

Cloud Computing with an emphasis on Google App Engine

Master Final Project

Author: Adam Gedymin

Academic Advisor: Xavier Franch Gutiérrez

Facultad de Informática de Barcelona
Máster en Tecnologías de la Información

September 2011

Index

1	Introduction	7
1.1	Project Context.....	7
1.2	Motivation	8
1.3	Project Objectives	8
1.4	Document Structure.....	9
2	Description and Analysis of Cloud Computing	10
2.1	Deployment Models	13
2.1.1	Private Cloud	13
2.1.1.1	Virtual Private Cloud (PVC).....	15
2.1.1.2	Private Cloud in Practice	16
2.1.2	Public Cloud.....	18
2.1.2.1	Migration.....	20
2.1.2.1.1	Requirements Specification	20
2.1.2.1.2	Assessing Security and Privacy Risks	21
2.1.2.1.3	Assessing the Competency of the Cloud Provider	21
2.1.3	Hybrid Cloud.....	22
2.1.4	Community Cloud.....	24
2.2	Service Models	25
2.2.1	SaaS (Software as a Service)	27
2.2.1.1	Benefits.....	29
2.2.1.2	Drawbacks	30
2.2.1.3	Main providers	31
2.2.1.3.1	Google	31
2.2.1.3.2	Salesforce.com	31
2.2.1.4	Case Studies.....	32
2.2.1.4.1	BBVA.....	32
2.2.1.4.2	IRB Barcelona	33
2.2.2	PaaS (Platform as a Service)	33
2.2.2.1	Benefits.....	36
2.2.2.2	Drawbacks	37

2.2.2.3	Main Providers	38
2.2.2.4	Case Study	38
2.2.2.4.1	MenuMate.....	38
2.2.3	IaaS (Infrastructure as a Service).....	40
2.2.3.1	Benefits.....	41
2.2.3.2	Drawbacks	42
2.2.3.3	Main Providers	43
2.2.3.3.1	Amazon	43
2.2.3.3.2	Joyent	44
2.2.3.3.3	Layered Technologies (Layered Tech)	45
2.2.3.3.4	Terremark.....	45
2.2.3.3.5	GoGrid	45
2.2.3.3.6	VMware	45
2.2.3.3.7	AT&T.....	46
2.2.3.3.8	Rackspace	46
2.2.3.4	Case study	46
2.2.3.4.1	Oil & Gas SME [37]	46
2.3	Summary	50
3	Comparison of Different PaaS solutions	52
3.1	PaaS Architecture	53
3.1.1	iPaaS (Integration Platform as a Service)	54
3.1.2	dbPaaS (Database as a Service).....	55
3.1.3	bpmPaaS (Business Process Management as a Service).....	55
3.1.4	Cloud Foundation	56
3.1.5	Performance Foundation	57
3.2	PaaS Providers Description	59
3.2.1	Windows Azure	59
3.2.2	Google App Engine	60
3.2.3	AWS Elastic BeanStalk	60
3.2.4	Force.com	61
3.2.5	Heroku	61
3.2.6	CloudFoundry	61

3.2.7	OpenShift.....	62
3.3	Comparative Table	63
3.3.1	Possible Deployments	63
3.3.2	Programming Language Frameworks.....	64
3.3.3	Developer Tools.....	65
3.3.4	Backend Infrastructure.....	65
3.3.5	Persistence Options.....	65
3.4	Private PaaS.....	68
3.5	Summary	70
4	Google App Engine	71
4.1	Technology Support	72
4.2	Google Database	75
4.2.1	Datastore.....	76
4.2.1.1	Data Repository	78
4.2.1.2	Datastore Interfaces.....	79
4.2.2	Cloud SQL	80
4.2.2.1	Features.....	80
4.2.2.2	Restrictions.....	80
4.3	Integrated Services.....	82
4.3.1	App Identity.....	82
4.3.2	Blobstore	83
4.3.3	Google Cloud Storage.....	83
4.3.4	Capabilities	84
4.3.5	Channel.....	84
4.3.6	Conversion.....	84
4.3.7	Images	84
4.3.8	Log Service.....	84
4.3.9	Mail.....	84
4.3.10	Memcache	85
4.3.11	Multitenancy	85
4.3.12	OAuth	85
4.3.13	Prospective Search	86

4.3.14	Search	86
4.3.15	Task Queues	86
4.3.16	URL Fetch	87
4.3.17	Users	87
4.3.18	XMPP	87
4.3.19	App Engine Cron Service	88
4.4	Tools	89
4.4.1	Development Server	89
4.4.2	Datastore Viewer	90
4.4.3	Task Queues	90
4.4.4	XMPP	91
4.4.5	Inbound Mail	91
4.4.6	Backends	91
4.4.7	Capabilities Status	92
4.4.8	Google Plugin for Eclipse	92
4.4.9	Local Unit Testing	93
4.4.10	Appstats	93
4.5	Limits, Quotas & Billing	94
4.6	Summary	96
5	Pilot Application	97
5.1	Project Strategy	98
5.1.1	Objective and Scope of the Pilot Application	98
5.1.2	Technology and Motivation	98
5.2	Requirement Management	99
5.2.1	Obtaining the Requirements	99
5.2.2	Non-Functional Requirements	100
5.3	Functional Design	102
5.3.1	Use Case Diagram	102
5.3.2	Functional Requirements	103
5.3.3	Conceptual Data Model	112
5.3.3.1	Conceptual Model Class Description	113
5.3.4	Navigation Map	115

5.3.5	User Interface Design	116
5.3.5.1	User Interface of the Default Site of the Application.....	117
5.3.5.2	User Interface of the Account Site	118
5.3.5.3	Chat and loan User Interface.....	119
5.3.6	Logical Architecture.....	120
5.3.7	Used Technologies	122
5.4	Technical Design.....	123
5.4.1	Sequence Diagrams	123
5.4.1.1	Show Blog News Use Case.....	123
5.4.1.2	Chat Use Case	124
5.4.1.3	Add New Account Use Case	125
5.4.2	Pilot Application's File Structure.....	126
5.4.3	Specification of the User Interface.....	128
5.5	Project Evaluation	130
5.5.1	Identified Obstacles.....	130
5.5.1.1	Running JavaServer Faces on Google App Engine.....	130
5.5.1.2	Communication with GAE Datastore	130
5.5.1.3	Application Performance.....	131
5.5.1.4	Serialization Issues	132
5.5.1.5	Memcache.....	132
5.5.2	Further Observations	133
5.5.2.1	Tests	133
5.5.2.2	Chat Implementation	133
6	Planning and Economic Study of the Project	135
6.1	Project Planning	135
6.1.1	Initial planning.....	135
6.1.2	Actual planning.....	137
6.2	Economic Study	139
7	Conclusions	141
8	Bibliography	143

1 Introduction

In an attempt to gain a competitive advantage, businesses are increasingly looking for innovative ways to minimize the costs while maximizing value – especially now, during the time of financial crisis. Organizations realize that they need to grow, but at the same time they are urged to save money. Such tendency has led to growing acceptance of innovative technologies and boom in cloud computing. Hence this process caused situation common to many innovations and new technologies i.e. common understanding of cloud computing is continuously evolving, thus the terminology and concepts necessary for defining it often need clarifying.

