possible to realize the platform with all its components.

Collaboration Platform As A Service The main idea of the project is presented here. The set of APIs made available through the developed platform will replace all those currently used by the various existing applications, improving the overall management and eliminating the non-managed update component deriving from the proprietary API of the various products adopted by Intesa Sanpaolo. Functional and non-functional requirements for the first row version of the platform are also listed here, followed by the main features of the architecture of the platform, with layers organizational schema. Finally, two main use cases of extended platform functionalities are explained. The first one concerns the need of Alarm Center in Intesa Sanpaolo to communicate with specific *Organizational Unit (OU)* about received alarm: nowadays the list of users is extracted manually from a dedicated web page and each user is contacted until one available is found; the new platform, instead, provides specific APIs to retrieve available users belonging to a particular *OU*. The second use case was born to fulfill the GDPR requirement that allows customers to ask for all their data collected by the company. In this way, the platform provides APIs to store all collected documents about a customer into Sharepoint MySite of applicant employee.

Conclusions The platform is still in an incubation state and many aspects can be improved. Surely efficiency can be increased and many other use cases can be enabled. The microservice technology has given significant advantages in terms of management and scalability, making the service stable, highly reliable and with disaster recovery. In conclusion, this project allowed us to build an abstraction layer that standardize all collaboration APIs, giving the possibility to make new others and making homogeneity in various paradigms, from security to development ones.

Acknowledgement

I would first like to thank my thesis advisor Professor Maurizio Morisio of the *Collegio di Ingegneria Informatica, del Cinema e Meccatronica* at Politecnico di Torino. The Professor was always available whenever I ran into a trouble spot or had a question about my research or writing. He steered me in the right direction whenever he thought I needed it.

I would also like to acknowledge Alessandro Bruno, my manager in Intesa Sanpaolo, and Gregorio Scialpi, the project manager from which this dissertation was born. I am gratefully indebted to them for their very valuable comments on this thesis and their support. Furthermore, I thank the company, for which I still work, Intesa Sanpaolo, which allowed me to finish my studies and write a thesis on an internal project.

Finally, I must express my very profound gratitude to my parents and relatives and to my friends and colleagues for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Thank you.

Author
Luca Giommoni

Contents

1	Introduction	1
1.1	Enterprise Collaboration Systems	1
1.2	Customers And Employee Experience	10
1.3	Layout And Main Components	12
1.4	Cloud Computing And Microservices	16
1.4.1	Cloud	16
1.4.2	Microservices	20
2	Main Problems	22
2.1	Single APIs Ecosystem	23
2.2	Improvement Of Services And API Management	24
2.3	Improvement Of Security	25
3	Intesa Sanpaolo	27
3.1	Enterprise Collaboration	27
3.2	Private Cloud	33
3.2.1	Cloud Framework	34
3.2.2	Services	36
3.3	OpenShift	43
3.3.1	Platform Selection Process	43
3.3.2	Architecture	45
3.3.3	Delivery Process	53

4	Collaboration Platform As A Service	57
4.1	Main Idea	57
4.2	Requirements	58
4.2.1	Functionals Requirements	58
4.2.2	Non-Functionals Requirements	61
4.3	Architecture	63
4.4	Use Cases	66
4.4.1	Alarm Center	66
4.4.2	GDPR	68
5	Conclusions	70

List of Figures

1	8Cs Framework for Enterprise Information Management [23] .	4
2	People Oriented And Information Oriented Functionalities [22]	4
3	Social Media And Enterprise Collaboration Systems Share The Same Features [17]	7
4	The Radicati Group, Inc. — Forecast 2016-2020 [18]	11
5	The Radicati Group, Inc. — Forecast 2016-2020 [18]	11
6	Company uses collaboration <i>SaaS</i> from public cloud	13
7	Company exposes a single point of connection used by cus- tomers and contractors	13
8	Company uses federarion for contractors and a DMZ for cus- tomers communication channel	13
9	Cloud Lifecycle Forecast	16
10	<i>7 Different Types of Cloud Computing Structures – uniprint.net</i> [5]	18
11	Company vs. Cloud Provider Responsibility	18
12	Microservices Architecture – by <i>Microsoft Azure</i> [21]	21
13	The new platform decouples applications from products	26
14	Exchange version update in Intesa Sanpaolo	26
15	Categories of offered services in Intesa Sanpaolo	28
16	Peer to Peer Sessions in Intesa Sanpaolo	29
17	Volumes of ISP collaboration technologies	29
18	Videoconference architecture in Intesa Sanpaolo	31
19	Exchange architecture in Intesa Sanpaolo	31

20	Skype4Business architecture in Intesa Sanpaolo	32
21	Classification of Services	33
22	Private Cloud framework in Intesa Sanpaolo	40
23	Brands	40
24	Cloud Web Portal	41
25	Cloud Web Portal	41
26	Screenshot Of ServiceIdentificator/ServiceManifest DB table .	42
27	Final Results	44
28	OpenShift Architecture in Intesa Sanpaolo	44
29	Cloud + Openshift Provisioning	54
30	New Openshift Project	54
31	User Details	55
32	Further Optional Details	55
33	Tecnological Parameters	56
34	Platform Architecture	65
35	Frontend	65
36	Swagger Skype APIs	67
37	Alarm Center use case in Intesa Sanpaolo	67
38	GDPR use case in Intesa Sanpaolo	69

Listings

1	Service Manifest	38
2	Openshift Cloud Service Manifest	49
3	Openshift Cloud Service Manifest Instance	51

Glossary

acronym (acronimo) In Intesa Sanpaolo the term *acronym* or *acronimo* is used to indicate an application that belongs to an application subsystem. It is used mainly for application registry and usually is formed by four letters and a number. It is called *Acronym* because the first four letter represent the name of the application. For example the *Acronym* of the Cloud is CLOU0 . 53

C4 J2EE application that takes the name from Command & Control Communications Cyber/Computers american project. It was designed to lead the discovery, development and integration of military technologies for the air force, space missions and cyber security. It integrates an Ansible job scheduler and exposes REST APIs that enable environments configuration for front-end and back-end applications . 49

container Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems. — What is a Container[19] . 21

GDPR It is a new european regulation that regulates the processing by an individual, a company or an organisation of personal data relating to individuals in the EU. It replace the Data Protection Directive 95/46/ec and, with its 99 articles, makes more transparent the processing of personal data. . II

NFS Network File System (NFS) is a distributed file system protocol originally developed by Sun Microsystems in 1984,[1] allowing a user on a client computer to access files over a computer network much like local storage is accessed. NFS, like many other protocols, builds on the Open Network Computing Remote Procedure Call (ONC RPC) system. The NFS is an open standard defined in Request for Comments (RFC), allowing anyone to implement the protocol. — *Wikipedia, NFS* . 46

1 Introduction

1.1 Enterprise Collaboration Systems

Technology growth, together with globalization event, have changed completely the work environment, enabling teams to collaborate and communicate across time, geography and different organizations. Collaboration systems make people work efficiently, at reduced costs, avoiding physical shifts from one working place to another, leading to better results.

The Enterprise Collaboration Systems (ECSs) are information systems enabling and enhancing communication, coordination and collaboration amongs the members of teams and workgroups. They can be seen as socio-technical systems that include hardware and software as well as people, processes and organizational aspects[16]. More precisely, ECS is the set of strategies, tools, applications, and softwares that create the communication structure for interfacing with ecosystem of customer and for managing the internal exchanges of informations. This way, ECS make possible to carry out work over computer networks, building up virtual teams that reduce the need for people to be collocated. Several definitions have been formulated about virtual teams (last definition is the most widely accepted nowadays):

“A virtual team is a group of people who work interdependently with a shared purpose across space, time, and organization boundaries using technology. ”

— Lipnack and Stamps[9]

“Groups of geographically, organizationally and/or time dispersed workers brought together by information technologies to accomplish one or more organization tasks ”

— Powell et al. [12]

From all these definitions come clear that virtual teams rely on advanced information technologies to communicate: being a cross-functional information system, ECS responds to this needs because it helps individuals in companies to manage documents, share informations and knowledges with each other and work cooperatively, joining projects and assignments, enabling overall efficiency.

All these aspects of ECS indicate its inclination to be a social network. In particular it enables what has been defined as *Social Business*. The term derives from the concept of *Social Media* that identifies systems in which there are features that support interaction and interchange: people voluntarily spend their free time chatting, playing and above all exchanging documents and ideas. So again a system that allows people to connect with each other, making them interact with the concept of sharing. The first definition of *Social Business* may have been given by IBM, which states:

“A business that embraces networks of people to create business value. Social businesses embrace technology to enhance relationships between employees, customers, and partners. They augment business processes and applications with social interactions and insight. They provide integrated activities that use business data and social data. Social businesses more fully integrate the collective knowledge of people-centric networks to accelerate decision making, strengthen business processes, and increase innovation that matters. ”

— IBM, Redguides for Business Leaders[6]

Another definition for *Social Business* can be found in the article written by David Kiron, Doug Palmer, Anh Nguyen Phillips and Nina Kruschwitz:

“In our survey, we defined social business as activities that use social media, social software and social networks to enable more efficient, effective and mutually useful connections between people, information and assets. These connections can drive business decisions, actions and outcomes across the enterprise. ”

— Kiron et al. (2012)[7]

ECS, having been categorized also as a set of *Social Software*, can be classified into the four inner areas of the *8C Model for Enterprise Information Management* [22] – **Figure 2**. This framework is divided into two zones with four areas each: the inner core contains typical groupware functionalities and can be thought of as *people oriented* and *information oriented*:

- **Communication:** *PEOPLE EXCHANGING MESSAGES*

Softwares and tools, such as chat tools, email, blogging, audio / video conferencing, forums, etc., involving people exchanging messages with

other people, and includes functions capable to support different communication modes in terms of time – synchronous or asynchronous –, relationships – unicast, multicast or broadcast –, location – collocated versus remoted –, media – audio / video / test / ... – etc.

- **Collaboration And Cooperation:** *PEOPLE WORKING TOGETHER*

It concerns mutual engagement of people achieving common goals, thus all the devices that enable group cooperation such as sharing documents, workspaces and information in general, as well as user profiles and the possibility to express evaluations.

- **Combination:** *RE-USE OF DIGITAL CONTENTS*

This area is really important in supporting most of the collaborative technologies – emails, blogs, persistent chats, documents sharing, etc. ... – producing contents that need somehow to be managed. Thus, it deals with the management and reuse of digital informations produced by the other areas, encompassing all softwares, tools and methods that simplify, facilitate, improve findability and support aggregation, integration and reuse of digital contents.

- **Coordination:** *ORCHESTRATION OF PROCESSES, WORKFLOWS, EVENTS AND TASKS*

It focuses on functionalities to support orchestration of workflows, processes – highly structured, semi-structured and ad hoc processes –, events, tasks and management of the access to resources such as meeting room, documents, digital contents etc., and, finally, tools for supporting scheduling of meetings via shared calendars.

The outer ring of the 8C Model is focused on the management areas of the Enterprise Information Management. In fact there is a need, for instance, to integrate the tested technologies from the experimental phase into the existing system and processes in a viable and compliant manner with respect to companies and government regulations. The areas being part of the outer zone of the model are the following:

- **Content Management:** *MANAGEMENT OF DIGITAL CONTENT DURING ALL ITS EXISTENCE*

This area deals with digital contents management, from their creation to their expiration time. It creates determination about management

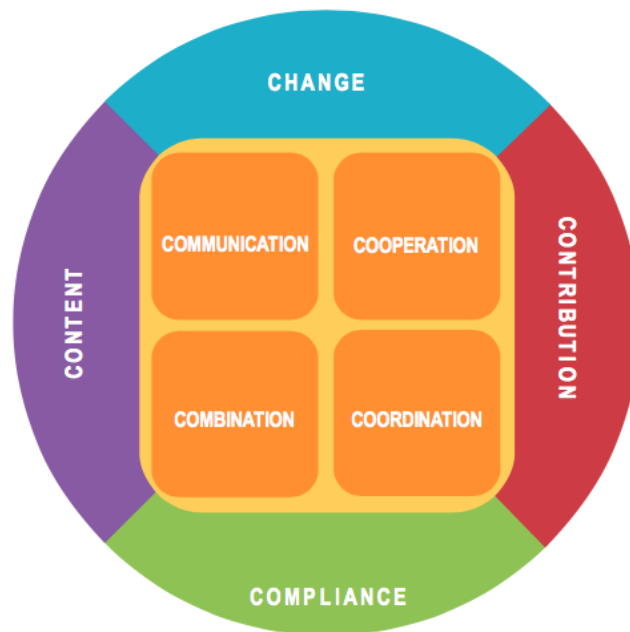


Figure 1: 8Cs Framework for Enterprise Information Management [23]

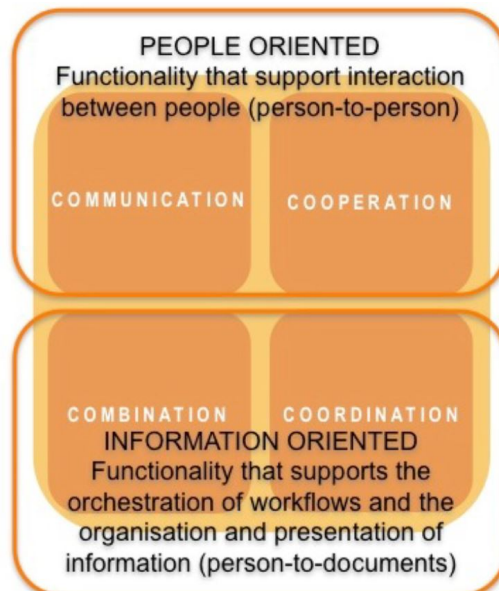


Figure 2: People Oriented And Information Oriented Functionalities [22]

and stewardship of informations generated by collaboration technologies during all their lifecycle. To achieve these goal, this area includes activities such as:

- Design of metadata for structuring documents;
- Implementation of storage;
- Retrieval and retention systems and policies;
- Rights management and monitoring of information findability.

- **Change:** *BUSINESS TRANSFORMATION AND PROCESS CHANGE MANAGEMENT*

The adoption of a new technology or a new collaboration paradigm almost always leads to a transformation of the business and the change of many business processes. For example, the introduction of new channels of communication to customers has radically changed the process of interaction with them. Or the availability of APIs offered by collaboration services has radically changed the way of writing software, more and more oriented to the user interaction, now made possible and easily implemented, with these services. This area deals with management of this kind of changement and with transformation of interactions and processes that happend into companies.

- **Contribution:** *COSTS AND BENEFITS*

This area deals with identification and measurement all costs and benefits achieved by companies from their investments in collaborative technologies; moreover it focuses on monitoring and management of achieved benefits over time.

- **Compliance:** *COMPLIANCE MANAGEMENT AND LEGISLATIVE ADEQUACY*

The introduction of new collaboration technologies and the adoption of new collaboration paradigms can strengthen and intensify the already numerous risks and add new ones. For example, most new risks are due to social media that can be used to monitor social profiles from unauthorized entities and to publish content that is not permitted or does not comply with the law. Therefore, this area becomes critical for the continuous search for the ever present threats and for the verification of the conformity of the contents with respect to the company norms and the government laws: constant monitoring is applied that guarantees contents and informations security.

The picture depicted in **Figure 1** shows the framework structure and comes clear as ECS can be collocated in the four central areas. Nevertheless, *Social Media* and Enterprise Collaboration Systems differ in many aspects, in particular:

- **Access:** Public *Social Media* are generally accessible by any Internet user, just a computer, tablet or smartphone and an internet connection to start using them. Furthermore, each interaction between different parties is decided and managed by the parties themselves - the service provider provides tools that allow the user to manage the interaction with any other user. In addition, the contents are accessible from any other platform according to the policies decided by the user. In companies, however, access to corporate social media takes place exclusively behind firewalls and content sharing is strictly controlled to ensure compliance with company rules and government regulations. Also the interaction between other users is strongly managed. from the company, allowing large freedom to the interaction between internal employees, but restricting the perimeter to external suppliers who can interact only after the federation of the systems.
- **Ownership:** Ownership of social profiles and content on public *Social Media* platforms is usually held by the service provider, based on the terms of use accepted by the user. This means that the user can manage the contents he produces – hide or modify, share and delete them – but this does not guarantee that, for example, the content deleted from the virtual space is also deleted on the service provider’s servers, which being the owner, can always keep a copy together with the chronology of the life cycle of the content, both for purposes of market studies or other internal activities or legislative obligations. This situation does not fit the policies of companies that instead need to be the owners of all the content produced within. So they cannot adopt public solutions they do not have control of, but must use products that allow full management of digital content and user profiles.

We can better understand what ECS and *Social Media* have in common looking at **Figure 3**: both *Social Media* platforms and Enterprise Collaboration Systems rely on *Social Softwares* which enable the typical features of the inner areas of *8C Model*.

Enterprise Collaboration Systems, sometimes, can be associated to Enterprise Resource Planning (ERP) systems too, but there are some substan-

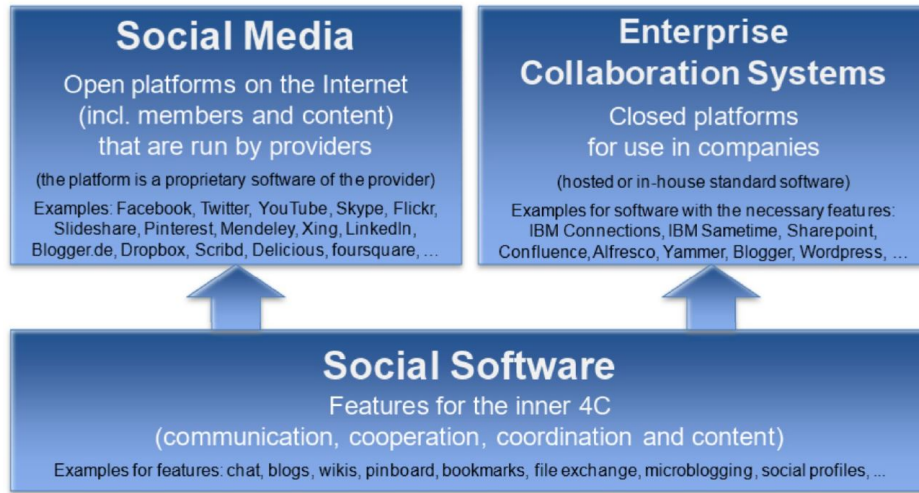


Figure 3: Social Media And Enterprise Collaboration Systems Share The Same Features [17]

tial differences between them. Indeed they differ from the at least four aspects [16]:

- *Application area*: ERP systems are based on a process-oriented view [2] with the aim of supporting clearly defined and repeatable business functions following built-in business rules [16]. Both systems are critical for companies, but ERP systems are more crucial to businesses because of supporting the core order fulfilment process, while ECS systems are designed to enable and support joint work among people in the workplace. They are supportive in nature and their continuous availability is usually less business critical than in the case of ERP systems [8].
- *Content structure*: ERPs data comprises highly structured master and transactional data that reflect resources of the company and business activities, while ECSs contain, for the most part, unstructured content such as documents, blogs or posts [16].
- *Implementation process*: The selection and implementation of ERP systems must follow a well-defined project plan [11], [1] whilst ECS are often reported to follow a “bottom up” [3] and rather experimental [13] introduction approach. As a consequence, ERP implementation projects are about understanding business processes and finding ways

to ideally support them, whilst ECS implementation projects are about identifying use cases and collaboration scenarios that best suit a specific company and the people working in it. By understanding the potential of the ECS, companies can create a better and more efficient digital workplace for their employees [16].

- *Purpose and use:* ERP systems are not designed to give room for creativity, indeed they impose their structure and their implemented order of events onto the user; their use is mandatory for activities in the order fulfilment process. ECSs, on the other side, are tools for ad-hoc use which offer choice and thus entail uncertainty [4]. Both system types require skills for their use, however, ERP skills are much more routine. ECS require the user to understand the suitability of a tool for a current task at hand and to make appropriate selections. ECS use is often voluntary so that the user has to acknowledge the benefits of using the tool. This is why “user acceptance” has traditionally played an important role in research on collaboration systems [14].

The communications and informations exchange are nowadays the foundation onto which companies establish their business. Their capabilities to build strong communication network between employees and customers ecosystem can determines the health of their investments. Moreover, the life perspective of companies is driven by the ability to create and manage communications between all its internal parts, ensuring a quick exchange of information that can largely determine the success or failure of projects. With the birth of ECS the culture of companies has changed completely because the idea of collaboration has gone from simple synchronous or asynchronous communication to user interaction on multiple levels, incorporating, as mentioned above, various aspects deriving from social media. So today it is essential for companies to keep this network alive and feed it as much as possible with new technologies because, by doing so, they will contribute to building a common and shared corporate conscience, making proactive actions more and more possible and feasible, which safeguard investments. Moreover, these new social aspects that have become part of the Enterprise Collaboration Systems consistently reinforce the sense of belonging that employees feels towards the company, making them much more involved and protagonists of company life. Especially in large companies these tools allow all employees to feel part of a single great team that works to achieve common goals, breaking the culture of the past that tended to circumscribe the feeling of a team to the next managerial reality.

So, overall, ECS is changing the way employees work together and, for this reason, they are attracting more and more attention with a consequent increase in investments by companies in this sector. Heinz and Kumar, in the IBM conference in Orlando, said: [17]

“The ESN (Enterprise Social Network, a/n) will be the backbone of future organizations – and thus a prerequisite for business operations ”

-- Heinz and Kumar, IBM Connect, Orlando, Feb 1, 2016

1.2 Customers And Employee Experience

The Enterprise Collaboration System has significantly improved the experience of company employees, but above all the customer experience. New communication channels are now available and new solutions that enable the social aspects of interpersonal interaction have been adopted together with new information management paradigms.

When the emails were introduced there was a real revolution because the information circulated much more quickly and, consequently, the corrective actions and the investments could be managed with greater speed. In addition, the customers had another channel available to communicate directly with the company, in a more fluid and fast way, avoiding costs and delays deriving from the classic correspondence. Over time, however, the amount of information has grown exponentially and with it, the amount of emails sent and received, going almost to saturate this channel, damaging, sometimes, the user experience during communication and exchange of information. Figures 4 and 5 show a forecast made by The Radicati Group, Inc. in which is evident the estimated growth, from year 2016 to year 2020, of the email user account and the quantity of sent and received emails. The chart shows that the number of email users increases by about a hundred million each year: it starts from more than two-thousand-six-hundred millions in 2016 and it will reach three thousand millions in 2020. In addition, the amount of emails sent and received will come to about two-hundred-sixty billions, starting from two-hundred-fifteen billions in 2016. It is therefore possible to notice a strong expansion of this communication channel until, in some situations, it arrives at a real abuse. The experience of customers and employees therefore does not always grow with the same growth coefficient as new collaboration technologies. In order to keep this ecosystem of experience constant, companies need to restructure the collaboration paradigm in order to preserve its integrity. Forrester defines the [15] **customer experience ecosystem** as:

“The web of relations among all aspects of a company — including its customers, employees, partners, and operating environment — that determine the quality of the customer experience.”

The *web of relations* in the Forrester quotes can be created, maintained and increased only with the correct management of the Enterprise Collaboration System, adopting strategies that aim to make the customer feel part of the

ecosystem of the product, creating a more intimate contact with it. It is no coincidence that nowadays almost every company decides to create its presence on the various social platforms and increasingly makes available both customers and employees effective and efficient collaboration channels. So customers can provide their feedbacks that goes directly into the product development cycle and, therefore, usable in a very short time.

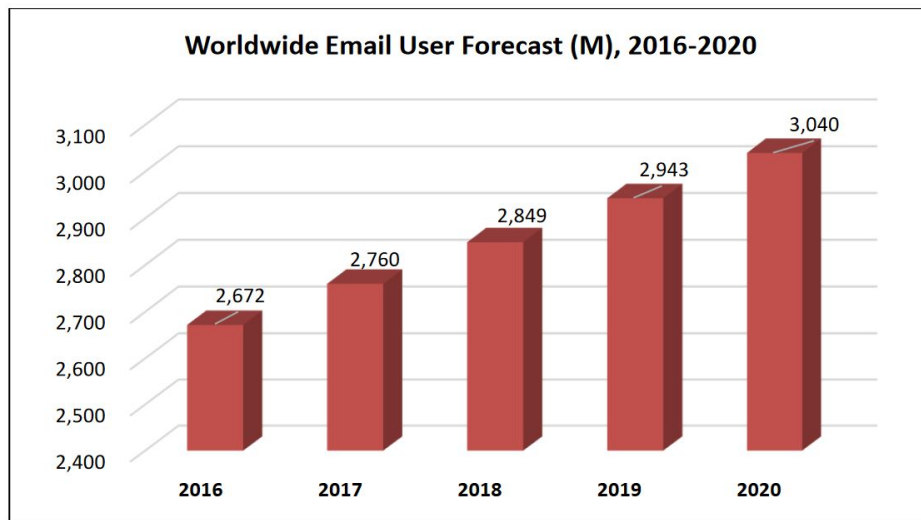


Figure 4: The Radicati Group, Inc. — Forecast 2016-2020 [18]

Daily Email Traffic	2016	2017	2018	2019	2020
Total Worldwide Emails Sent/Received Per Day (B)	215.3	225.3	235.6	246.5	257.7
% Growth		4.6%	4.6%	4.6%	4.5%

Figure 5: The Radicati Group, Inc. — Forecast 2016-2020 [18]

1.3 Layout And Main Components

To support collaboration, social business, communication, content and knowledge sharing within organizations, ECS seeks to combine social software components, such as social profiles, with traditional groupware components (e-mail, for example, shared agenda, shared workspace for documents). The main needs that the ECS usually faces are telephony, instant messaging, e-mail, file exchange, the ability to work simultaneously on shared documents, video conferencing and meeting rooms. All the components listed above could be declined very differently, depending on size, business, work modalities and territorial dispositions of companies. The layout of the ECS can therefore vary a lot. Small enterprises, for example, with few employees and without particular legal restrictions, can use a single application for internal and external communications, another application for the exchange of files and another for e-mails. The listed applications could all be used through Cloud *SaaS* (Software As A Service). The layout in this case is very simple and does not require any dedicated particular infrastructure, with the exception of some licenses. The services offered by Google, such as Hangouts, Gmail and Drive could represent a realization of the layout described above – **figure 6**. On the other hand, the most complex and medium / high-sized companies could have different needs. If they process sensitive data or generate a lot of traffic, a hybrid or totally in-house solution could fit better – **Figures 7 and 8**. This type of layout requires that companies have a private or managed server farm, in which to dedicate the infrastructure to manage the various components of ECS. In this way you can have full control of traffic and content. The investments in this mode are of CapEx (*Capital Expense*, a/n) type, i.e. the company supports a cost for the purchase and maintenance of the technology. With public Cloud solutions, instead, there are investments of type OpEx (*Operational Expense*, a/n) in which the cost is directly proportional to the use of technology (you pay the allocation and time of use of a resource). However, medium / large companies may also need direct interfacing technologies with the user, such as conference rooms, adding this type of asset to investments and potentially increasing the complexity of the ECS layout.

Figure 6 shows an example of a solution that totally adopts the public Cloud: the company buys enterprise licences and accesses the services offered by the Cloud provider via the Internet. This is the simplest layout because it does not require any server farms being managed by the company. In addition, product support and scalability is totally managed by

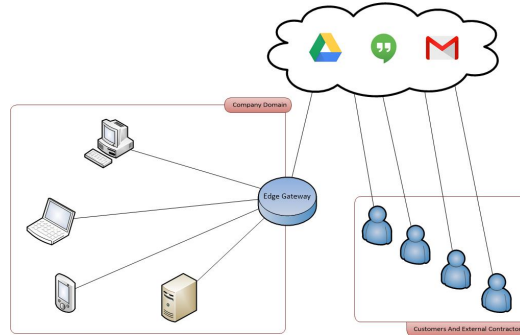


Figure 6: Company uses collaboration *SaaS* from public cloud

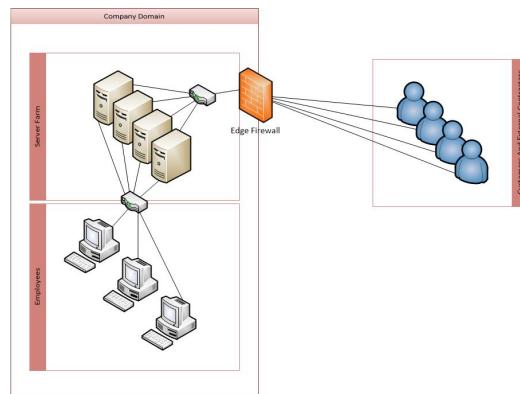


Figure 7: Company exposes a single point of connection used by customers and contractors

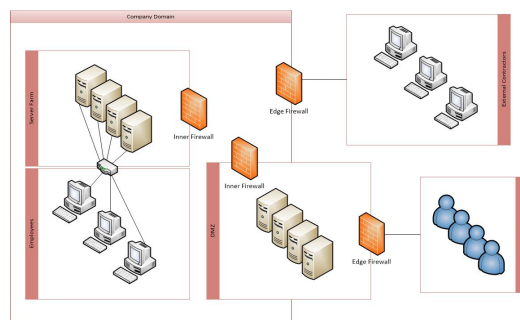


Figure 8: Company uses federation for contractors and a DMZ for customers communication channel

the cloud provider. The company that adopts this solution will only have operating costs that will scale proportionally to the growth of its business: when intense work requires more resources, these are paid and allocated on demand, then disposed when no longer necessary. This paradigm translates the business needs into perfectly adjusted company costs. Obviously, as mentioned previously, all this freedom has the price of no longer having full control of infrastructures.

A more complex example of the layout of an Enterprise Collaboration System is shown in *Figure 7*. Here the public component no longer exists because everything is managed on premises. In this case, the company manages the infrastructure for collaboration applications, exposing a single point of contact to the outside for establishing communications and interactions with external suppliers and customers. In practice it is similar to what happened with the public cloud, but reported in the private domain of the company. This layout has the advantage of using the same applications for both internal and external communications, but exposes some security issues – there is no distinction between an employee, a supplier or any other user on the network, as is the case with telephone conversations – that companies of considerable size and importance can not afford.

Figure 8 represents a layout suitable for companies of great importance and size as it fills many gaps in the preceding layout. In fact, in this model we can distinguish three domains:

- The internal domain for the communications between the employees
- The domain for communications with the suppliers and the external federates
- The domain for the communication channels to the customers

The latter is usually found on a secure network, much more controlled and with many reduced functionalities. With this layout it is possible to use enterprise applications for communications and collaboration within the company, while customers are increasingly looking to use the applications they already own, such as Whatsapp and Telegram, to make communication more comfortable – the customer uses an app that he already knows and does not have the burden of downloading others.

In conclusion, there are many layouts that can be adopted by the companies: according to their needs they can choose a solution totally cloud, in

which all the management and support of technology is left to the service provider and the cost is proportional to the use, hybrid solutions or totally in place. Companies will choose the right solution based on their business and the legislative constraints they must comply with.

1.4 Cloud Computing And Microservices

In the *section 1.3* we talked about Cloud solutions for ECS layouts. To better understand these concepts, we see here what Cloud computing is and how it is done.

1.4.1 Cloud

“Cloud computing is the delivery of computing services – servers, storage, databases, networking, software, analytics, and more – over the Internet (“The Cloud”). ”

– What is cloud computing? – Microsoft Azure [20]

From the quotation found on the Microsoft Azure website, we understand that Cloud Computing is a paradigm for IT resources delivery, such as virtual servers, storage, runtime platforms, applications and services, etc. It was born in 2010, with a lot of skepticism, so it started its real development in 2015 and finally the forecasts place its maturity between 2020 and 2025, as shown in **Figure 9**. This new paradigm wants to replace the precedent that

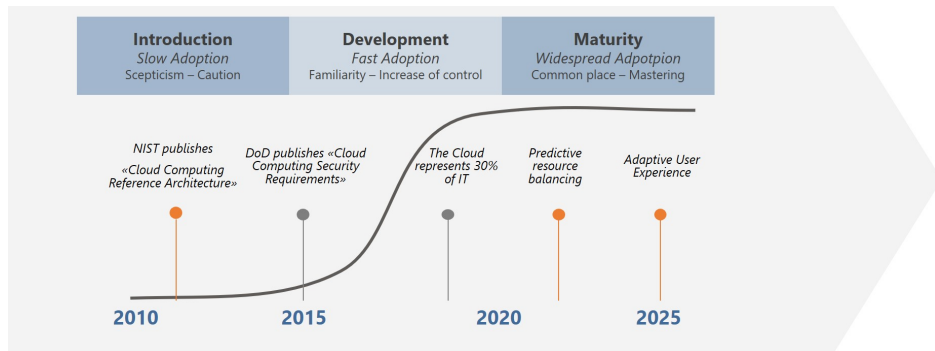


Figure 9: Cloud Lifecycle Forecast

forced companies to build their own server farm and fill it with all the types of technological infrastructures they needed. Before the advent of Cloud Computing, as a new delivery paradigm, whenever the need arose to create a new application or adopt a new technology to support a new product, the company had to take on the task of buying, installing, maintaining and support all the necessary technological infrastructure. Delivery times and time-to-market were much longer and investments were even more risky. In

fact, if the company wanted to test a new product or software, it was forced to add all the necessary technology to its server farm, without the guarantee that this type of long-term investment would lead to an economic return. Today, with the various Cloud providers on the market, you can have all the IT services you need, even for experimentation, without the need for long-term investments, paying only for what you use, for the time when you use it. This paradigm obviously allows to scale or reduce the infrastructure as needed, perfectly adapting the costs to the business needs. So Cloud Computing brings many benefits to the company that decides to adopt it [20]:

- **Cost:** you pay what you use
- **Speed:** vast amounts of computing resources can be provisioned in minutes, typically with just a few mouse clicks
- **Global Scale:** ability to scale when its needed, and from the right geographic location
- **Productivity:** no waste of time in installation and configuration
- **Performance:** the Cloud provider is responsible for periodically updating all the technology infrastructure with the most modern technology available
- **Reliability:** data backup, disaster recovery, and business continuity easier and less expensive

Cloud computing offers three main types of services, as shown in **Figure 10**:

- **IaaS – Infrastructure-as-a-Service:**
computing infrastructure like virtual machines, storage, networking, provisioned and managed over the Internet
- **PaaS – Platform-as-a-Service:**
on-demand environment for developing, testing, delivering and managing software applications (e.g. Oracle WebLogic or RedHat JBoss as application server)
- **SaaS – Software-as-a-Service:**
Software applications provisioned and used over the Internet, usually asking for subscription (e.g. Google Gmail, Microsoft Office 365, etc.)