However, before an organization decides to migrate to the cloud, it is crucial to realize what should be done, and what provider should be chosen. Not all cloud computing providers are the same. The range and quality of offered services varies extremely, so it is recommended to investigate the market thoroughly, with a clearly defined set of requirements in mind.

1.1 Project Context

This project was developed as a part of an agreement between the Master in Information Technologies (MIT) of the Faculty of Informatics in Barcelona (FIB) of the Technical University of Catalonia (UPC) and the company Everis. The project has been developed on premises of Everis, under full-time contract with the objectives of: putting into practice the knowledge gained during the studies in Master program, conducting an in-depth research of cloud computing, and testing Google App Engine – the cloud platform of Google.

Everis is a multinational consulting company providing its services (business, strategy, and development) to the organizations from the following sectors: Telecommunication, Finance, Energy & Utilities, Banking, Insurance, Public Administration, Media, Business, and Health. Currently Everis operates in many countries from Europe, United States, and South America, and hires over 10,000 employees.

Below is presented the office distribution of Everis' offices.

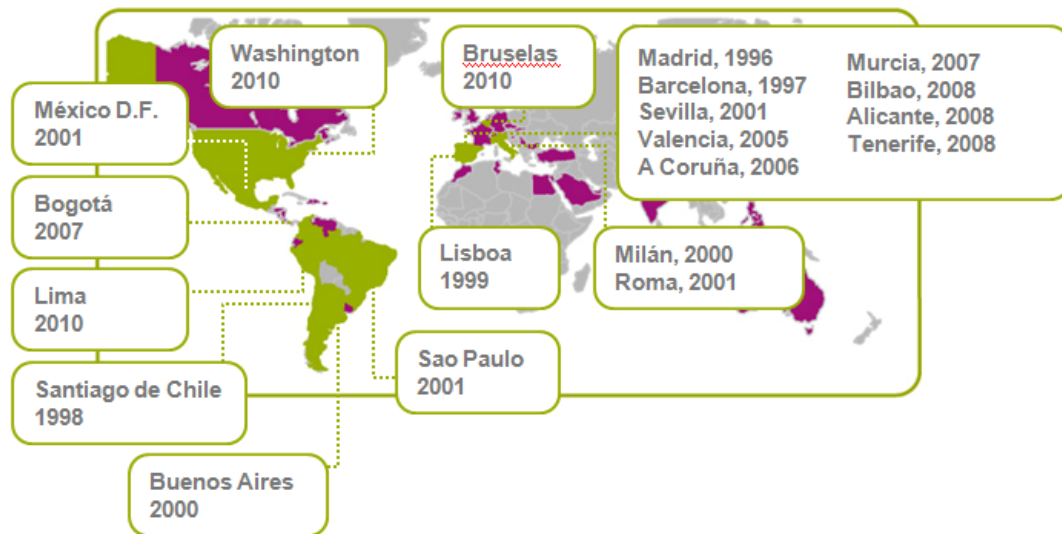


Figure 1.1 Everis' office distribution

1.2 Motivation

One of many approaches to exploit new technologies within the area of innovation TEC in Everis is through hiring students willing to realize their final projects in the company. These students with help of a senior employee conduct market analysis, make product comparisons, or develop sample applications. Thus such collaboration between the intern and the company is very convenient as many of those tasks require many hours of work which Everis employees might not have.

Nowadays, Everis encounters itself in a very difficult time, when more and more companies decide to move into cloud, finding it as a cost-cutting solution. In order to continue providing an innovative approach Everis has to be able to offer what its customers want i.e. innovative cloud solutions. Hence, the company wants to specialize in PaaS (platform as a service), and as such it was decided to dedicate the internship for such purposes.

1.3 Project Objectives

The following are to objectives given by Everis in for successful accomplishment of the project:

- To describe Cloud Computing and its elements in general.
- To identify leading providers of Platform as a Service cloud and to compare them, and their solutions.

- To describe in detail Google's PaaS platform, its tools, environment, and functionalities.
- To develop a small pilot application in order to test against some of the Google's integrated APIs and discover the way they function.

1.4 Document Structure

- Chapter 2 - Definition of cloud computing and its deployment and service models. Some case studies were presented to facilitate understanding of each.
- Chapter 3 - In-depth definition of Platform as a Service cloud and its structure, comparison of leading public PaaS providers and their solutions.
- Chapter 4 – Specification of Google App Engine and its components. The Google's platform was given a deeper look and there was explained Google's approach to cloud, along with billing information and how to operate within the local environment.
- Chapter 5 – Description of the pilot application following the Everis' methodology.
- Chapter 6 – Project planning with Gantt diagrams showing the initial and final planning of the project. Also economic study was included in that chapter.
- Chapter 7 – Observations concluded once the project is finished.

2 Description and Analysis of Cloud Computing

The roots of Cloud Computing can be found in the advancement of new technologies in the areas of:

- Hardware, i.e. virtualization, multi-core chips
- Internet Technologies, i.e. Web services, service-oriented architectures, Web 2.0
- Distributed Computing, i.e. clusters, grids
- Systems Management, i.e. autonomic computing, data center automation

The emergence of cloud computing itself is closely linked to the maturity of the above-mentioned technologies.

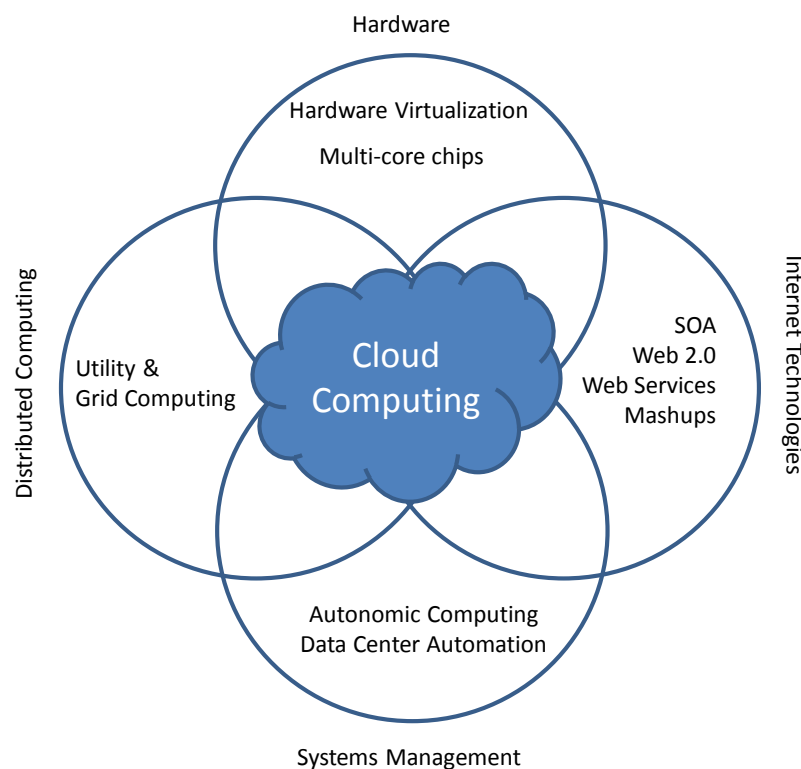


Figure 2.1 Convergence of various technologies leading to creation of cloud computing