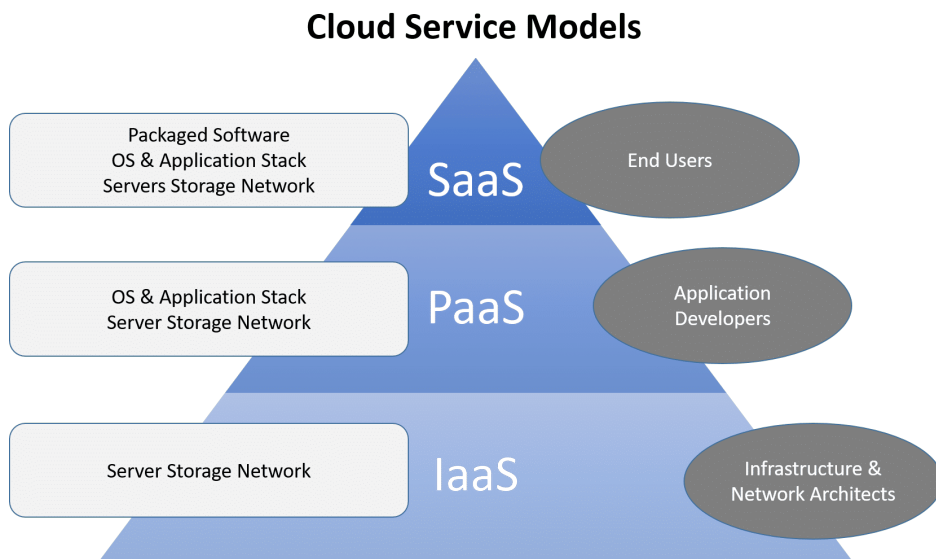


Figure 10: 7 Different Types of Cloud Computing Structures – *uniprint.net* [5]

According to this classification of services, the responsibility of the user, and therefore of the company, changes in the management of the infrastructure. **Figure 11** shows clearly how the responsibility of the company, regarding completely in-house solutions, regards the entire infrastructure stack, starting from the base with the different technological components (network, storage, etc.) up to the application. In this case the company is responsible for

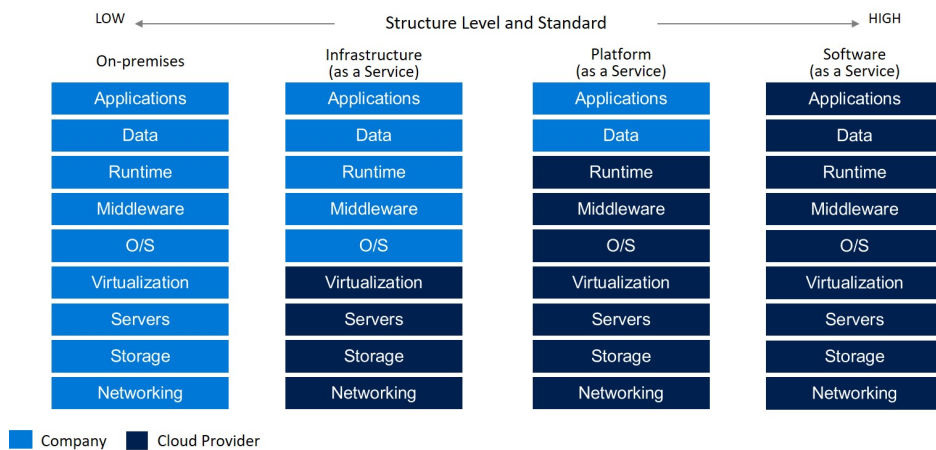


Figure 11: Company vs. Cloud Provider Responsibility

the technological renewal, the support of all the parts and the updating and patching of the application. As soon as you move into Cloud solutions you immediately notice a reduction in responsibility assumed by the company. For IaaS services, the Cloud provider is responsible for all the low-level technological components, while the company remains the medium-high part of the infrastructure stack that starts from the operating system. With *platform-as-a-service* the responsibility of the company is still reduced, concerning only the management of data and application software, while in the *SaaS* solutions the entire infrastructure stack is managed by the Cloud provider and the company only has to worry about how to use the application. Finally, the Cloud be declined in:

- **Private:** The company owns its private datacenter, or pays a third party hosting it, and implements the Cloud paradigm with orchestrators and infrastructure software. In this case the responsibility of the company encompasses the whole infrastructure stack. Many companies opt for this declination because they already have a datacenter or because they are obliged by law or internal statute to have full control of the digital content produced.
- **Public:** All IT resources are provided by a Cloud provider outside the company. In this case the model of responsibility falls into the one shown by the last three stacks on the right in *Picture 11*. This case is very common to small or startup companies that can not yet afford a datacenter or simply does not suit them. With this declination, a startup can be immediately up to speed by minimizing the weight of information technology on time-to-market. Obviously this type of solution requires a careful choice of the Cloud provider, trying to standardize the terms of use offered with the requirements and legal obligations.
- **Hybrid:** The hybrid solution is the most chosen by older and larger companies as it combines the advantages of external management with the control of digital content guaranteed by the possession of an internal data center. For large companies it is not easy, neither physically, nor culturally, to switch from one technological solution to another, so companies that are born with the old delivery paradigm are slowly coming to the new one first building their own private Cloud and then combining it with the public one with platforms (the so-called *IPaaS – Integration-Platforma-as-a-Service*) that make it possible to share data between the two types of Cloud.

1.4.2 Microservices

Until a few years ago, before the advent of Cloud Computing, the technology was static and applications were designed to fit statically on users' devices and on-premise corporate servers. To give applications more flexibility and possibility to scale, the 3 tier architecture was adopted: presentation, business logic and data. But it was still a monolithic structure, released on pre-scaled hardware to withstand estimated load peaks. When the maximum load limit was reached, IT scale-up was performed to avoid reconfiguring other servers and rewriting the application's architecture. Because of this, developers created application capabilities highly coupled with each other, even in different tiers. This meant that a change, even small, to a single functionality forced the redeploy of the entire application on all the tiers and any update could lead to unforeseen situations with consequent application downtime. Nowadays applications are no more performed locally, neither on the user's devices nor on pre-set on-premise hardware, but are distributed on different technological infrastructures, even in different geographical places. Furthermore, the large number of Cloud service providers on the market has lowered the entry barrier for IT service competitors, making concepts like *maintenance windows* or *update downtime* give way to *always on*, *high availability* and *service continuity* ones. The microservice architecture was created to meet all these challenges, trying first to eliminate the coupling between the various application features. The name "microservices", in fact, refers precisely to the decomposition of the application into small independent functional units that has its own API, which is typically in REST format, with which it can interact with the other microservices (12); it can also be updated independently from the rest of the microservices that are part of the application. Unlike monolithic applications, for which the developer declares requirements directly to IT staff, microservices applications communicate their resource requirements to orchestration software – also called *cluster manager* – which assigns them to the VM^1 of the most suitable cluster to respect the terms of efficiency and optimization of resources. This enables on-demand scale-out, using resources only when needed and eliminating any single-point-of-failure. Updates are also managed in a much simpler and more consistent way, with the possibility of updating even a single function, testing it and, in case of problems, restore the initial situation by making a simple rollback. This decoupled system enables *CD / CI paradigm*², allowing

¹Virtual Machines: virtual servers that emulate physical ones and are created upon a cluster of physical servers

²Continuous Delivery and Continuous Integration

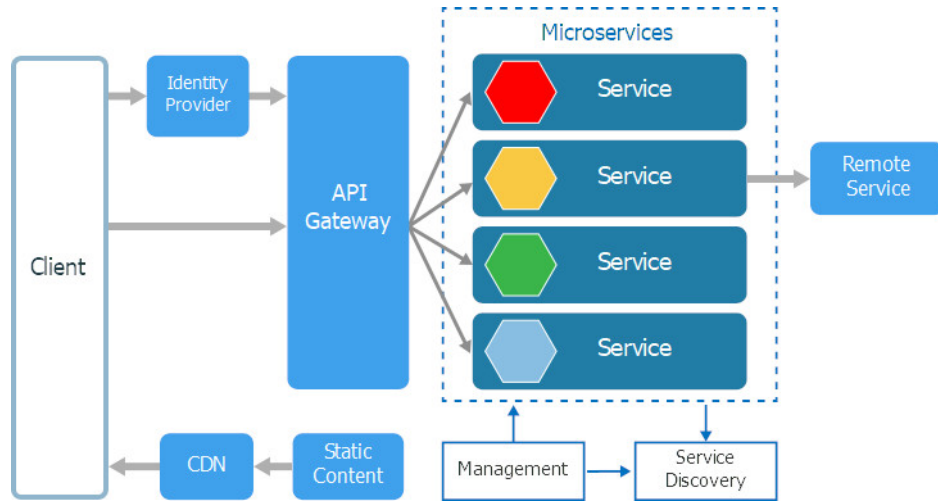


Figure 12: Microservices Architecture – by *Microsoft Azure* [21]

developers to frequently evolve the application.

Containers technology represents the most common way to implement a microservices architecture and, in order to take full advantage of the paradigm, an orchestration platform is required, which automatically manages the life cycle of each container: in case of load peak, the platform automatically deploys new containers; if a virtual server fails, it moves the respective containers to another server without interruption. Finally, it can also manage geolocality, deploying the containers in the datacenter closest to the user (if the Cloud provider allows it) so as to always guarantee the best performance.

2 Main Problems

Nowadays *Virtual Teams* rely on sophisticated technologies to communicate – see *section 1.1* – that can be synchronous, like chat or audio / video conference, or asynchronous like email and fax. Many times it is hard for these products to communicate with each other because of the different technology with which they were developed or the different platforms in which they are performed. This situation makes the IT environment chaotic, costly and counterproductive, with consequences like repeated miscommunication, difficulties to locate data, etc. This also applies to IT staff striving to monitor and control collaboration tools[10]. Hence, a lot of today’s collaboration technology initiatives do not succeed in expressing their transformational potential because of the absence of unified collaboration strategy. Furthermore, every time a major update affects those applications, developers shall rebuild softwares that use old APIs: this job generally happens very slowly, drawing up a lot of transition periods that overlap, increasing collaboration ecosystem complexity and security issues. So the introduction of an Enterprise Collaboration System within an already consolidated company, as Intesa Sanpaolo is, brings many challenges and hindrances. In fact, a number of critical situations have emerged along with the benefits of collaboration products, making a fully home-managed solution a necessary remedy for them. It is not always easy to identify interaction problems with the collaboration ecosystem. Many critical issues are indeed often ascribed to *telecommunications*, that form the backbone of the ECS, or to other supporting technologies, such as server infrastructure, storage, etc. Moreover, once the real cause has been found, the solution could not be straightforward. In Intesa Sanpaolo the problems presented by the various groups of application developers were collected, therefore the design of the platform was elaborated to expose the collaboration services required as a solution that best fits the infrastructural, application and management needs, together with the technological ones and economic.

The three main problems addressed are described in the following sub-sections:

- In *sub-section 2.1* it is explained how the lack of a single ecosystem of APIs has led to a fragmentation in the management of applications that interface with multiple collaboration products; in fact each product has its APIs that evolve independently from those of other products: this inevitably leads to a continuous restructuring of the software.

- In *sub-section 2.2* expands the previous topic to explain how this allows not only to have a single entrypoint for the APIs of collaboration used by the applications, but also to improve the management of them, creating additional services and decoupling the functionality from the product.
- In *sub-section 2.3* the topic of *security* is taken up again, explaining the pitfalls that often hide in the various products that make up the ECS; it is therefore explained what remedy has been chosen.

2.1 Single APIs Ecosystem

One of the main problems that often arise in Intesa Sanpaolo derives from the large number of products present in the company and the heterogeneity of the interfaces exposed by them. In this way, there is no single point of contact (*SPOC*) but as many as the services exposed by each product. This situation makes the management and maintenance of applications using those interfaces very difficult and inflexible, forcing developers to update – and sometimes to totally restructure – the software every time the vendor releases a new version of the product, removing the support to the previous ones. This complexity is fully realized specially in the case of one application interfacing with multiple products, each of which has different life cycle and version. Furthermore, it becomes really hard to write quality software in an environment with several formats for exchanged data and with a large amount of different APIs available. The new platform meets the standardization requirements in the exposure and use of collaboration services, filling the gap between the *SPOC* required by applications and the multitude of APIs available. Whenever there will be a need to interface with the ECS, the platform will provide a single access point through uniform and managed APIs whose life cycles and development will follow the natural physiology of the company. Developers will no longer be forced to restructure and / or rewrite applications every time a collaboration product version changes as they will not be aware of this any more: the platform, assuming the role of middleware, will hide this complexity, guaranteeing continuity of service and stability of communication. The advantages brought by this new platform are not only increased stability and security for the consumer applications, but also the enhancement given to developers who can now implement additional features instead of wasting time continually updating existing ones in order to adapt them to ECS changes. The ecosystem of the new APIs, in fact, is not

a simple broker that wants to mask the complexity of ECS, but a real new level of abstraction that controls, filters and balances the communications between the world of applications and that of collaborations. This improves overall safety and stability. Another advantage derives from the possibility of sharing code among different projects thanks to the unique interface provided by the platform for the use collaboration services.

2.2 Improvement Of Services And API Management

Lots of collaboration problems in Intesa Sanpaolo come from product migrations and transition periods. As shown in *Figure 14*, every time a new version of Exchange is released, it takes at least a year for the migration to finish. But once the migration is completed, many servers still remain running with the old product version to support applications that use the related APIs. For instance, the last servers with Exchange 2007 have been deactivated in 2016 – after three generations have passed! – and many servers that support the 2010 version are still active. In general, there are still running servers that support three versions of Exchange because of some old applications – probably developed with monolithic structure – using the APIs of those versions that are difficult to update. It clearly requires the need for an abstraction layer that finally decouples the business functionalities of the collaboration products from the product APIs, strongly tied to the version. The same goes for Sharepoint and Skype and in general for all the services that are part of the Intesa Sanpaolo ECS. Obviously it is not just the fault of the products and version updates, but also of the architecture of the applications that does not allow a flexible update. Unfortunately, Intesa Sanpaolo, being a great reality and not a software house, but a bank, is not always able to keep up with all the new software technologies and architectures and having a very large application ecosystem, it also has large inertia towards change. This does not mean that applications are not updated, but that they do it more slowly than products they interface with. This is why the new platform also assumes the fundamental role of temporal shock absorber, filling the gap between the update times of external products and internal applications. Furthermore, new APIs, that form the abstraction layer, will maintain, during update, all old functionalities that are stable and used by old applications, adding the new ones in a low impact mode for the entire ecosystem. In this way no more restructuring of software will be necessary, creating a lot of stability among applications environment. In conclusion, the new ecosystem that will group together all the available APIs related to

the several collaboration products will enable a centralized management and maintenance of the exposed services, decoupling the development of applications from the deep knowledge of products and their lifecycle.

2.3 Improvement Of Security

The last major problem faced with this project concerns computer security. The evolution of a product to a new version – and related change of its APIs – makes, indeed, hard the maintenance of the security level for many old applications which are probably designed highly tied to the API present at the time and for those with special requirements. This situation happens specially when time constraint is very small and the update must be released quickly. Moreover, each collaboration infrastructure (Skype, Exchange, etc.) has its own security degree and paradigm that contributes to make the management and maintenance of overall security fragmented. The new developed platform try to responds to the issue creating a unique layer of exposure of collaboration services that has a uniformed security paradigm, fully compliant with company's regulations. So developers and users can exploit these APIs without take care of security compliant because the platform provides the same for all services. Furthermore, the architecture of the microservices (*see section 3.3*) on which the platform is built provides another layer of security: with two high availability sites, the Openshift architecture in Intesa Sanpaolo guarantees resilience, always on, continuous delivery / continuous integration without downtime due to any kind of failure. Finally, since the platform does not expose all the features made available by the product, this also limits the damages that may be caused by an illegal, inappropriate or accidental use of those API. Moreover, building and managing internally an intermediate layer, as the new platform is, it enhances the suitability to the company – and governments when it needs – terms, since the product APIs not always are fully compliant with these and this situation is often detected too late, when some developer decides to use those APIs.