Cloud Computing is a relatively new model of providing IT services that can seriously reduce costs and complexity of the IT infrastructure. Thus it can have a decent impact on the improvement of IT services. This model is highly scalable, convenient for its users, and feasible to apply for specific economic circumstances. The development of business virtualization and possibilities of conducting complex computations in clouds have given a unique opportunity for transformation of the companies' information resources from centers of cost generation to

an important strategic factor of the economic organ. NIST¹ (American National Institute of Standards and Technology) - defines the cloud model as a composition of five essential characteristics, three service models, and four deployment models. Thus NIST describes them the following way [21].

- **On-demand self-service.** A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
- **Broad network access.** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
- **Resource pooling.** The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.
- **Rapid elasticity.** Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
- **Measured service.** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Service Models identified by NIST are the following [21]:

- **Software as a Service (SaaS).** The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers,

¹ NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets; but such standards and guidelines shall not apply to national security systems

operating systems, storage, or even individual application capabilities, with the possible exception of limited user specific application configuration settings.

- **Platform as a Service (PaaS).** The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.
- **Infrastructure as a Service (IaaS).** The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

According to NIST the Deployment Models are the ones described below [21]:

- **Private cloud.** The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.
- **Community cloud.** The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.
- **Public cloud.** The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.
- **Hybrid cloud.** The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

2.1 Deployment Models

Although cloud computing has emerged mainly from the appearance of public computing utilities, other deployment models, with variations in physical location and distribution, have been adopted [24].

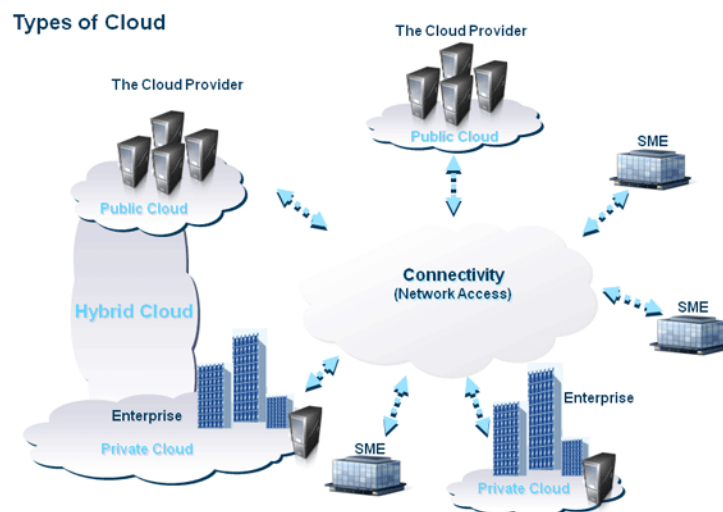


Figure 2.2 Cloud Deployment Models [24]

2.1.1 Private Cloud

A private cloud provides a single organization the exclusive access, and usage of the infrastructure and computational resources. Such cloud can be managed by the organization itself or by a third party. Thus it can be hosted on the organization's premises (i.e. on-site private clouds) or outsourced to a hosting company (i.e. outsourced private clouds).

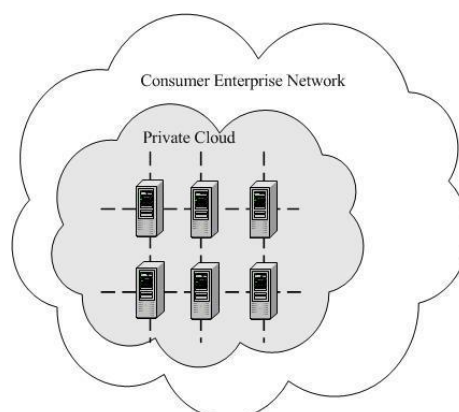


Figure 2.3 Private Cloud [27]

Although private cloud has attracted criticism like no benefit from lower up-front cost and less hands-on management, it still provides for some, many benefits such as securing the private

data. Generally, private cloud is deployed on an enterprise's data center, which is located behind firewalls, and it can also be deployed on a safe hosting place. The main issue is operational complexity, since the environment is hosted and managed by internal resources environment.

At present, the most discussed type of cloud is public cloud, but security is one of the major problems in this type of clouds. The users have to trust the cloud service provider. However, many corporations cannot accept that their important and confidential data be placed in public cloud. On the contrary, private cloud gives users a flexible and agile private infrastructure to run service workloads within their administrative domains [23]. In case of considering private cloud implementation below is the table pointing out four primary considerations [24].

Consideration	Rationale
Security	Applications that require direct control and custody over data for security or privacy reasons
Availability	Applications that require certain access to a defined set of computing resources that cannot be guaranteed in a shared resource pool environment
User Community	Organization with a large number of users, perhaps geographically distributed, who need access to utility computing resources
Economies of Scale	Existing data center and hardware resources that can be used, and the ability to purchase capital equipment at favorable pricing levels

Table 2.1. Four primary private cloud considerations

Many companies struggle with undertaking a final choice whether to deploy a private cloud. The main concerns are the following:

- **Small Scale of private clouds** – volume drives costs down through the huge economies of scale. The bigger the cloud, the bigger are savings.
- **Legacy Applications** – when moved to a private cloud will see marginal improvements at best.
- **On-site is not necessarily more secure** – public cloud providers such as Amazon or Google spend billions of dollars for security of their datacenters. Thus private clouds will always be behind public clouds in that sense.

If, despite the above concerns, a company decides to build a private cloud, there are several options for doing so. Possible private cloud implementation categories and example solutions are shown in the table below [24].