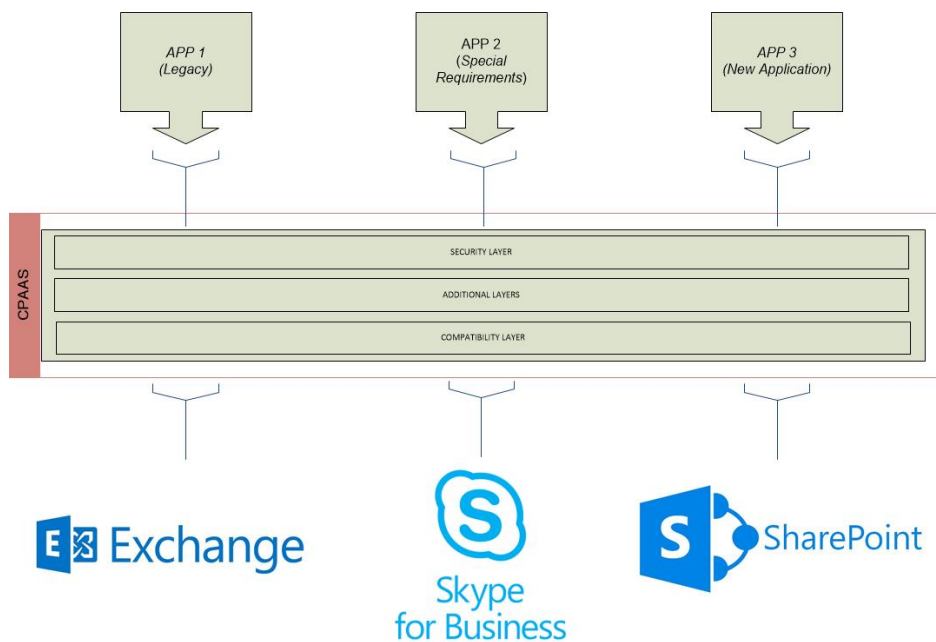


Figure 13: The new platform decouples applications from products

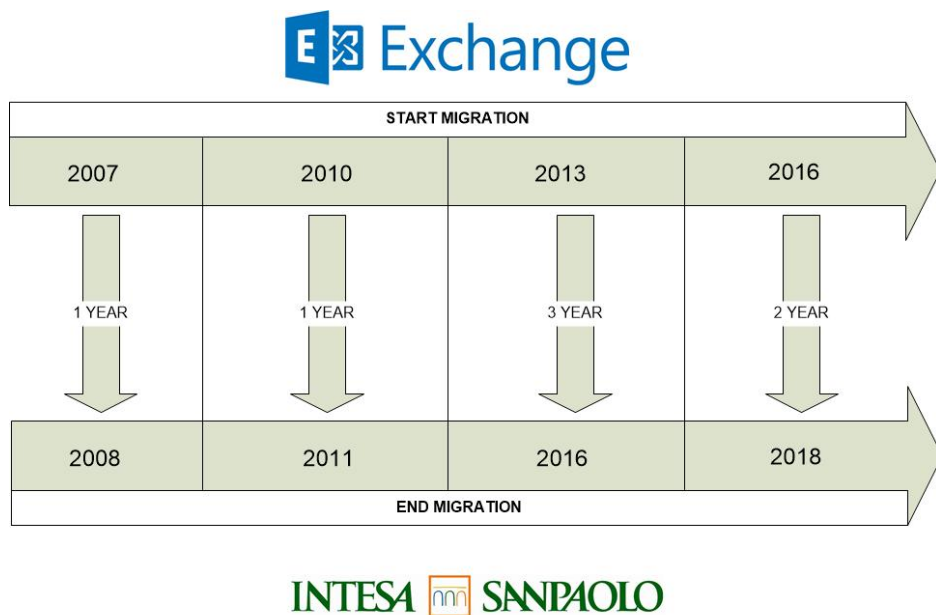


Figure 14: Exchange version update in Intesa Sanpaolo

3 Intesa Sanpaolo

3.1 Enterprise Collaboration

Over the years, the banking group of Intesa Sanpaolo has grown increasingly, becoming a very large company, with a strong presence even beyond the Italian national border. For this reason, the collaboration has had to grow and develop to cope with the exponential growth of users. The ECS has therefore become a great complex system that incorporates all the necessary features that allow to provide a continuous service even in case of accidents. Security, high availability and disaster recovery are the cornerstones on which all of its infrastructure is based.

Collaboration system in Intesa Sanpaolo follows three main principles:

- *Consistent and Improved User Experience* – access on any device, anywhere, and at any time.
- *Highly Availability and Disaster Recovery* – designed to keep the number of unplanned and/or extended outages low, it enables patching and upgrading activities without violating service-level agreements. Additionally, two pools in different geographical areas provide disaster recovery: if an entire pool goes down, the backup one continues to provide service.
- *Centrality* – the system is deployed in a centralized manner within three data centres located in Moncalieri – Settimo (Turin) and Parma.

The Collaboration Bureau in Intesa Sanpaolo Group deals with providing services to all the companies belonging to the group, having branches in every part of the globe. The services offered can be categorized into four main stacks, as shown in **Figure 15**. At the bottom of each stack there is a service belonging to standard package that includes Skype for Business, Exchange, Outlook, SharePoint Mysite and Teamsite. In addition, optional services are offered such as video conferencing, SMS and Fax Server, certified e-mail, File Sharing and Weshare.

Skype for business (S4B) Formerly Lync, it is the application used today by all employees of the Bank group for activities like instant messaging, audio

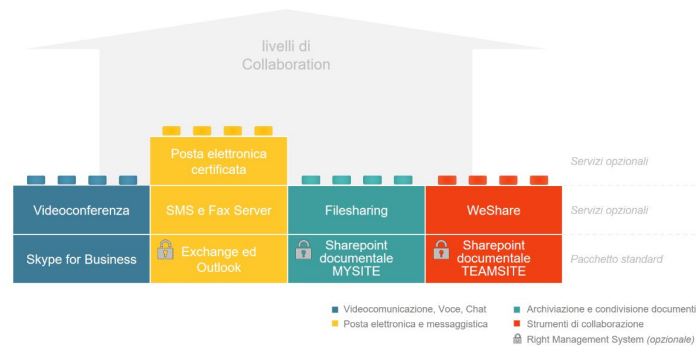


Figure 15: Categories of offered services in Intesa Sanpaolo

and video conferencing, management of switchboards, sharing of desktops and applications and online meetings. The service is totally on-premises and relies on a three-pole infrastructure. Today it serves 72,000 users with an average of 500,000 p2p sessions (**Figure 16**), 9,000 audio calls and 3,000 conferences per day. This kind of services allows employees to always be reachable on their client regardless of their physical location. S4B integrates completely with traditional telephony making sure that users are reachable also from any telephone, internal and external. This integration is today a service offered to a portion of users; in particular to the colleagues of the "new concept" bank branches, in which they chose not to install any traditional telephone, and in the central offices.

Video Conference It gives the possibility to perform point-to-point video-communication (1500 sessions per month), multipoint (2400 sessions per month), LiveCast, recording sessions on request. The infrastructure is fully integrated with Skype for Business giving the possibility also to connect users from traditional telephone devices.

Microsoft Exchange It provides a way to centralize e-mail messaging, calendars, notes, task lists and address books management. Microsoft Outlook, the most used client to connect to an Exchange server, is the company's official one together with its Web interface: Outlook Web Access (OWA). It also offers additional features such as ten years e-mail archiving, the ability to archive e-mail directly online, integration with fax and SMS systems, integrated chat in webmail, autonomous management of permissions and integration with Sharepoint. Furthermore, one of the most important feature comes from the close integration with the Active Directory system, which

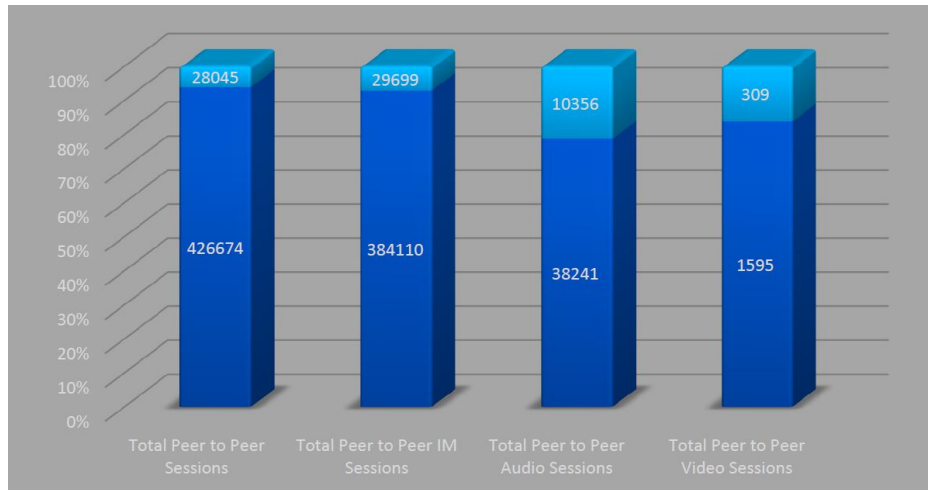


Figure 16: Peer to Peer Sessions in Intesa Sanpaolo

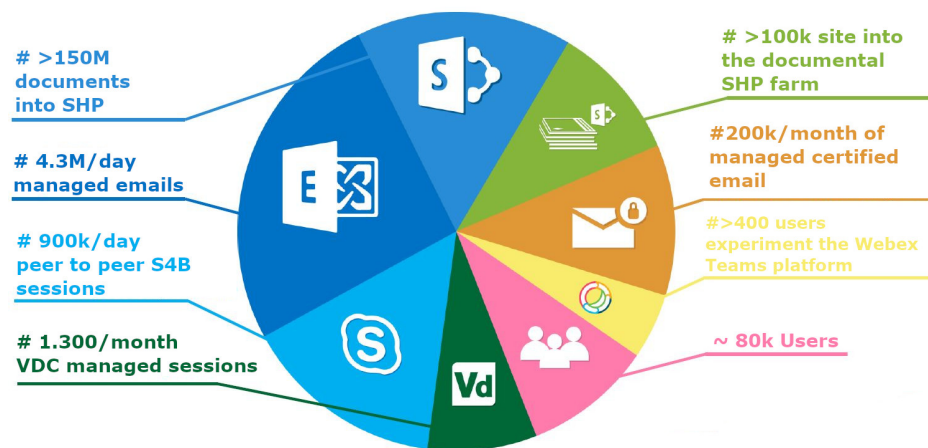


Figure 17: Volumes of ISP collaboration technologies

facilitates the management of users and distribution groups implemented at the level of Organizational Units (OU).

Right Management System Enabling encryption of Office suite documents – word, excel, power point and e-mail – this tool also provides integration with mobile devices using the VPN and the use of pre-set protection templates: not printing, not forwarding, reserved, not responding to all, etc. An additional feature of particular relevance is the application of the product for the Sharepoint suite.

The Intesa Sanpaolo group offers to employees the certified e-mail service, managing today around 850 accounts, with future extension to all branches of the Bank. Through this service, the e-mail messages have legal validity, equivalent to a registered letter with a return receipt. Moreover, the possibility to send telegrams, create and manage virtual PEC accounts, track access and mail flows and digital signature are offered. Additional features include the possibility of using a standard archiving service through MailDOCpro.

Sharepoint It is divided into three sub-services: Mysite, Teamsite and WeShare. Mysite and Teamsite respectively allow the creation of personal document areas, currently about 72,000, and group document areas, about 13,000, whose documents can be shared even outside, offering data backup, versioning, authorization management in autonomy and automatic provisioning of user profiles. Other interesting features are the recycle bin through which documents can be recovered autonomously up to 15 days after the elimination and co-editing through the Office Web App that allows you to edit a file simultaneously. WeShare, with 80 working groups currently active, offers easier projects and work groups management, integrating more solutions offered by different market products including discussion groups, task management, document sharing. The showcase section makes possible to illustrate the project or the objectives of the group externally through descriptions or FAQ section.

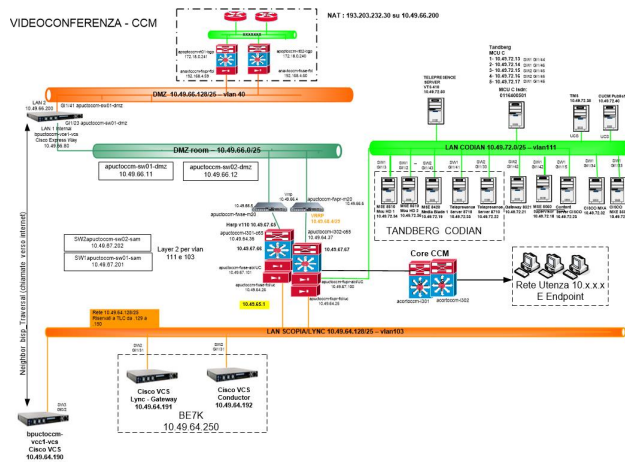


Figure 18: Videoconference architecture in Intesa Sanpaolo

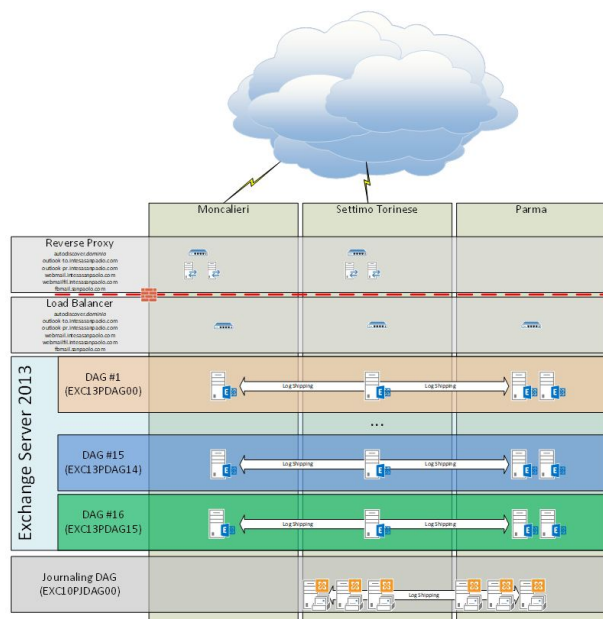


Figure 19: Exchange architecture in Intesa Sanpaolo

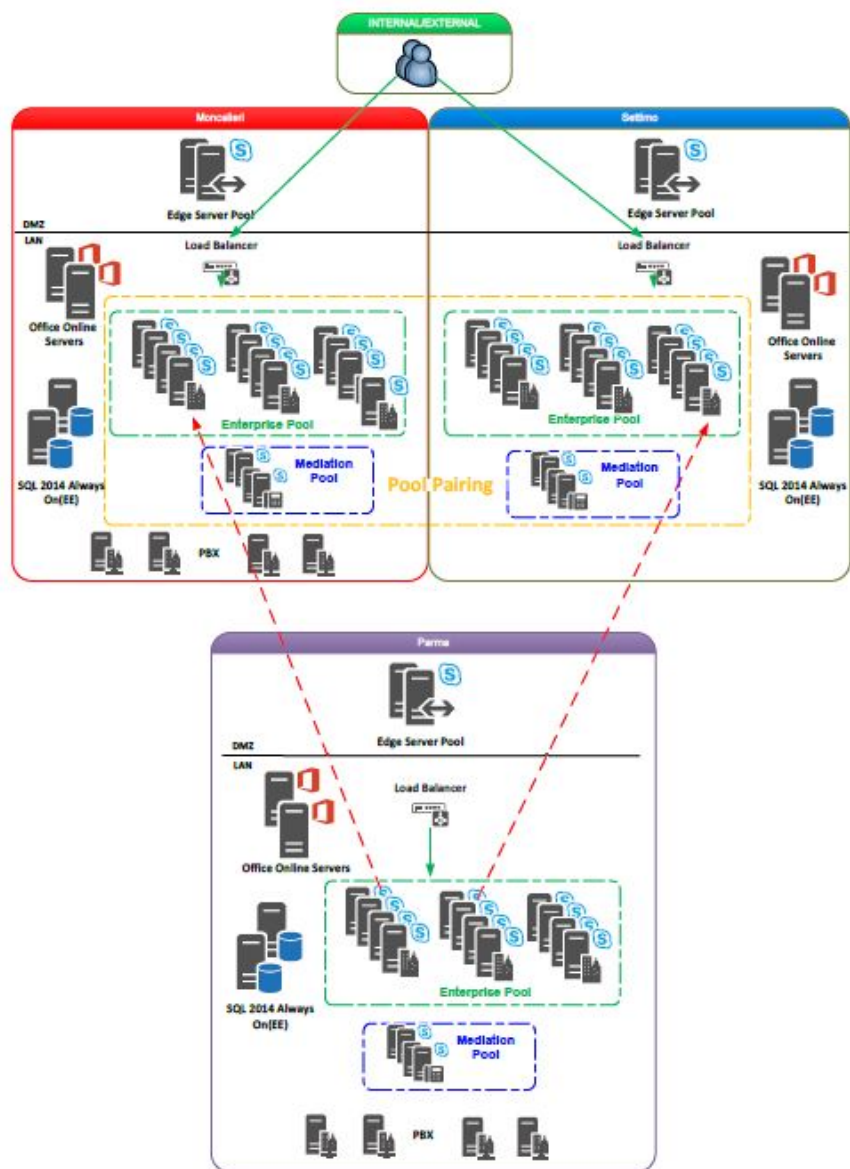


Figure 20: Skype4Business architecture in Intesa Sanpaolo

3.2 Private Cloud

In 2011, the paradigm of *Private Cloud* was adopted in Intesa Sanpaolo as a means of technology infrastructures delivery. Today, after almost 8 years, around 80% of the departmental infrastructure is managed using Cloud technologies. The delivery speed has increased and has been made also standard. Before Cloud, indeed, servers, storage, network and all other IT resources and runtime environments, were strongly customized for the needs of applications. Now, as the Cloud offers a set of pre-defined services with which to obtain IT resources, developers need to adapt in the same way they would if they were using public cloud service providers.