Provider Type	Example Vendors	Description
Open Source	Eucalyptus, OpenNebula	Free software for implementing a private cloud on UNIX-based systems
Proprietary Software	VMware, RedHat, Appistry	Proprietary private clouds main benefits are: <ul style="list-style-type: none"> • Virtualization • Storage • Management
Hosted	Savvis, OpSource, SunGard	Dedicated hardware hosted in a cloud model for a single customer, built using either open source or a proprietary solution
System integrator	Appirio, Accenture, Infosys	Specialty providers or practice

Table 2.2 Private Cloud deployment options by type

2.1.1.1 Virtual Private Cloud (PVC)

The virtual private cloud was first created by Amazon. It connects a data center to Amazon's EC2². This provides a solution to a situation where the traffic exceeds on-premise capacity. In such event the Amazon EC2 instances add additional web-facing servers to the application. Whenever the demand subsides, the amazon EC2 that are no longer required can be terminated. This is cloud bursting³ even though Amazon itself does not call it that way. It is also worth mentioning that Google has a similar structure called Secure Data Connector, which connects legacy infrastructure to Google's App Engine PaaS public cloud.

² **Amazon Elastic Compute Cloud (EC2)** is a central part of Amazon.com's cloud computing platform, Amazon Web Services (AWS). EC2 allows users to rent virtual computers on which to run their own computer applications.

³ **Cloudbursting** is an application deployment model in which an application runs in a private cloud or data center and bursts into a public cloud when the demand for computing capacity spikes. The advantage of such a hybrid cloud deployment is that an organization only pays for extra compute resources when they are needed

2.1.1.2 Private Cloud in Practice

Below is being described a case study based on the Bechtel company with a purpose to familiarize a reader with private cloud computing.

Bechtel Project Services Network (PSN)

Bechtel is a big company with over 40,000 employees. The company is active in 50 countries worldwide. The CIO of Bechtel, Geir Ramleth in 2006 decided to transform its IT department and model it by following main Internet enterprises such as YouTube, Google, Amazon.com and Salesforce.com. The cause of such action was a study conducted by Bechtel. The study results pointed out the following. Eventually, Bechtel turned itself into a software-as-a-service (SaaS) provider for internal users, subcontractors and business partners. Bechtel came up with estimates for how much money YouTube spends on networking Google system, administrations amazon.com on storage, and salesfore.com on software maintenance [25].

- Bechtel estimated that YouTube spent \$10 to \$15 per megabit for bandwidth, while Bechtel is spending \$500 per megabit for its Internet-based VPN.
- Bechtel estimated that Google used 12 system administrators for every 200,000 servers, or roughly 17,000 servers per system administrator. Bechtel, on the other hand, was operating with 1,000 servers per system administrator.
- While Amazon.com was offering storage for 10 cents per gigabyte per month, Bechtel's internal rates in the United States was \$3.75 per gigabyte.
- Salesforce.com has only one version of its application servicing 1 million users, which it upgrades four times a year with little downtime and few training requirements. In comparison, Bechtel used 230 different applications with up to 5 versions each, amounting to almost 800 different versions of applications servicing 40,000 employees.

Bechtel scrapped its existing data centers and built three new facilities featuring the latest in server and storage virtualization at a time. Bechtel also designed a new Gigabit Ethernet network with hubs at Internet exchange points that it is managing itself instead of using carriers. A Gigabit Ethernet ring is connecting the three new data centers, with dual paths for failover. Bechtel is buying raw bandwidth from a variety of providers -- Cox, AboveNet, Qwest, Level 3 and Sprint. Bechtel built the portal using Microsoft's SharePoint software as Ramleth stated the portal gives them consumerization⁴ and the new security model [25]. Now, Bechtel

⁴ **Consumerization [Wikipedia]** is an increasingly accepted term used to describe the growing tendency for new information technology to emerge first in the consumer market and then spread into business and government organizations.

is slashing its portfolio of software applications to simplify operations as the end user experience.

Transforming the IT infrastructure into a private cloud saved 25 to 30 percent in overall IT costs.

2.1.2 Public Cloud

For last couple of years the public-cloud-computing market has grown tremendously. Differently to private clouds, used internally, public cloud platforms are available to virtually anyone with a credit card. Thus customers of that platform can take advantage of hundreds of virtual machines in a few minutes, and pay only for what they will have used, with no up-front investment. Such flexibility drew attention of organizations from many industries including, governments, schools, enterprises, and content providers [26].

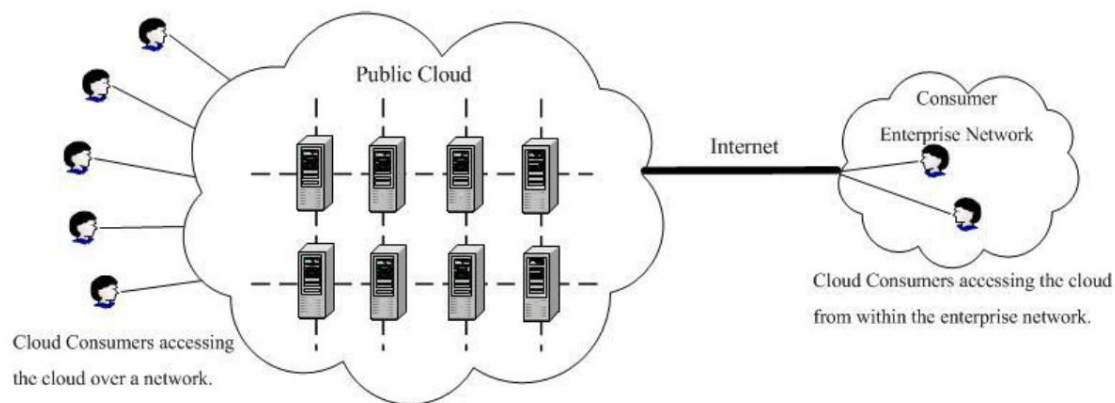


Figure 2.4 Simple view of public cloud and its users [27]

As it can be observed in the figure above, cloud infrastructure and computing resources are made available to the general public over a public network. Such infrastructure is owned by an organization selling cloud services, and serves a diverse pool of clients.

Public cloud could have unpredictable tenants co-existing with each other; therefore, workload isolation is less of a security concern in a private cloud than in a public cloud [27].

Although many organizations have doubts about moving to public cloud mainly because they fear for their security and privacy, some of them could profit in that area as well. Thus potential areas of improvement are pointed out as the following [28].

- **Staff specialization** - Increases in the scale of computing induce specialization, which in turn allows security staff to shed other duties and concentrate exclusively on security and privacy issues.
- **Platform Strength** - The structure of cloud computing platforms is typically more uniform than that of most traditional computing centers. Greater uniformity and homogeneity facilitate platform hardening and enable better automation of security management activities like configuration control, vulnerability testing, security audits, and security patching of platform components.