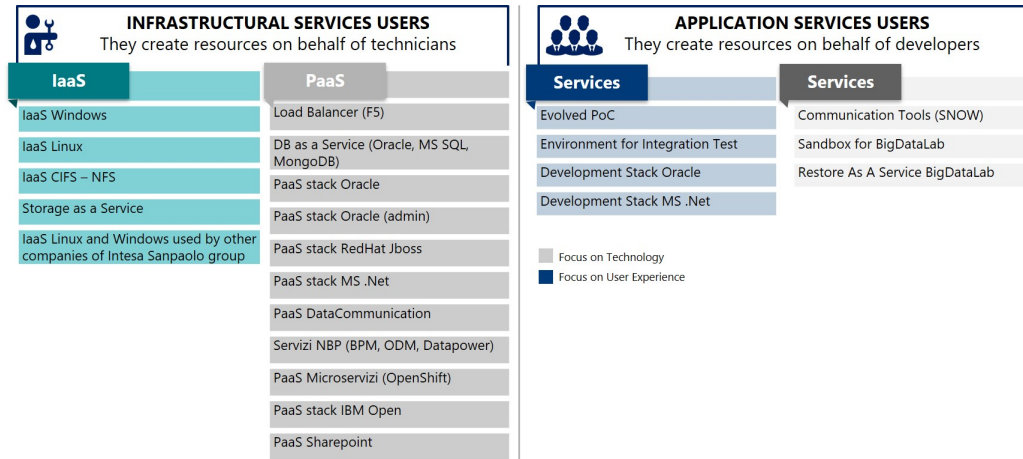


Figure 21: Classification of Services

However, the use of the private Cloud is limited to the group of IT experts in which there is an office, called Project Coordination, which interfaces with application developers. This office translates the application needs according to the technology present in the company and then provides it to the developers using the Cloud. Developers can only use services that deliver resources in PoC (Proof of Concept) environment (**Figure 21**). This is because they are usually unable to understand IT needs of applications, also because the DevOps³ culture was not adopted at the same time as the private Cloud

³A software engineering culture and practice that aims at unifying software development (Dev) and software operation (Ops).[\[https://en.wikipedia.org/wiki/DevOps\]](https://en.wikipedia.org/wiki/DevOps)

was. All other employees not belonging neither to IT nor developers groups can not even access the Private Cloud web portal. This precaution is due to the fact that the services that may be requested via Cloud are economically reported a posteriori and the IT resources in the server farm are estimated for developers needs and not to be used by everyone. Moreover, the services in the private cloud are designed keeping in mind that users are aware of the infrastructure present in the company and of the various customizations that have been made over the years. Then, for example, users should know which cluster to use or on which network to create a virtual server. Since developers are not aware of this information, the corporate private Cloud exposes to them only simple services to realize PoC, while all other more complex services are used by experienced and authorized users.

In the following subsections the backbone framework of private Cloud and the structure of services are explained.

3.2.1 Cloud Framework

The framework of the private Cloud, in Intesa Sanpaolo, is composed of three tier (**Figure 22**): Presentation, Service Orchestration and Service Providers.

Presentation Layer This is the front-end layer, in which the main component is the web portal (**Figure 24 and 25**), actually realized via Microfocus CSA⁴. It implements the user authentication and authorization via single sign on (SSO) and comprehends, furthermore, the services catalog on which users can select the service they prefer. The authorization takes place through the binding with Active Directory in which groups allowing access and use of the web portal are recorded. The webpage of a service, commonly called *form*, is composed by several fields – drop-down menus, free text input – that user will fill in with parameters like sizes for disk, cpu, ram and OS version. As said above, every forms have static fields (such as drop-down menus) that are valued via JSP⁵ scripts that retrieve proper informations from a dedicated DB. It also checks free text input against predefined policies, preventing wrong

⁴<https://software.microfocus.com/en-us/products/cloud-service-automation/overview>

⁵Is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types [...] JSP is similar to PHP and ASP, but it uses the Java programming language [https://en.wikipedia.org/wiki/JavaServer_Pages]

information entered by user. While the user fills in the fields, the portal also queries the CMDB to retrieve and verify some information. Finally, once user has submitted the request, the web portal perform a series of pre-operations to evaluate some hidden informations and then it invokes the orchestrator located in the *Service Orchestration Layer*, sending a JSON payload to it with all the informations related to the request.

Service Orchestration Layer This layer represents the core of the entire framework because it takes all the necessary informations and orchestrates the service providers (*see next paragraph*) in order to build the resource requested by the user. The actual orchestrator software is HP OO⁶ of Micro-focus HCM suite. It interacts many times with cloud portal to synergically evaluates all the necessary informations to build the IT resource. When the user request goes into orchestrator management, once web portal finished, it first select the service manifest, a JSON template reporting all the informations needed for resource creation, then, from that template, it creates an instance that contains all the informations evaluated so far. Finally it invokes low level service providers, using a specific mapper for each of them. Orchestrator continually monitors service providers waiting for responses. If something goes wrong, it stops all the process and sends a failure ticket to the competence technical group. It is almost always possible to restart the automation from the break point, without restarting the request from the beginning (manually from the web portal). In the case it is not possible, there are dedicated flows that clean up all the changes done so far, performing a sort of rollback, to free up any allocated resources. If, indeed, all operations succeed, the orchestrator performs the final censuses into CMDB and send email to the user with request confirmation and all the details regarding the resource – in the case of virtual server, the way to access it. This layer is not used only when triggered by web portal with new coming requests, but there are scheduled flows that autonomously perform checks, log activities and monitoring.

Service Providers Layer It groups together all service providers that, performing a series of operations, contribute to the creation of the final IT resource. There are seven main service providers in this layer, with some other minor ones. For Linux environment the main service provider is VMware VRO, an orchestrator that creates resources in VMware vcenter and Exa-

⁶<https://software.microfocus.com/en-us/home>

logic. This powerful software allows any kind of management on VMware resources and it is also used to run Ansible playbooks, that configure virtual servers, and to manage RedHat Satellite, that provides licenses, repository and updates on linux machines. Microsoft Orchestrator is, indeed, used to create and manage resources on Hyper-V clusters, hence, virtual Windows servers for instance. It performs more less the same operations that VRO does. The third service provider manages storage resources, with large use of the Rundeck job scheduler⁷. Then there is a TLC orchestrator that creates new rules on load balancers and manage SDN⁸, enabling network operations via cloud services. The fifth service provider is the one that deals with censuses in CMDB / GSS, set up of Change Console, the framework responsible of software change, DNS registrations and all other registration operations. Finally there are service providers for microservices and Azure services: the first one is represented by OpenShift that exposes APIs used by the high orchestrator; the last one is a set of tools, including Terraform⁹, to interface with Azure and deploy resources on this public Cloud.

3.2.2 Services

Type of Services Intesa Sanapalo Private Cloud provides services related to several technology brands (**Figure 23**) and it continually evolves to integrate new ones. Furthermore, there are dedicated services for different organizations belonging to the banking group like *Finanza*, *Fideuram - Polo Assicurativo* and the *Foreign Banks*. The main categories in which services are grouped are:

- IaaS
 - **Storage**
 - * EMC
 - * PURE
 - * NetApp
 - * Share CIFS

⁷<https://rundeck.org/>

⁸ [...] SDN architectures decouple network control and forwarding functions, enabling network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services [https://en.wikipedia.org/wiki/Software-defined_networking]

⁹<https://www.terraform.io/>

- * Share NFS
- Network
- * Load Balancer Rules
- PaaS
 - Integration Test Environment
 - Oracle WebLogic
 - RedHat JBoss
 - Microsoft IIS
 - JAVA
 - MQ and IIB
 - IBM WebSphere
 - OpenShift
- POC
 - Microsoft .NET
 - JAVA
 - Microsoft SharePoint
 - RedHat Linux
 - Microsoft Windows Server
 - MongoDB
 - Oracle DB
- DBaaS
 - Microsoft SQL Server
 - Oracle DB
 - MongoDB

The type of services includes different technologies between virtual servers, databases, runtime application environments, storage and TLC. One of the most innovative services introduced recently is OpenShift PaaS which allows the orchestration of containers.

Structure Each service in the Cloud catalog is identified with a unique *Service Identifier*. This identifier is associated with a JSON-formatted record, called *Service Manifest* that specifies various technological parameters together with service providers used for the resource creation and delivery. *Service Identifier* is recalled in several steps of the framework in order to identify proper IT options – such as cluster, network, storage typology, etc. – and workflows specific for that resource requested. **Picture 26** shows a screenshot of a table of Cloud DB in which are recorded all the associations between *Service Identifiers* and *Service Manifests*. *Service Manifest*, instead, is a template from which an instance is created every time a request is submitted from the web portal. Each instance is identified by a number called *Requisition ID* and contains all the options specified by the user – cpu, ram, disk size, etc. – and further back-end evaluations. **Listing 1** reports a simplified version of the *manifest* related to the VMware PaaS services. In conclusion, since the company is CMDB centric, i.e. all informations about IT is stored and managed by CMDB, the Cloud cannot manage the resource lifecycle independently. For this reason there is a service for each of the following operations:

- **Creation:** used only once for the each resource that becomes available from the moment of delivery.
- **Modification:** can be used more than once for each resource.
- **Elimination:** used only once for each resource (this is deallocated and becomes available again).

Listing 1: Service Manifest

```

1 {
2   "serviceIdentifier": "PaaSVMware",
3   "contentVersion": "1.0",
4   "inputsFromRequest": {
5     "RequisitionID": "${RequisitionID}",
6     "Environment": "${Environment}",
7     "Version": "${Version}",
8     "Domain": "${Domain}",
9     "Description": "${Description}",
10    "DiskSizeKey": "${DiskSizeKey}",
11    "DiskSize": "${DiskSize}",
12    "ServiceOwner": "${ServiceOwner}",
13    "Heap-size-giga": "${Heap_size_giga}",
14    "MemorySize": "${MemorySize}",
15    "Platform": "${Platform}",

```



```

16     "User": "${User}",
17     "OS": "${OS}",
18     "Acronym": "${Acronym}",
19     "CPU": "${CPU}",
20     "BackupPolicy": "${BackupPolicy}",
21     "DR": "${DR}",
22     "jdk-patch-level": "${jdk-patch-level}",
23     "jax-rs": "${jax-rs}",
24     "authentication": "${authentication}"
25 },
26 "inputsFromAutomation": {
27     "Datacenter": "${Datacenter}",
28     "vCenter": "${vCenter}",
29     "Cluster": "${Cluster}",
30     "Hostname": "${Hostname}",
31     "StorageLevel": "${StorageLevel}",
32     "Networks": "${Networks}",
33     "Port": "${Port}"
34 },
35 "prerequisites": [
36     "Site",
37     "Datastore",
38     "Hostname",
39     "Port"
40 ],
41 "serviceProviders": [
42     "CMDB",
43     "DNS",
44     "VRO",
45     "ChangeConsole",
46     "GSS"
47 ]
48 }