- **Resource Availability** - Redundancy and disaster recovery capabilities are built into cloud computing environments and on-demand resource capacity can be used for better resilience when faced with increased service demands or distributed denial of service attacks, and for quicker recovery from serious incidents. Availability can also bolster privacy through better opportunities for individuals to access and correct records and for records to be ready for use when needed for the purposes collected.
- **Backup and Recovery** - The backup and recovery policies and procedures of a cloud provider typically are superior to those of the organization and also they are more robust. Data maintained within a cloud happens to be more available, faster to restore, and more reliable in many circumstances than that maintained in a traditional data center, and also meet offsite backup storage and geographical compliance requirements.
- **Mobile Endpoints** - Since the main computational resources needed by cloud-based applications are typically held by the cloud provider, clients can generally be lightweight computationally and easily supported on laptops, notebooks, and netbooks
- **Data Concentration** - Data maintained and processed in a public cloud may present less of a risk to an organization with a mobile workforce than having that data dispersed on portable computers, embedded devices, or removable media out in the field, where theft and loss routinely occur.

Despite all the above-mentioned benefits, public clouds have their security and privacy downsides as well. Some of the most critical concerns include the following [28]:

- **System Complexity** – A public cloud infrastructure compared with traditional data center is extremely complex. Security depends not only on the correctness and effectiveness of many components, but also on the interactions among them.
- **Shared Multi-tenant Environment** — client organizations typically share components and resources with other consumers that are unknown to them. Thus sharing an infrastructure with unknown outside parties can be a major drawback for some applications and require a high level of assurance pertaining to the strength of the security mechanisms used for logical separation.
- **Internet-facing Services** – After transforming to public cloud organizations have to face a new threat from network. Such threat previously was not an issue as data was defended by the organization's intranet.

- **Loss of control** - Transitioning to a public cloud requires a transfer of responsibility and control to the provider over information as well as system components that were previously under the organization's direct control. The transition is usually accompanied by the lack of a direct point of contact with the management of operations and influence over decisions made about the computing environment. This situation makes the organization dependent on the cooperation of the cloud provider to carry out activities that span the responsibilities of both parties, such as continuous monitoring and incident response.

2.1.2.1 Migration

Migration is one of most if not the most important aspect of public cloud computing. A general way of how to proceed when it comes to transfer the IT infrastructure to a public cloud can be seen as specified by NIST [28].

2.1.2.1.1 Requirements Specification

When migrating to public cloud, organization needs to identify the requirements for cloud service, which will be the criterion for the selection of a cloud provider [28].

- Personnel requirements, including clearances, roles, and responsibilities
- Regulatory requirements
- Service availability
- Problem reporting, review, and resolution
- Information handling and disclosure agreements and procedures
- Physical and logical access controls
- Network access control, connectivity, and filtering
- Data protection
- System configuration and patch management
- Backup and recovery
- Data retention and sanitization
- Security and vulnerability scanning
- Risk management
- Incident reporting, handling, and response
- Continuity of operations
- Resource management
- Certification and accreditation
- Assurance levels
- Independent auditing of services.

During the requirements analysis an organization will narrow the choice among IaaS, PaaS, and SaaS to one that is appropriate for the organization's needs.

Also it is very important to establish an exit strategy that should be factored into the analysis of requirements. The possibility to export the organization's data in a usable format through a secure, reliable and efficient means, and in a timely manner, is crucial for defining such strategy.

2.1.2.1.2 Assessing Security and Privacy Risks

The risk analysis should include factors such as the service model, purpose and scope of the service, types and levels of access to the new infrastructure, the service duration, dependencies, and the strength of protection provided by the cloud provider.

Moreover, it is crucial for conducting an accurate risk analysis to understand the underlying technologies the cloud provider uses to provision services.

2.1.2.1.3 Assessing the Competency of the Cloud Provider

Before contracting for outsourcing services, the cloud provider's ability and commitment to deliver the services over the target timeframe and meeting the security and privacy levels should be evaluated. Such items as the following should also be given a consideration [28].

- Experience and technical expertise of personnel
- The vetting process personnel undergo
- Quality and frequency of security and privacy awareness training provided to personnel
- Account management practices and accountability
- The type and effectiveness of the security services provided and underlying mechanisms used
- The adoption rate of new technologies
- Change management procedures and processes
- The cloud provider's track record
- The ability of the cloud provider to meet the organization's security and privacy policy, procedures, and regulatory compliance needs.

2.1.3 Hybrid Cloud

Hybrid cloud computing is a platform that interoperates between private cloud and public cloud. It is deployed by organizations, which do not want to put everything in the external cloud (public cloud) while hosting some servers in their own internal cloud infrastructure. The cloud providers are able to process applications, which can work seamlessly between those boundaries [29].

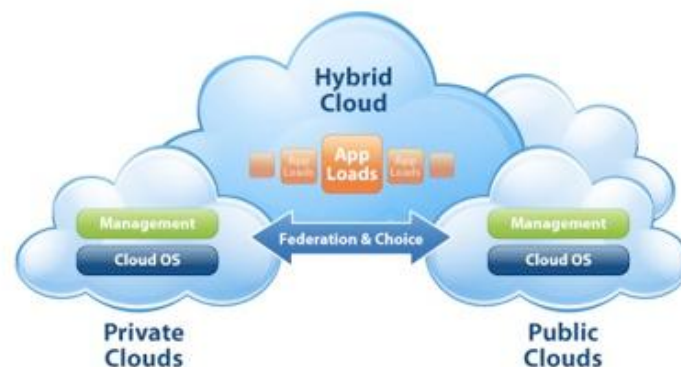


Figure 2.5 Hybrid cloud computing [29]

In a case where the public cloud fails to handle an application, the request can be forwarded to the private cloud as shown in the figure 2.5. The hybrid cloud validates the fact that not all information technology resources should remain in the public cloud today. When considering the security restrictions and the performance, the need of a private cloud is a fact today. Enterprises have to know, which kind of data can be kept locally and what can be processed remotely [29].

Hybrid Cloud in comparison to other deployment types, demands additional and specific functionalities that have to be considered while designing software systems supporting the execution of applications in hybrid and dynamic environments. These features according to NIST, together with some guidelines on how to implement them are the following [31]:

- **Heterogeneity Support.** Hybrid clouds are produced heterogeneously by resources such as clusters, public or private virtual infrastructures, and workstations. In particular, the biggest concern of a virtual machine manager, it must be possible to integrate additional cloud service providers (mostly IaaS providers) without major changes to the entire system design and codebase. Thus, the specific code related to a particular cloud resource provider should be kept isolated behind interfaces and within pluggable components.

- **Dynamic and Open Systems Support.** Over time the composition and topology change in Hybrid clouds. They form as a result of constantly changing conditions such as peak demands or specific Service Level Agreements attached to the applications are currently executed. An open and extensible architecture that allows for easy plugging new components and rapidly integrating new features is of a great value in this case. Specific enterprise architectural patterns can be considered when designing such software systems. In particular, inversion of control and dependency injection in component-based systems is really helpful.
- **Basic VM Operation Management Support.** Hybrid clouds integrate virtual infrastructures with existing physical systems. Virtual infrastructures are produced by virtual instances. Hence, software frameworks that support hypervisor-based execution should implement a minimum set of operations. They include requesting a virtual instance, controlling its status, terminating its execution, and keeping track of all the instances that have been requested.
- **Flexible Scheduling Policies Support.** The heterogeneity of resources that compose a hybrid infrastructure naturally demands for flexible scheduling policies. Public and private resources can be differently utilized, and the workload should be dynamically partitioned into different streams according to their security and quality of service (QoS) requirements. There is also the need of being able to transparently change scheduling policies over time with a minimum impact on the existing infrastructure and almost now downtimes. Thus configurable scheduling policies are an important feature.
- **Workload Monitoring Support.** Workload monitoring becomes even more important in the case of hybrid clouds where a subset of resources is leased and resources can be dismissed if they are no longer necessary. Workload monitoring is an important feature for any distributed middleware, in the case of hybrid clouds, it is necessary to integrate this feature with scheduling policies that either directly or indirectly govern the management of virtual instances and their leases.

One most common case where the hybrid approach sounds beneficial is when making sure the performance is unaffected in case of sudden or unexpected load spikes. Instances of the same application can reside on both private and public cloud. A cloud integration manager directs the request to the public cloud when internal infrastructure underlying the private cloud fails to handle the increased load. This is sometimes called cloud bursting and it enables for the limited usage of the massively scalable environment of public clouds. At the same time, it also allows enterprises to use their existing infrastructure.

2.1.4 Community Cloud

Community Cloud is somewhat similar to a private cloud, but the infrastructure and computational resources are exclusive to two or more organizations that have common privacy, security, and regulatory considerations, rather than a single organization [28]. One example of this is OpenCirrus formed by HP, Intel, Yahoo, and others.

The idea of community cloud aroused from concerns over Cloud Computing, in particular control by vendors and lack of environmental sustainability. Thus by the concept itself is to replace vendor Clouds by shaping the under-utilized resources of user machines to form a Community Cloud with each node having a potential to fulfill all roles, consumer (green), producer (yellow) and coordinator (red). Such cloud can be represented as in the following figure [30].

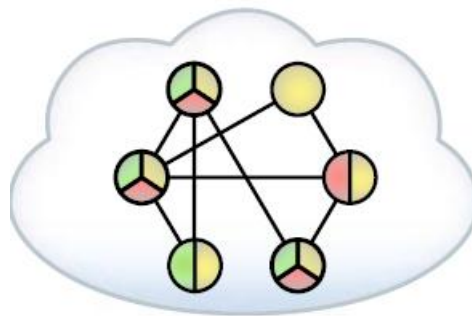


Figure 2.6 Community Cloud [30]

It is necessary for at least one community member to provide cloud services for a community cloud to be functional. The figure depicts members that provide cloud services (and possibly consume them also). Assuming that each organization implements a security perimeter, the participant organizations are connected via links between the boundary controllers that allow access through their security perimeters. Optionally, organizations may implement extra security perimeters to isolate the local cloud resources from other local resources. Many network configurations are possible.

A paradigm for community cloud without dependence on Cloud vendors, such as Amazon, Microsoft or Google can be described by the factors below.

- Openness – no dependency on vendors make
- Community
- Individual Autonomy
- Graceful Failures
- Convenience and Control
- Community Concurrency
- Quality of Service
- Environmental Sustainability

2.2 Service Models

For many years now, cloud computing has been developing unknowingly. Dividing a cloud into service models is something very recent, though one of those models has been used for a very long time. That model is SaaS (Software as a Service), which was present almost since the beginning of internet along with online email client. The other model – IaaS - is not conceptually new, as people have been collocating in data centers since data centers have been around. What is different about it though is the tooling behind it. Proper IaaS platform should provide a mechanism to replace all of the data center hardware needs. Unlike IaaS, PaaS is a much more abstract concept, and it provides a web development platform for others to use. Taking a look at a Cloud Computing concept as a stack, PaaS would be in the middle of that stack, with IaaS at the bottom and SaaS at the top. Such representation is shown in the figure below.

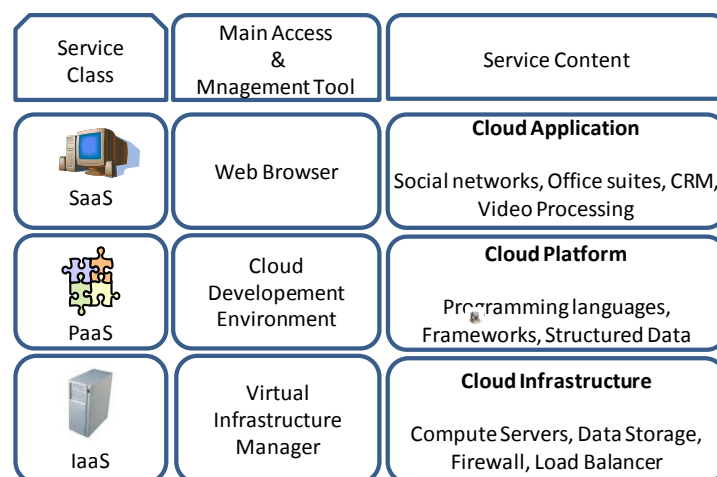


Figure 2.7 Cloud Computing Stack Representation

During the last decade, main providers of cloud computing platforms were formed along with cloud computing development. Following the above division, another distinction of providers could be pointed out as in the figure below.















<i>Software as a Service (SaaS)</i>	    
<i>Platform as a Service (PaaS)</i>	    
<i>Infrastructure as a Service (IaaS)</i>	   

Figure 2.8 Cloud stack and leading providers

2.2.1 Saas (Software as a Service)

In reality the term SaaS dates from the 1990s and thus it predates the term cloud computing itself [31]. Thus email clients such as Gmail or Hotmail, and many different software solutions accessible over Internet, could be given as an example of SaaS, that is software offered as a service. This leads to a most descriptive definition of SaaS which is “Software deployed as a hosted service and accessed over the Internet”. Below is the table pointing out the main features of SaaS platform [31].

Software as a Service	
The consumers	<ul style="list-style-type: none"> • Organizations providing their employees/members with access to office applications such as email • Direct users using the software applications on their own behalf or on the behalf of their organization • Application administrators that configure an application for end users
Consumer Acquires	<ul style="list-style-type: none"> • Right to use specific applications on demand • Application data management i.e. backup and data sharing between consumers.
Fees Calculation	Normally the fees are based on the number of users, the time of usage, per-execution, per-record-processed, network bandwidth consumed, and quantity/duration of data stored

Tabla 1.3 SaaS general features

SaaS can be seen as a Platform for renting access to an application. In order to take a closer look at consumer/producer interaction dynamics the following figure will serve as a reference [32].