```

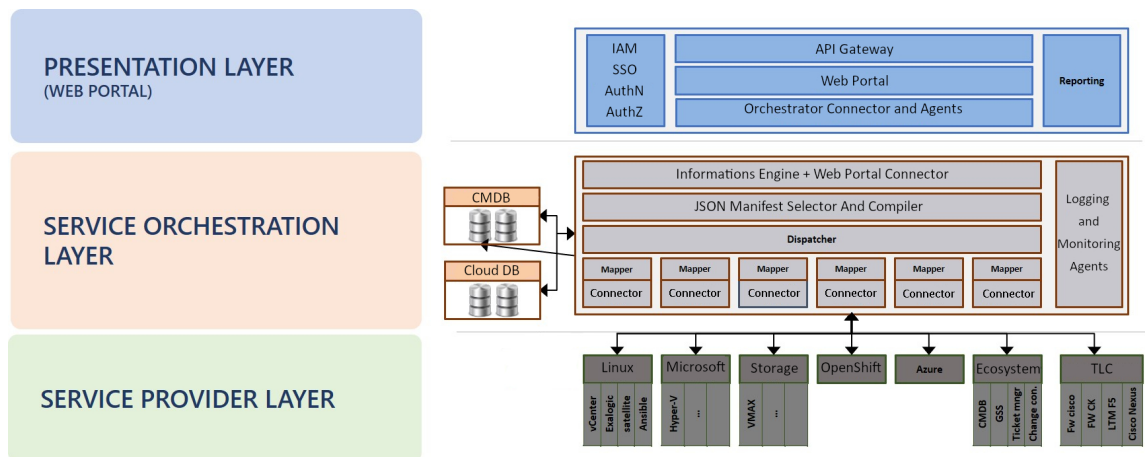


Figure 22: Private Cloud framework in Intesa Sanpaolo



Figure 23: Brands

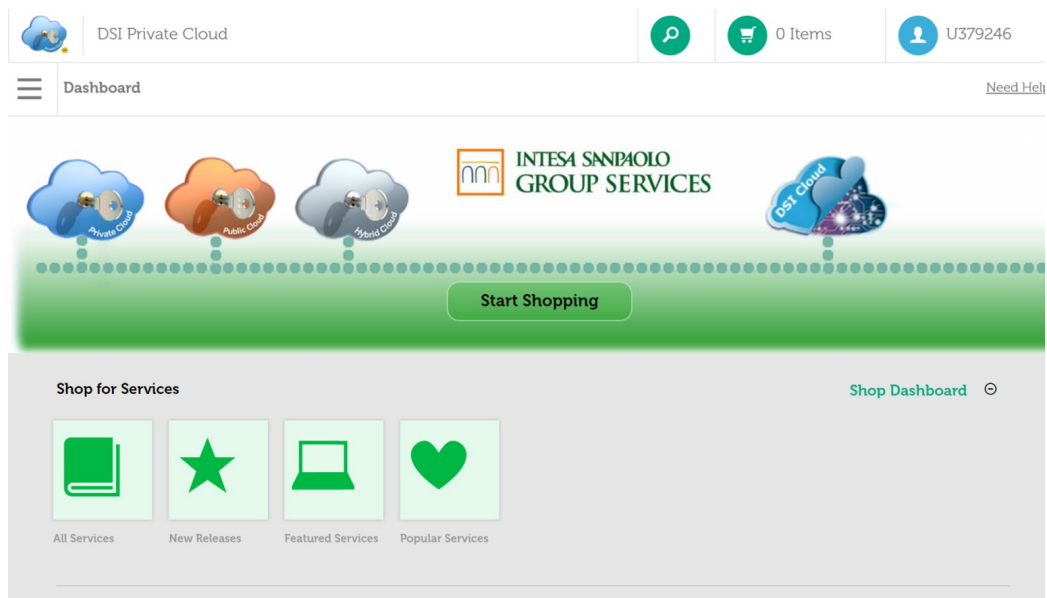


Figure 24: Cloud Web Portal

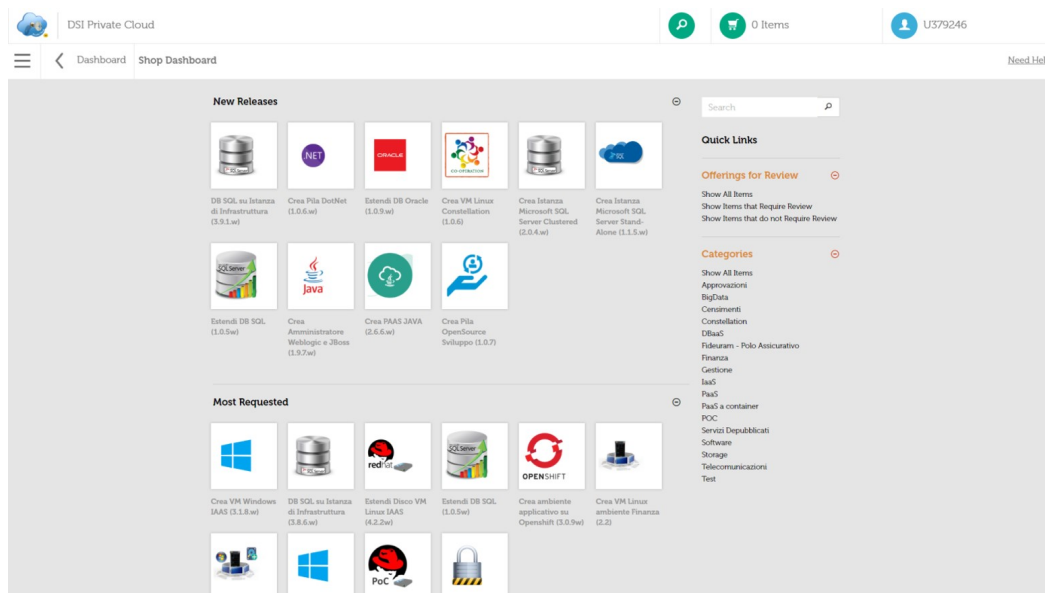


Figure 25: Cloud Web Portal

	ServiceIdentifier	ServiceManifest
1	CreaDBMongo	{"serviceIdentifier":"CreaDBMongo","contentVer...
2	CreaF5	{"serviceIdentifier":"CreaF5","contentVersion":"...
3	CreaPaaSMicrosoft	{ "serviceIdentifier": "CreaPaaSMicrosoft", "cont...
4	CreaPaaSVMware	{"serviceIdentifier":"CreaPaaSVMware","content...
5	CreaPoCAzureLinux	{"serviceIdentifier":"CreaPoCAzureLinux","conte...
6	CreaPoCAzureWindows	{"serviceIdentifier":"CreaPoCAzureWindows","c...
7	CreaShareCIFS	{ "contentVersion": "0.1", "serviceIdentifier": "Cr...
8	CreaShareNFS	{"contentVersion":"0.1","serviceIdentifier":"Crea...
9	CreaStorageHyperV	{ "serviceIdentifier": "CreaStorageHyperV", "con...
10	CreaStorageNetApp	{ "serviceIdentifier": "CreaStorageNetApp", "con...
11	CreaStoragePure	{ "serviceIdentifier": "CreaStoragePure", "conten...
12	CreaVMCitrixMicrosoft	{"serviceIdentifier":"CreaVMCitrixMicrosoft","con...
13	CreaVMISMicrosoft	{"serviceIdentifier":"CreaVMISMicrosoft","conte...
14	CreaVMMicrosoft	{"serviceIdentifier":"CreaVMMicrosoft","content...

Figure 26: Screenshot Of ServiceIdentifier/ServiceManifest DB table

3.3 OpenShift

3.3.1 Platform Selection Process

During last years of Cloud development the need of orchestration platform for microservices arose in Intesa Sanpaolo. Hence some main brands was analyzed to check compliance against company regulation. The most known firm that offered that kind of product were:

- RedHat with OpenShift
- Pivotal with Cloud Foundry
- Docker with Docker Datacenter
- IBM with Bluemix
- Oracle with Oracle Cloud

The last two listed platforms have not been deeply analyzed because they were fully managed by vendor and this represented a sort of lock-in that did not fit with company regulation. The first two platforms, indeed, have been tested against the main technologies used so far: three Java Stand Alone applications, one application running on JBoss runtime environment, one application running on Weblogic runtime environment and one MongoDB backend have been cloned into containers orchestrated by Openshift and Cloud Foundry. Conversely, Docker Datacenter has been tested using parallel development environment for MUREX softwares. At the end of the experimentation, a comparative analyses has been produced (*Figure 27*):

1. Three evaluation areas were defined
2. Evaluation criteria for each area were adopted
3. A score from 1 to 6 was associated to each platform

The choice was driven by the total score gained by the platform and some considerations about its maturity in enterprise context.

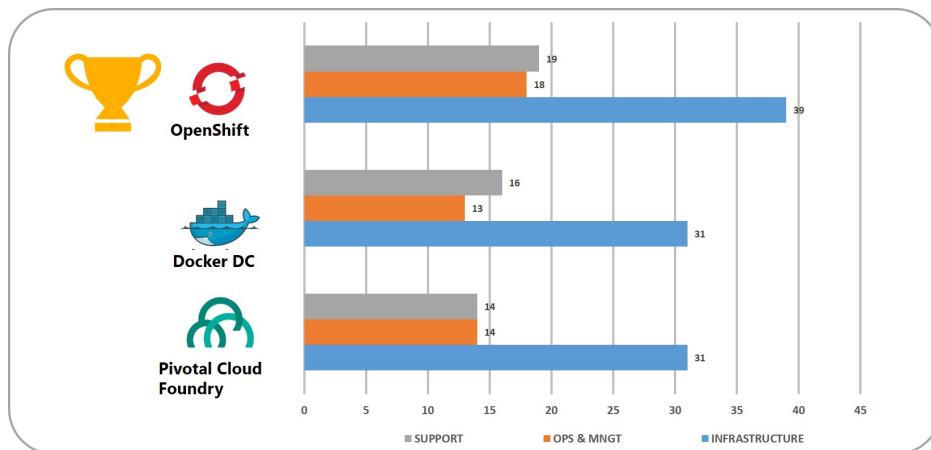


Figure 27: Final Results

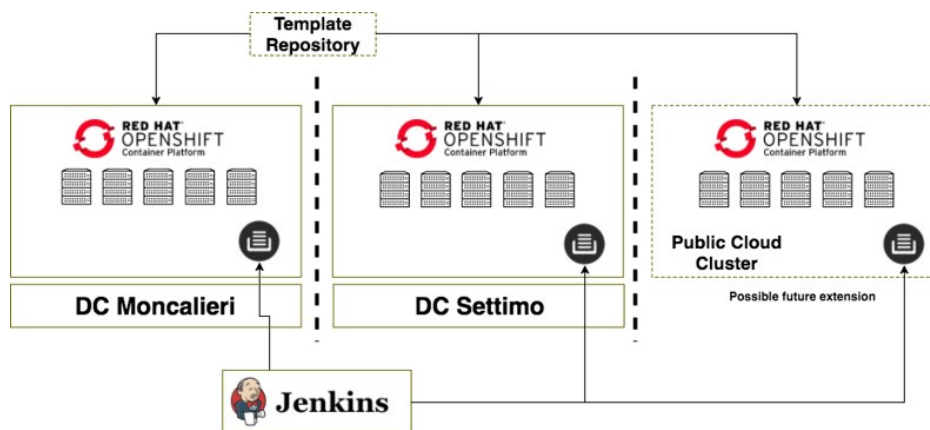


Figure 28: OpenShift Architecture in Intesa Sanpaolo

3.3.2 Architecture

To fulfill the development cycle adopted by Intesa Sanpaolo, Openshift architecture was built with the ability to supply containers in the three main environments:

- *Development* – Used by developers
- *System Test* – Used for quality test
- *Production*

The first two environments are provided from the same Openshift installation, whilst the *Production* one has a dedicated installation. Developers can access only the first environment because the *System Test* and *Production* deployment is done via *Change Console*, a specific internal software for application change management integrated with GitHub Enterprise and CloudBees Jenkins Enterprise instances. Another *Laboratory* installation of Openshift was built in order to test and evolve the platform itself.

The physical hardware that hosts Openshift instances is spread among different servers located in Moncalieri and Settimo Torinese datacenters, in order to guarantee geo-redundance. Another site for *failover* has been located in Parma datacenter. The virtual servers running OpenShift are located into VMware ESXi cluster. There are two independent clusters, the first one hosts *Laboratory*, *Development* and *System Test* instances, the last one hosts *Production* instance only. Furthermore, there are three available storage types according to data replication requirements:

- **Bronze:** primary datastore hosted in Moncalieri; it is used for non-critical environments (e.g. *Laboratory*)
- **Silver:** Primary datastore hosted in Moncalieri with asynchronous replica in Parma datacenter; it is mainly used for applications that require disaster recovery
- **Gold:** Primary datastore hosted in Moncalieri with synchronous replica in Settimo Torinese datacenter and asynchronous replica in Parma datacenter; it is mainly used for applications that require disaster recovery (high availability and disaster recovery)

The network between Moncalieri and Settimo Torinese datacenters in an layer 2¹⁰ stretched network. The synchronous / asynchronous replica data is performed at storage level.

Each microservice running in a container is stateless by definition, i.e. does not need persistent storage to save data because it interfaces with other microservices dedicated to do so. In some cases, however, it could be necessary for a microservice to save its computed data for further operations. Since, generally, distinct executions of an application can be done in different nodes belonging to the OpenShift cluster, persistent storage accessible from the cluster's network was required. In this way NFS¹¹ was chosen as a solution for all the environments. So, if an application has this special need regarding persistent storage the following step are executed:

1. Storage is provisioned manually on NFS servers
2. Specific authorizations are associated to the created storage area
3. The directory is created and exported on the NFS server
4. After prvisioning, the storage area is linked to the application creating a *Persistent Volume* or a *Persistent Volume Claim*

OpenShift constantly checks *pod* status and infrastructure monitoring is performed via native tools, like **Hawkular** and **Heapster**, collecting metrics that are stored on a dedicated Cassandra database, located on a persistent 35GiB volume. Other performance checks are performed via **HP Performance Manager**, **Application Performance Manager (APM)** and other internal softwares.

Finally, in order to avoid unauthorized changes performed by developers on applications, OpenShift implements **RBAC policies** *out of the box*. In this way is possible to assign one or more predefined roles to each user.

For what concern the Cloud service, the web portal exposes a rather long form in which to enter the service parameters. **Figure 31** reports the first two part of the form related to user and economic details:

- the user part – the highest one – is self-filled using SSO parameters

¹⁰ISO / OSI stack

¹¹Network File System, *see glossary*

- the second part must be filled by the user
 - the first field is optional: if user has made an economic estimate through the internal dedicated portal must enter here the budget ID
 - **WBE** field is a dropdown menu and states the office that will pay the resources
 - last field concerns the year of expenditure

This part is common to all of Cloud services.

Figure 33 shows the main body of the form in which infrastructural parameters must be entered.

1. **Ambiente:** Dropdown menu where to select one between *Sviluppo*, *Test* and *Produzione*
2. **Acronimo:** The application code registered on CMDB
3. **Descrizione Acronimo:** Description of *acronimo*. Is self-filled (the value is retrieved from CMDB)
4. **Criticità Acronimo:** Identify the importance of the application and help support groups to give the correct priority. User can select one of the following value:
 - *Non Critico*
 - *Critico*
 - *Vitale*
5. **Cluster:** self-filled value, evaluated using the value of field *Ambiente* – specify the right cluster between *Development / Test* and *Production* ones
6. **Nome microservizio:** user can specify here the name of the microservice
7. **Verifica Ambiente:** self-filled fields that states if the environment check succeeded or not
8. **Tecnologia:** user can specify the technology used by his application (e.g. Java, Python, etc.)

9. **Versione:** user can specify the version of chosen technology
10. **Prodotto:** static *no* value (field for future uses)
11. **Size:** user can specify the size of the *POD* between the following:
 - Tiny – 1 CPU / 1 GB RAM
 - Small – 2 CPU / 2 GB RAM
 - Medium – 4 CPU / 4 GB RAM
 - Large – 8 CPU / 8 GB RAM
 - Extra-Large – 16 CPU / 16 GB RAM
12. **Infrastruttura:** user can choose from *Intranet* and *Internet*; the last one gives possibility to interface with internet, hence, to expose the service "outside"
13. **Context root:** Optional field; user can specify a custom endpoint with which to communicate with the application
14. **Verifica Nome Microservizio e Context root:** self-filled fiels; gives the result of checks performed on microservice name and context root
15. **Autenticazione:** user can choose hte type of authentication for his application between
 - swa2 (intranet)
 - fiam (internet)
 - no-auth (intranet and internet)
16. **Verifica Tipo Autenticazione:** self-filled fields with the result of authentication checks
17. **Verifica Porta:** self-filled field with the result of TCP port checks
18. **Utenza web:** self-filled field that gives the auto-computed value of web user (a local user to perform some operations on th front-end size)
19. **Verifica Utenza Web:** self-filled field with the result of web user checks (it checks if user already exists)
20. **Dominio:** user can choose the domain in which the application will be placed

21. **Service Owner:** Specify the owner of the service; this value is useful in cas of failures because support groups will work together with him to restore the service

The JSON *Listing 2* is the OpenShift service *Manifest* on the Cloud portal. Most of the fields it contains correspond to those found in the form described above. In *InputsFromRequest* object there are fields for values that will be produced by web portal (user inserted values and auto-computed ones). In *InputsFromAutomation* object, instead, there are further fields that will contain values produced by the first orchestrator (the middle layer in the Cloud Framework – see *Figure 22*). Workflows that will produce these values are specified into *Prerequisites* object. Finally in *ServiceProviders* there is a list of service providers (the ones situated in the bottom layer in the Cloud Framework – see *Figure 22*) that will produce each part of the resource required. In the case of OpenShift service, there are two service providers only:

1. **VRO:** it prepares some configurations and run C4 to start the provisioning on OpenShift platform (through REST APIs)
2. **GSS:** it register the microservices application to CMDB / GSS

The JSON *Listing 3* represents an instance of the service *Manifest*. One of this is produced every time a new Cloud request – for the specific service – is submitted. It contains all the values displayed in the form of Cloud web portal together with the others described above.

Listing 2: Openshift Cloud Service Manifest

```

1 {
2   "serviceIdentificator": "CreaPaaSOpenshift",
3   "contentVersion": "0.1",
4   "inputsFromRequest": {
5     "annoCompetenza": "${annoCompetenza}",
6     "Service-level": "${Service_level}",
7     "RequisitionID": "${RequisitionID}",
8     "WBE": "${WBE}",
9     "SLA": "${SLA}",
10    "ServiceName": "${ServiceName}",
11    "Type": "${Type}",
12    "C4Environment": "${C4Environment}",
13    "Environment": "${Environment}",
14    "ARuolo": "${ARuolo}",

```

```

15     "Layer": "${Layer}",
16     "Bank": "${Bank}",
17     "Servizio": "${Servizio}",
18     "Version": "${Version}",
19     "Domain": "${Domain}",
20     "CriticoAcronimo": "${CriticoAcronimo}",
21     "Descrizione": "${Descrizione}",
22     "ServiceOwner": "${ServiceOwner}",
23     "Infrastructure": "${Infrastructure}",
24     "NomeServizioGSS": "${NomeServizioGSS}",
25     "SistemaInformativo": "${ServizioInformativo}",
26     "User": "${User}",
27     "ServicePhase": "${ServicePhase}",
28     "Service-name": "${Project_name}",
29     "IdProfiloDR": "${IdProfiloDR}",
30     "Acronym": "${Acronym}",
31     "ServiceOfficeOwner": "${ServiceOfficeOwner}",
32     "DR": "${DR}",
33     "isPaaS": "${isPaaS}",
34     "http-hostname": "${http-hostname}",
35     "authentication": "${authentication}",
36     "C4Profile": "${C4Profile}",
37     "Project-name": "${Project_name}",
38     "size": "${size}",
39     "openshift_cluster": "${openshift_cluster}",
40     "description": "${description}",
41     "ssa_profile": "${ssa_profile}",
42     "wildcard": "${wildcard}",
43     "path-http": "${path-http}",
44     "web-user": "${web-user}",
45     "http-port": "${Port}",
46     "language": "${language}",
47     "source": "${source}"
48 },
49 "inputsFromAutomation": {
50     "webservers": "${webservers}"
51 },
52 "prerequisites": [
53     {
54         "prerequisiteIdentificator": "Webservers",
55         "sequenceNumber": "1",
56         "prerequisiteOutput": [
57             "webservers"
58         ]
59     }
60 ],
61 "serviceProviders": [
62     {
63         "providerIdentificator": "VR0",

```

```

64     "serviceIdentificatorInProvider": "c18d8c68-2289-41ec-
        ab80-7d65bac519b9",
65     "payloadIdentificator": "C4",
66     "sequenceNumber": "1",
67     "providerOutputs": [
68         "listen-port",
69         "url",
70         "siteminder-version"
71     ]
72 },
73 {
74     "providerIdentificator": "GSS",
75     "serviceIdentificatorInProvider": "InsertService",
76     "payloadIdentificator": "APService",
77     "sequenceNumber": "2",
78     "providerOutputs": [
79         "IDServizioGSS"
80     ]
81 }
82 ]
83 }