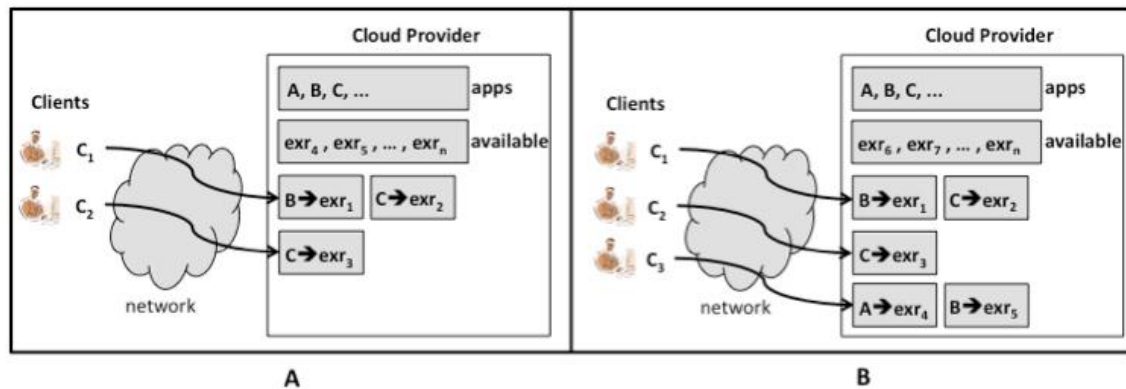


Figure 2.9 SaaS Consumer/Provider Interaction Dynamics [31]

Figure 2.9 A. represents a cloud providing services to two clients, C₁ and C₂. In a private cloud, the clients will belong to (or be associated with) a single consumer organization; in other deployment models the clients may represent different consumers. Abstractly, the cloud provider possesses a set of software applications ("apps" in the figure) that it is offering to the clients for use over the network. Moreover, the cloud provider manages application execution resources ("exr" in the figure). In Figure 2.9 A, client C₁ is currently using two applications, B and C. To execute the apps for client C₁, the cloud provider has allocated two execution resources, exr₁ and exr₂, with exr₁ supplying the processing power and other resources to run the B application ("B→exr₁" in the figure), and exr₂ supplying the processing power and other resources to run the C application ("C→exr₂" in the figure). An execution resource could be, e.g., a physical computer, a virtual machine (discussed in Section 7), or a running server program that is capable of serving client requests, start a virtual machine, or even rent computing cycles and storage from another organization. Similarly, client C₂, is using one application, C, which is supported by execution resource exr₃. It should be noticed that the same application (C in this case) can be rented out to multiple clients at the same time, as long as the cloud provider can provide the execution resources to support the application. As shown in Figure 2.9 B, when an additional client requests applications from the cloud, the cloud provider allocates extra execution resources for supporting the requested applications [31].

In order to facilitate the understanding of scope and division of roles between cloud consumer and cloud provider, the following figure is placed as a reference.

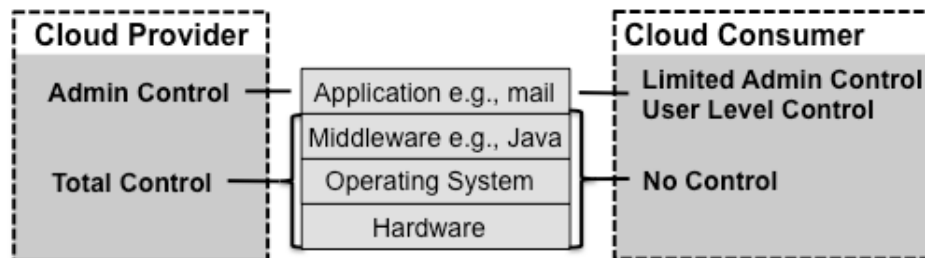


Figure 2.10 SaaS Provider/Consumer Scope of Control [31]

The figure above depicts a “user level control”, which represents that a consumer has control over the application-specific resources that SaaS application makes available. In some cases, a consumer also has some limited administrative control over an application.

A provider normally has significantly more administrative control at the application level. The responsibilities of a provider are to deploy, configure, update, and manage the operation of the application in order to provide expected service levels to consumers [31].

The middleware layer provides software blocks that are the base of an application. It can take various forms, ranging from: traditional software libraries, to software interpreters, to invocations of remote network services. Moreover, middleware components can provide database services, user authentication services, identity managements, etc. Basically consumers cannot have an access to this layer; neither should they have access to the operating system nor hardware layers [31].

2.2.1.1 Benefits

Nowadays, more and more companies decide to take advantage of SaaS solutions, as they provide scalability and also shift significant burdens from consumers to providers. Thus it gives better efficiency and sometimes even better performance. The main benefits of SaaS cloud can be pointed as follows [31].

- **Browser based.** SaaS application deployment is very convenient and efficient with typically almost no software required.
- **Licence management.** Consumers can employ a single license on multiple computers at different times instead of purchasing extra licenses for separate computers that may not be used and thus over-provisioning the license. Moreover, traditional license management protocols and license servers are not necessary to protect the intellectual property of application developers because the software runs in the provider's infrastructure and can be directly metered and billed.

- **Centralized data administration.** From the consumer's point of view in SaaS model, management and data are centralized. As such the SaaS provider can supply professional management of the data, including compliance checking, security scanning, backup, and disaster recovery. When these services are provided off-premises SaaS management of data gives protection against the possibility of a single catastrophe destroying both the consumer's facility and data. For on-site private and community SaaS clouds, the benefits of centralized management are similar however there is less resilience against catastrophic losses unless consumers explicitly plan for those contingencies.
- **No infrastructure involvement.** In case of outsourced or public SaaS clouds, consumers need not become involved with the management of a provider's infrastructure.
- **Pay for what you use model.** Public SaaS clouds allow a consumer to begin using an application without the up-front costs of equipment acquisition, but potentially with a recurring usage fee.

2.2.1.2 Drawbacks

For all scenarios, SaaS clouds place significant reliance on consumer browsers as most of computation is done on provider side. This brings up number of issues and concerns [31].

- **Lack of 100% Security.** Although browsers encrypt their communications with cloud providers, subtle disclosures of information are still possible. For example, the very presence or absence of message traffic, or the sizes of messages sent, or the originating locations may leak information that is indirect but still of importance to some consumers. Moreover man-in-the-middle attacks on the cryptographic protocols used by browsers can allow an attacker to hijack a consumer's cloud resources.
- **Browser Dependence.** If a consumer visits a malicious Web site and the browser becomes contaminated, subsequent access to a SaaS application might compromise the consumer's data. Data from different SaaS applications might be inadvertently mixed on consumer systems within consumer Web browsers.
- **Network Dependence** - In the public SaaS cloud scenario, the network's reliability cannot be guaranteed either by the cloud consumer or by the cloud provider as the Internet is not controlled by either one.
- **No Portability.** Formats for exporting and importing data may not be entirely compatible between SaaS clouds. Customized workflow and business rules, user

interface and application settings, support scripts, data extensions, and add-ons developed over time can also be vendor specific and not easily transferable.