```

Listing 3: Openshift Cloud Service Manifest Instance

```

1 {
2     "serviceIdentificator": "CreaPaaSOpenshift",
3     "contentVersion": "1.0",
4     "inputsFromRequest": {
5         "annoCompetenza": "2018",
6         "Service-level": "standard",
7         "RequisitionID": "49100",
8         "WBE": "CLOUD",
9         "SLA": "",
10        "ServiceName": "Crea VM",
11        "Type": "OCP",
12        "C4Environment": "svil",
13        "Environment": "Sviluppo",
14        "ARuolo": "",
15        "Layer": "AP",
16        "Bank": "MB",
17        "Servizio": "PaaS Openshift",
18        "Version": "1.8",
19        "Domain": "sede.corp.sanpaoloimi.com",
20        "CriticoAcronimo": "Non Critico",
21        "Descrizione": "",
22        "ServiceOwner": "LUCA GIOMMONI",
23        "Infrastructure": "Intranet",
24        "NomeServizioGSS": "cpaas",
25        "SistemaInformativo": "INTESASANPAOLO",

```

```

26     "User": "U379246",
27     "ServicePhase": "Preallocazione",
28     "Service-name": "CLOU0",
29     "IdProfiloDR": "",
30     "Acronym": "CLOU0",
31     "ServiceOfficeOwner": "No Referente Applicativo",
32     "DR": "",
33     "isPaaS": "2",
34     "http-hostname": "cpaas-CLOU0-svil.cloudapps-be-test.
        intesasanpaolo.com",
35     "authentication": "SWA2",
36     "C4Profile": "standard",
37     "Project-name": "CLOU0",
38     "size": "medium",
39     "openshift_cluster": "ocp-test-api.syssede.systest.
        sanpaoloimi.com",
40     "description": "Project used by CLOU0 new platform for
        collaboration APIs management",
41     "ssa_profile": "Y8",
42     "wildcard": "cloudapps-be-test.intesasanpaolo.com",
43     "path-http": "\\cpaas\\api",
44     "web-user": "wwtclou0",
45     "http-port": "",
46     "language": "JAVA"
47 },
48 "inputsFromAutomation": {
49     "webservers": ""
50 },
51 "prerequisites": [
52     {
53         "prerequisiteIdentificator": "Webservers",
54         "sequenceNumber": "1",
55         "prerequisiteOutput": [
56             "webservers"
57         ]
58     }
59 ],
60 "serviceProviders": [
61     {
62         "providerIdentificator": "VRO",
63         "serviceIdentificatorInProvider": "c18d8c68-2289-41ec-
            ab80-7d65bac519b9",
64         "payloadIdentificator": "C4",
65         "sequenceNumber": "1",
66         "providerOutputs": [
67             "listen-port",
68             "url",
69             "sitemap-version"
70         ]

```

```

71     },
72     {
73         "providerIdentiicator": "GSS",
74         "serviceIdentiicatorInProvider": "InsertService",
75         "payloadIdentiicator": "APService",
76         "sequenceNumber": "2",
77         "providerOutputs": [
78             "IDServizioGSS"
79         ]
80     }
81 ]
82 }

```

3.3.3 Delivery Process

The OpenShift platform can only be used after registering a microservice through the Cloud portal, which displays a special service. To integrate the existing ecosystem and technology with the infrastructure made available by OpenShift, a powerful Cloud automation has been created that takes care of performing all the necessary censuses on CMDB and environments provisioning for the new created project.

The application environment delivery process (**Figure 29**) concerning OpenShift platform follows five main steps:

- A developers group send to *Project Coordination Office* details about the application sub-system – i.e. the logical namespace domain in which the application must be placed – and the size of the *POD* (CPU & RAM)
- *Project Coordination Office* registers a new *Acronym (acronimo)* of container type, specifying Openshift as container platform and GitHub as versioning tool
- *Project Coordination Office* uses Cloud to create a new project in a development environment in Openshift
- Change configurations and OCP-Configuration repository are created in *Change Console*
- The developers group receive URL to connect to OpenShift console related to the new created project and the GitHub endpoint to perform code versioning

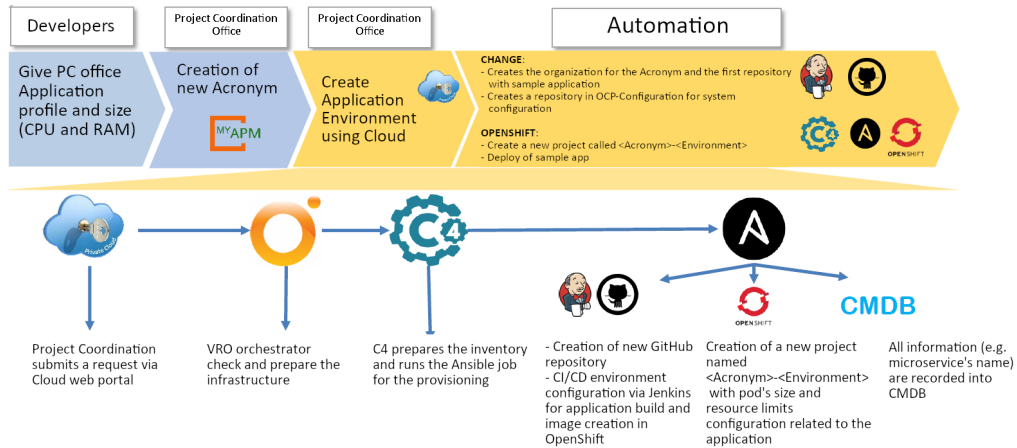


Figure 29: Cloud + Openshift Provisioning

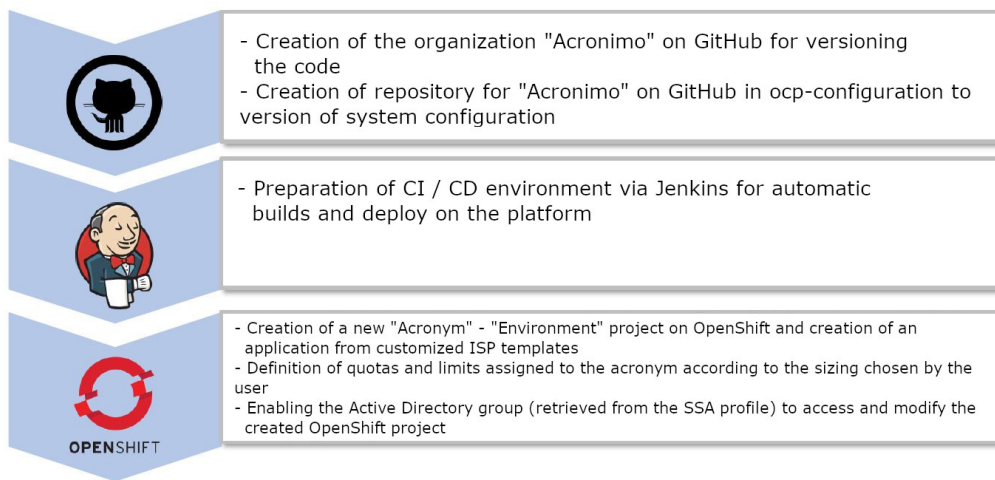




Figure 30: New Openshift Project



Crea ambiente applicativo su Openshift

PaaS a container

Publicato il 06/09/2018 14:49:05



Richiedente

- Dettagli**

Option Description

Nome Utente

U379246

Nome del Richiedente

LUCA GIOMMONI

Email del richiedente

luca.giommoni@intesasanpaolo.com



Inserimento WBE

- Dettagli**

CONTEST

Inserire ID Stimma censito su portale CONTEST

WBE

Selezionare il tipo di WBE, per casi non elencati o chiarimenti contattare coordinamento.progetti@intesasanpaolo.com


CLOUD - A. BRUNO

Anno di Competenza

Seleziona l'anno di competenza della WBE che stai utilizzando

2018

Figure 31: User Details




Notifiche Aggiuntive

- Dettagli**

Destinatario Aggiuntivo (opzionale)

Inserisci qui l'indirizzo email aggiuntivo che riceverà le sole notifiche della richiesta (es la mail del tuo gruppo di lavoro).

Figure 32: Further Optional Details


Richiedi ambiente

• Dettagli

Ambiente

Seleziona l'ambiente in cui verranno generati i servizi IT che stai richiedendo

Sviluppo

Acronimo

Acronimo dell'applicazione

cloud

Descrizione Acronimo

APERTO - PRIVATE CLOUD

Criticità Acronimo

Seleziona la criticità del servizio

Vitale

Cluster

ocp-test-api.sysedge.systest.sargpaolomi.com

Nome microservizio

Nome utilizzato per connettimento su platform e CDS

CPAAS

Verifica Ambiente

Ambiente Sviluppo idoneo

Tecnologia

JAVA

Versione

1.8

Prodotto

No

Size

medium

Infrastruttura

Intranet

Context root

Indirizzo applicazione. Se lasciato all'OCCORRENZA, il default sarà il nome microservizio

cpaas/api/

Verifica Nome Microservizio e Context Root

Dati validi

Autenticazione

swa2

Verifica Tipo Autenticazione

Campo che calcola il ServiceIdentifier in base al tipo di autenticazione scelto dall'utente

Tipo autenticazione SWA2 valido

Verifica Porta

Controllo porta OK

Utenza web

Verifica UtenzaWeb

VALIDO

Dominio

cloudapps-test.intesasanpaolo.com

Service Owner

Figure 33: Tecnological Parameters

4 Collaboration Platform As A Service

4.1 Main Idea

The major problems addressed, explained in *Section 2*, required the solution to provide stability, security and the creation of a new ecosystem that groups together all the APIs, and new derived functionalities, related to the collaboration products present inside the company. This comprehends also a set of secondary requirements that are not satisfiable with a simple infrastructure or a unique monolithic software. So the best idea chosen to solve these needs was to exploit the flexibility and power offered by the Cloud and microservice architecture to create an abstraction layer that will expose uniform APIs for ECS services. This allows us to add or remove functionalities without having to restructure the architecture every time and also provides centralized access to collaboration services, creating a single endpoint to expose the APIs: new functionalities can be implemented with the most appropriate technology and in a completely transparent way for developers. Such a large solution requires monitoring, high reliability and security and, again, the microservice architecture offered by the Cloud and OpenShift meets the requirements well. In fact, OpenShift natively implements many monitoring tools and integrates well with those already existing in the company. Furthermore, as implemented in Intesa Sanpaolo, it already offers out-of-the-box high reliability and disaster recovery. Finally, container technology natively provides kernel-level isolation and thanks to the RBAC policies adopted by OpenShift, it is possible to create customized access rules.

In conclusion, many options were considered: at the beginning we thought of creating a single application in a Java environment that would have provided the only point of contact with the collaboration services and that would have been performed on the WebLogic application server. But this solution did not meet the need for flexibility and stability and would require many additional monitoring and management tools. Moreover it would not allow to easily modify the structure of the framework, forcing to do heavy non-regression tests at each update. Finally, it would not have been possible to integrate features written in other programming languages. Fortunately, in recent years the microservice infrastructure has been consolidated at Intesa Sanpaolo, which fully meets all the requirements in question. For these reasons the choice fell on it.

4.2 Requirements

4.2.1 Functionals Requirements

Architecture

- Continuous delivery and continuous integration paradigm must be possible — this will allow to update services without outages
- Updates must not cause any service interruption — the architecture of each microservices must be properly designed
- On demand scale out must be possible — any time a usage peak occurs, the infrastructure must be able to manage the situation without outages
- High availability must come from deployment on different pools — multiple pools guarantee high availability of infrastructure
- Disaster recovery must come from datacenter at least 300 km away — in case of disaster (completely loss of main datacenter) service continuity must be guaranteed

Reachability

- APIs must be provided via REST paradigm — communications are more flexible and more suitable for applications
- There must be a single alias to access endpoints — this decreases complexity and increases overall management
- The platform must be reachable from Intranet
- The platform must be reachable from Internet via VPN

Access

- Access restricted to specific clients only — to avoid uncontrolled access and to increase stability

- User authentication via Active Directory binding — to allow access control via authenticated integration
- User authorization via Active Directory ad hoc groups — to restrict access to authorized user only
- Access via SSO only — to avoid some sort of hacking and to facilitate the connections
- Each access must be recorded in a dedicated database — to allow audit and troubleshooting

Data

- Exchanged data must be JSON formatted — this is a commonly adopted standard, already widespread into Intesa Sanpaolo; it is easy to read and debug
- Encoding must be UTF-8 — this is also an already present standard in the company and reflects the applications' dictionary
- Payload of each request / response must be at most 10 MBs — to avoid loss and retransmission of too heavy packet and to allow a better traffic management

Regulatory / Compliance

- Each application must have a dedicated user ID — to ascribe each activity to the right application and facilitate audit and troubleshooting
- Each user ID must be associated with strictly necessary Active Directory groups only — this guarantees the minimum privileges paradigm
- Passwords related to each user ID must be managed by Operational Management Bureau only — this is a common Intesa Sanpaolo security policy

Security

- No privileges escalation admitted — this prevents users and applications to perform high level unauthorized operations
- All communication (both intranet and internet) must be protected through SSL / TLS — this is a common infrastructure into the company that has a dedicate CA¹²
- APIs gateway must be placed in DMZ network — to be reachable from internet without security risks
- All databases must be placed in hardened network — to enhance security and prevent data leak

Session

- Each session must last at most one hour — to increase session security and stability and avoid stall
- Each session must be retrievable in logs or databases using a unique session identifier — for audit and troubleshooting
- For each session the following information must be collected
 - Start time
 - End time
 - Total duration of the session
 - User ID as recorded in Active Directory
 - List of operations
 - Types of operations
 - Amount of received data for each operation
 - Amount of sent data for each operation
 - Total amount of exchanged data for each operation
 - Total amount of received data for the entire session
 - Total amount of sent data for the entire session

¹²Certification Authority

- Total amount of exchanged data for the entire session
- Type of data exchanged for each operation
- Percentage of each type of data exchanged in the session

Client Informations

- User-Agent
- IP address
- Geolocality

LOG

- Logging must not be done locally but via dedicated remote servers
- Each operations must be logged
- Each logged record must report
 - Timestamp
 - Session ID
 - Event type
 - Event description

4.2.2 Non-Functionals Requirements

Communication

- Response time must below 3 seconds max
- In case of lost connection, session data must last at most 30 minutes and, if connection is not set up again, they must be deleted

Reachability

- In case of disaster event (the main pool goes down), the platform must return available in at most two hours to make less outage possible
- Data replication at most every second to guarantee minimum data loss

Regulatory / Compliance

- All data related to a specific user must be retrievable in at most one hour in order to facilitate audit and to better fulfill GDPR

4.3 Architecture

The platform has been designed in order to satisfy the requirements of flexibility, security, agility and reliability. To satisfy these containers infrastructure was chosen to implement a microservices architecture in which each container implements a functionality. This way, each container can be managed independently from the others, ensuring stability and reliability of the framework. If there is a need to update or restructure a microservice, this operation can be done in a very low impact mode. In addition, there is no need to develop the entire platform strictly with one programming language only because each container can host a different environment from the others. This allows to leverage the language that better fit the functions needs. Finally, the central OpenShift console allow us to manage and monitor all the containers belonging to the platform, making consumption reports, security analysis and enhancement suggestions. Globally, the platform has been structured on three layers logical as explained in the following paragraphs.

Front-End Exposes the API used by applications (*Figure 35*). In this first version all its functions is implemented with four container only: two for intranet domain and two for the internet. In this layer access control and communication proxy activities are performed. Once the authentication and authorization to access the platform as been proved by the first line containers, the subsequent connections will be managed by the second line container only. The user check is performed using Active Directory APIs, overtaking the middleware.

Middleware This is the core of the platform and hosts all the microservices implementing business logics of collaboration services. They receive requests from the front-end and perform operations with backend. When the work in done, they send responses to the front-end that proxies them to the application. It is possible to add as microservices as the number of ECS services that can be exposed. The power of the solution can be well understood here because the flexibility provided by microservices architecture allows at any time to add or remove functionality, on demand, without affecting the others. Development cycle is faster and the APIs management is fully centralized.

Back-End This layer groups together all the external and supporting functionalities like databases for sessions and logs recording, Active Directory, documental server and so on. It provides the way with which middleware microservices performs their operation with the non-ECS applications. Furthermore here is possible to meet the last requirement of filling the gap between all business features and ECS ecosystem.

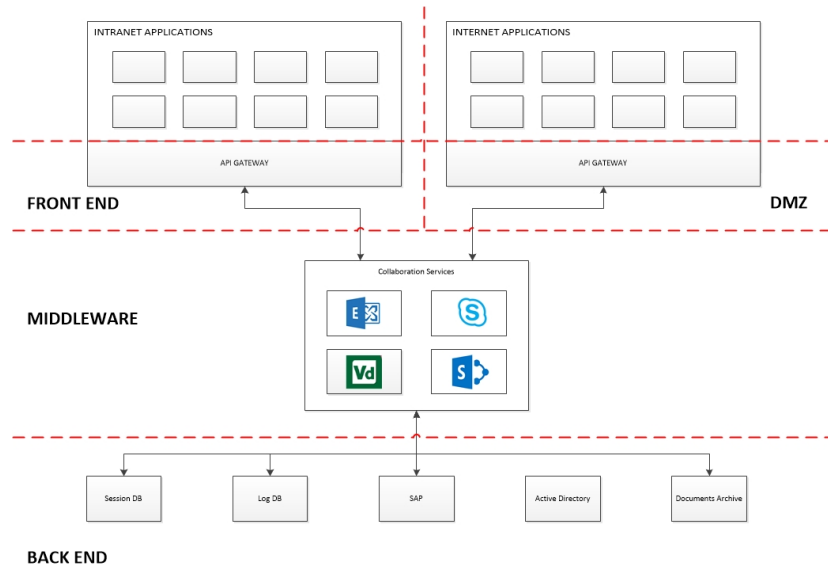


Figure 34: Platform Architecture

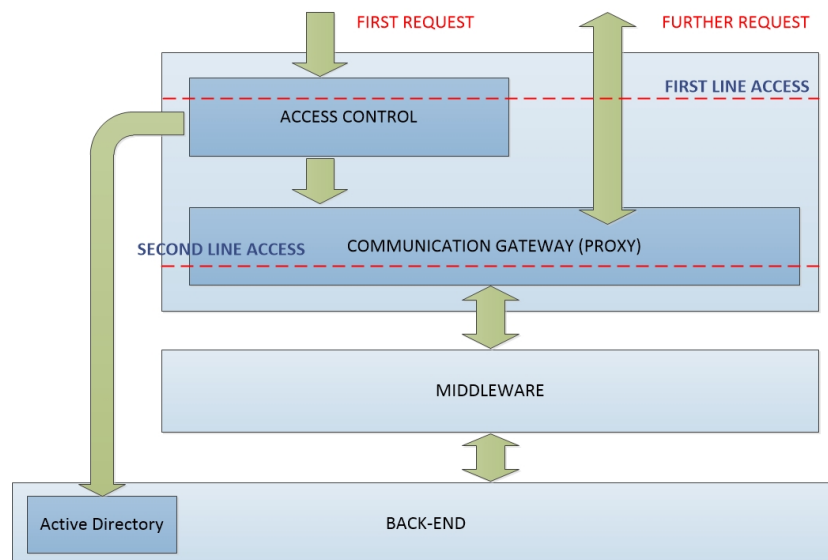


Figure 35: Frontend

4.4 Use Cases

In this chapter two use cases are reported. These concern the main microservices added to the platform. **Figure 36** shows a screenshot of web interface of Swagger reporting the APIs exposed.

4.4.1 Alarm Center

In the following use case, the application used by the Alarm Center will be called ***The Application*** while the developed platform will be called ***The Platform***

1. ***The Application*** receives an alarm related to an organizational unit (OU)
2. ***The Application*** sends the OU identifier to ***The Platform***
3. ***The Platform*** sends the OU identifier to the *Registry Service* microservice
4. The Registry Service microservice sends back to ***The Platform*** the list of *SIP* addresses of the Skype users whose OU number is the same as the one from which the alarm started and who have the status equals to active
5. ***The Platform*** sends to ***The Application*** the *SIP* addresses
6. ***The Application*** uses the APIs provided by ***The Platform*** to send a warning Skype message to the available colleagues belonging to the OU
7. ***The Platform*** uses the Skype APIs to deliver the message to the users specified
8. Skype actually delivers the message
9. ***The Platform*** polls on Skype to retrieve any messages sent back by the contacted users
10. If some message has been sent by users, ***The Platform*** retrieves it from Skype
11. ***The Platform*** delivers the message to ***The Application***

SharepointFiles			▼
PUT	/api/sharepoint/files	New file upload	🔒
SkypeConnection			▼
PUT	/api/skype/connections	New Skype connection	🔒
DELETE	/api/skype/connections/{connectionId}	Disconnect from Skype	🔒
SkypeConversation			▼
GET	/api/skype/conversations/{conversationId}	Return all messages belonging to a specific conversatio	🔒
POST	/api/skype/conversations/{conversationId}	Update an existing conversation	🔒
DELETE	/api/skype/conversations/{conversationId}	Delete an existing conversation together with all its messages	🔒
PUT	/api/skype/conversations	Create new conversation(s)	🔒
POST	/api/skype/conversations	Update more existing conversations	🔒
SkypeUser			▼
GET	/api/skype/users	Return Skype user(s)	🔒

Figure 36: Swagger Skype APIs

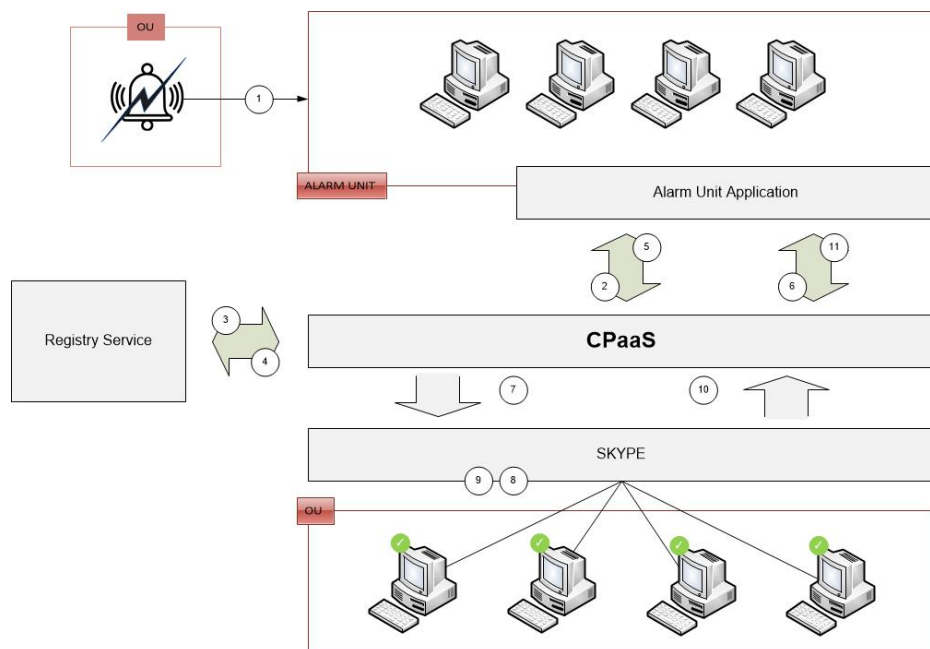


Figure 37: Alarm Center use case in Intesa Sanpaolo

4.4.2 GDPR

This use case is very important for the Bank in order to be more compliant with GDPR that states:

“The controller shall provide a copy of the personal data undergoing processing. ”

— GDPR – Article 15 (3)

The *Platform* provides a crucial functionality for attempting this purpose. In fact it enables the copy of data in a safe place from which the controller can download them and subsequently deliver to the subject.

In the following, the employee of the Bank will be called *The Employee* while the developed platform will be called *The Platform*

1. *The Employee*, through a *crawler* application, retrieves all the files related to one of his clients
2. The *crawler* application search files into the farm of documentations and makes a compressed *zip* archive of them
3. *crawler* application sends the compressed archive, in binary format, together with the user ID of *The Employee* to *The Platform* via APIs
4. *The Platform* retrieves the *url* of Sharepoint MySite of *The Employee* by means of *Registry Service* microservice
5. *The Platform* stores the archive into Sharepoint MySite of *The Employee*
6. *The Platform* sends a return code to the *crawler* application
7. The *crawler* application sends a notification to *The Employee*
8. *The Employee* can download / read the contents required from his Sharepoint MySite

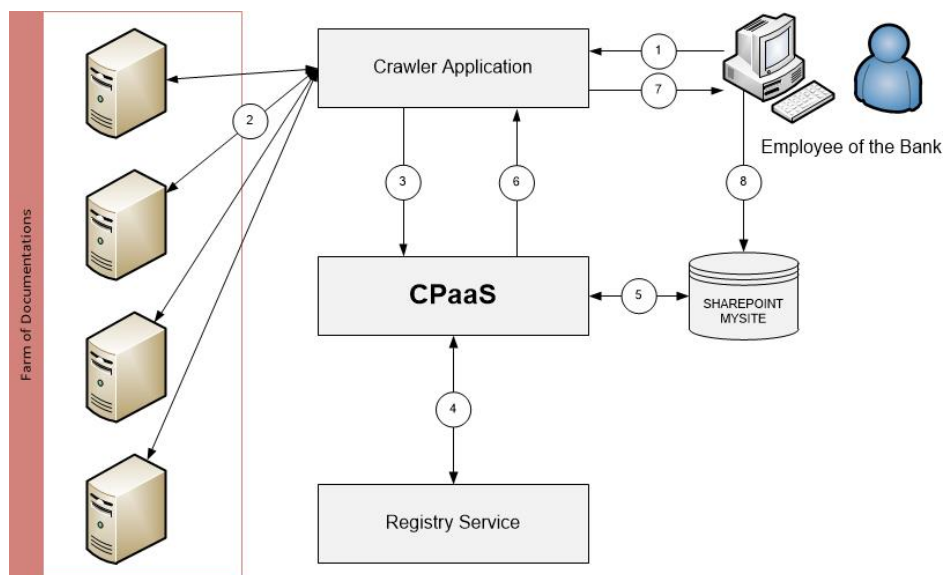


Figure 38: GDPR use case in Intesa Sanpaolo

5 Conclusions

Nowadays collaboration has become indispensable for companies. While years ago companies were forced to create their own dedicated infrastructure, today there are many possibilities offered by the market, such as the Google or Cisco suite, which give the possibility to buy only the services, without the need to install anything physical. In this scenario, small companies and startups that do not want or can not bear the costs of such infrastructures fall very often. The medium and large companies instead prefer, for internal or legislative needs, to create their own dedicated infrastructure. These are usually forced to face many problems related to the ECS that they have created. In the case of Intesa Sanpaolo, the main problems are due to the heterogeneity of the services that make up the ECS and the difficulty that developers encounter when using them, due to changes in version or differences in interfacing. In fact every product has its architecture and API that evolves with the different versions that the vendor releases. So the applications that interface with these products must evolve in the same way, but with a multiplying factor linked to the number of products with which they interface. Last but not least, security, a very important aspect especially in a banking system like that of Intesa Sanpaolo, is declined for each of the collaboration products adopted.

To solve the major problems that ECS exposes, many options have been considered, including that of creating a single Java application that would have been run on a WebLogic application server. However, this solution proved to be unsuccessful because it left many aspects unresolved and forced to integrate many other tools for monitoring and management. The best solution we have decided to adopt has led to the creation of a platform for the exposure of collaboration services, structured so as to fully exploit the microservice architecture made available through the company's Private Cloud. In this way it is possible to expand the range of services provided by the platform itself simply by adding microservices to the general architecture. In addition, it performs automatic scaling and centralized management, directly from the OpenShift console. The services are no longer linked to a single technology or programming language, but it is possible to use all those authorized by the OpenShift platform itself according to the rules established in the company. Whenever an application needs to interface with a collaboration tool, it will do so using the new platform. It will no longer be necessary for the developer to know the APIs of each product, nor all the endpoints present, but only one platform-related endpoint will be available. If the company de-

cides, for example, to replace instant messaging rather than email software, the APIs exposed by the new platform will remain the same, avoiding the work of updating all the applications involved. With this solution it is also possible to add functionality in the provision of collaboration services, such as security functions or pipelines connected to other services outside of ECS. The potential offered by this solution is many and thanks to the flexibility that comes from the underlying technology of containers, it will be possible in the future to fill all the gaps in the world of ECS and beyond. Since ECS has now become the center of the universe to which all corporate applications and workflows belong, with this new platform the company finally has the possibility of making all its application ecosystem homogeneous and centrally managed.

In conclusion, there are no simple solutions to solve the problems that are encountered interfacing with the world of collaboration. The more time passes and the more it becomes indispensable, being an enabler of smart working and remote communication. On the other hand, the new cloud technologies make a big contribution to creating ad hoc, flexible and secure solutions, making it easier to manage the ECS. In addition, the market already presents frameworks designed for the integration of the many collaboration products present in the company, highlighting the awareness of the importance that this system has acquired.

This project allowed us to experience for the first time in a complete way the creation of a microservice infrastructure created solely by cloud. So many challenges have been faced on this front, which have allowed us, however, to create a consistent experience that will be added to the know-how already present in the company and that will certainly be reused for all the other applications that are part of the banking ecosystem of Intesa Sanpaolo.

References

- [1] Jörg Becker, Oliver Vering, and Axel Winkelmann. *Softwareauswahl und-einführung in Industrie und Handel: Vorgehen bei und Erfahrungen mit ERP-und Warenwirtschaftssystemen; mit 21 Tabellen*. Springer, 2007.
- [2] Thomas H Davenport. “Putting the enterprise into the enterprise system”. In: *Harvard business review* 76.4 (1998).
- [3] Roland Diehl, Tim Kuettner, and Petra Schubert. “Introduction of enterprise collaboration systems: In-depth studies show that laissez-faire does not work”. In: (2013).
- [4] Paul Dourish. “The appropriation of interactive technologies: Some lessons from placeless documents”. In: *Computer Supported Cooperative Work (CSCW)* 12.4 (2003), pp. 465–490.
- [5] Arron Fu. *7 Different Types of Cloud Computing Structures*. Mar. 2017. URL: <https://www.uniprint.net/en/7-types-cloud-computing-structures/>.
- [6] IBM. *Using IBM Social Business to Take Your Business Relationships to the Next Level: A Game Changer for Small, Medium, and Large Businesses*. Mar. 2014.
- [7] David Kiron et al. “What managers really think about social business”. In: *MIT Sloan Management Review* 53.4 (2012), p. 51.
- [8] Tim Kuettner, Roland Diehl, and Petra Schubert. “Change factors in Enterprise 2.0 initiatives: Can we learn from ERP?” In: *Electronic Markets* 23.4 (2013), pp. 329–340.
- [9] Jessica Lipnack and Jeffrey Stamps. “Virtual teams”. In: *People working across boundaries with technology* 2 (2000).
- [10] Goran Markovski, Natasa Koceska, and Saso Koceski. “Improving enterprise efficiency using IT collaboration systems”. In: *Journal of Applied Economics and Business* 1.4 (2013), pp. 95–103.
- [11] M Lynne Markus and Cornelis Tanis. “The enterprise systems experience from adoption to success”. In: *Framing the domains of IT research: Glimpsing the future through the past* 173 (2000), pp. 207–173.
- [12] Gabriele Piccoli, Anne Powell, and Blake Ives. “Virtual teams: team control structure, work processes, and team effectiveness”. In: *Information Technology & People* 17.4 (2004), pp. 359–379.

- [13] Alexander Richter and Alexander Stocker. “Exploration & Promotion: Einführungsstrategien von Corporate Social Software”. In: *Proceedings of the 10th International Conference on Wirtschaftsinformatik*. 2011, pp. 1114–1123.
- [14] Kai Riemer and Robert Bruce Johnston. “Place-making: A phenomenological theory of technology appropriation”. In: (2012).
- [15] Art Schoeller. “Gain A Competitive Advantage Through Enterprise Collaboration”. In: *Forrester* (2017).
- [16] Petra Schubert and Johannes H Glitsch. “Use Cases and Collaboration Scenarios: How employees use socially-enabled Enterprise Collaboration Systems (ECS)”. In: *Int. J. Inf. Syst. Proj. Manag* 4.2 (2016), pp. 41–62.
- [17] Petra Schubert and Susan P Williams. “The Concept of Social Business: Oxymoron or Sign of a Changing Work Culture?” In: *Bled eConference*. 2013, p. 26.
- [18] Radicati Team. *Email Statistics Report, 2016-2020. The Radicati Group*. 2016.
- [19] *What is a Container – A standardized unit of software*. URL: <https://www.docker.com/resources/what-container>.
- [20] *What is cloud computing? – A beginner’s guide*. URL: <https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/>.
- [21] *Why a microservices approach to building applications?* URL: <https://docs.microsoft.com/en-us/azure/service-fabric/media/service-fabric-overview-microservices/>.
- [22] Susan P Williams. “Enterprise 2.0 and collaborative technologies”. In: *Koblenz: Working Report of the Research Group Business Software, University of Koblenz-Landau* (2011).
- [23] Susan P Williams, Petra Schubert, et al. “An empirical study of enterprise 2.0 in context”. In: *Bled eConference*. 2011, p. 44.