2.2.1.3 Main providers

Theoretically any email client or online software provider could be called a SaaS provider. Thus two leading cloud providers that identify their services as SaaS will be described.

2.2.1.3.1 Google

On February 2006, Google created a beta version of Gmail For Your Domain for an invitation only that allowed Gmail to be used with a custom domain name. Then on August of 2006 Google expanded on this service and developed Google Apps For Your Domain [35]. Those two steps were founding stone of Google's SaaS platform – Google Apps.



Currently Google Apps offers three options:

- Google Apps – Individuals, groups and entrepreneurs can get up to 10 custom accounts, all for free.
- Google Apps for Business - It is free for 30 first days, later on the cost of each month is of 5\$ per month and user account or 50\$ per year.
- Google Apps for Education – Entirely free solution for schools and educational institutions.


	Google Apps	Google Apps for Business	Google Apps for education
Messaging apps	x	x	x
Collaboration apps	x	x	x
Business security	x	x	x
Additional business apps		x	x
Business features		x	x
Business support and reliability		x	x

Table 2.4 Services offered by Google Apps

2.2.1.3.2 Salesforce.com

Salesforce.com is the leading CRM SaaS vendor and had 10.6% of the overall CRM market in 2008. The CRM of Salesforce.com is broken down into various categories [36]:

- **The Sales Cloud** - it includes a real-time sales collaborative tool that is Chatter, it provides sales representatives with a customer profile and account history. It also

allows the user to manage marketing campaign spending and performance across a variety of channels from a single application, tracks all opportunity-related data including milestones, decision makers, customer communications, and any other information unique to the company's sales process.

- **The Service Cloud** - The Service Cloud provides companies with a call center-like view that enables companies to create and track cases coming in from every channel, and automatically route and escalate what is important.
- **Chatter** - It is a real-time collaboration platform for users. The service sends information via a real-time news stream. Users can follow coworkers and receive broadcast updates about project and customer status. Users can also form groups and post messages on each other's profiles to collaborate on projects.
- **AppExchange** - It is a marketplace for cloud computing applications built for the Salesforce.com community and delivered by partners or by third-party developers, which users can purchase and add to their Salesforce.com.
- **Configuration** - Salesforce users can configure their CRM application. In the system, there are tabs such as "Contacts," "Reports," and "Accounts." Each tab contains associated information.
- **Web services** - In addition to the web interface, salesforce.com provides a SOAP/REST Web service API that allows for integration with other systems.

2.2.1.4 Case Studies

2.2.1.4.1 BBVA

BBVA is a customer-centric global financial services group founded in Spain in 1857. They now have over 110,000 employees in more than 30 countries across the world.

On January 11th 2012 it was made public that BBVA decided for migrating their business to Google Apps to increase efficiency and to help their teams to collaborate more easily, regardless of location.

According to José Olalla CIO at BBVA, the corporation will migrate their old email systems to the cloud with Gmail. Google Talk, Google Sites and Google Docs will allow their teams to communicate and share ideas more easily - working in a way that they have never experienced before [33].

Currently more than 35,000 BBVA workers in Spain are using the Google Apps productivity tools. By the end of 2012 it is expected that they will have migrated 110,000 employees across BBVA global network.

Although the financial details of this cooperation were never made public, many speculate that as Customers at average often save about 50 to 70 percent compared to their previous software solutions, the same savings will be brought to BBVA.

2.2.1.4.2 IRB Barcelona

The institute of Biomedical Investigation in Barcelona (IRB Barcelona) is an independent non-profit research institution dedicated to applied and basic biomedical science, its objective is to contribute to improving life through biomedical advancements. The center was founded by the end of 2005 by Generalitat de Catalunya, the University of Barcelona (UB) and the Parque Científico de Barcelona (PCB). Currently around 400 people work for IRB Barcelona.

Because of the way IRB Barcelona works, email service is a crucial tool all the time.

Because of big quantity of emails, spam had converted in a big concern of many employees. At the time the center did not have sufficient human resources to implement and maintain the email service by them. That is why it was decided to look for outsourced services that would satisfy their needs [34].

Before contracting Google Apps, IRB Barcelona was familiar with services offered by Google, although they were not certain whether Google could offer hosting of the email service.

The first tryouts were conducted on July of 2006. Later on, the Institute created email accounts with Google Apps for Business for all their employees, achieving what they were looking for: a trustworthy email service that offers 99,9% uptime, under their own domain, free of spam and viruses, multiplatform, dedicated for hundreds of users and that could be rapidly deployed with little resources and all of it with no loss of control [34].

Moreover, Google Apps brought to the center not only the email service, but also gave opportunity to elaborate calendars such as personal agendas, shared agendas or resource reservation [34].

Since the implementation of Google Apps in IRB Barcelona, the savings for the organization in maintenance and IT support have ranged between 15.000 and 20.000 yearly. Moreover, Google allowed IRB Barcelona to achieve other advantages applicable to their job. i.e.:

- Spam reduced to zero
- Functional, Easy and convenient interface
- Bigger email storage
- Constantly evolving platform
- Additional applications

2.2.2 PaaS (Platform as a Service)

PaaS providers offer a platform for others to use. What is being provided is partially operating system and partially middleware. A proper PaaS provider has to take care of everything that is necessary in order to run a specific language or technology stack.

Platform as a Service	
The consumers	<ul style="list-style-type: none"> • Application developers, that design and implement an application's software • Application testers, who run applications in various testing environments • Application deployers, who publish completed applications into the cloud, and manage possible conflicts arising from multiple versions of an application • Application administrators, who configure, tune, and monitor application performance on a platform • Application end users, who subscribe to the applications deployed on a PaaS cloud: to end users, access to applications is the same as using a SaaS cloud
Consumer Acquires	The right to use the PaaS cloud provider's tools and execution resources to develop, test, deploy and administer applications.
Fees Calculation	Normally the fees are based on the number of consumers, storage, processing, or network resources consumed by the platform, requests serviced, and the time the platform is in use

Table 2.5 PaaS general features

PaaS provides a carefree environment for developers to work i.e. it lets them focus on code without having to worry about configuration and maintenance of the underlying infrastructure. System engineers and administrators on the other hand can go back to doing systems work instead of server work. Architects though, might take advantage of the flexibility provided by PaaS. Individual computing needs, like a database, can be used without requiring internal expertise for running it. Such flexibility is very useful as prototypes can be put together in days, not weeks or months [40].

Below is shown a simplified view of Provider-Consumer interaction flow in PaaS [31].

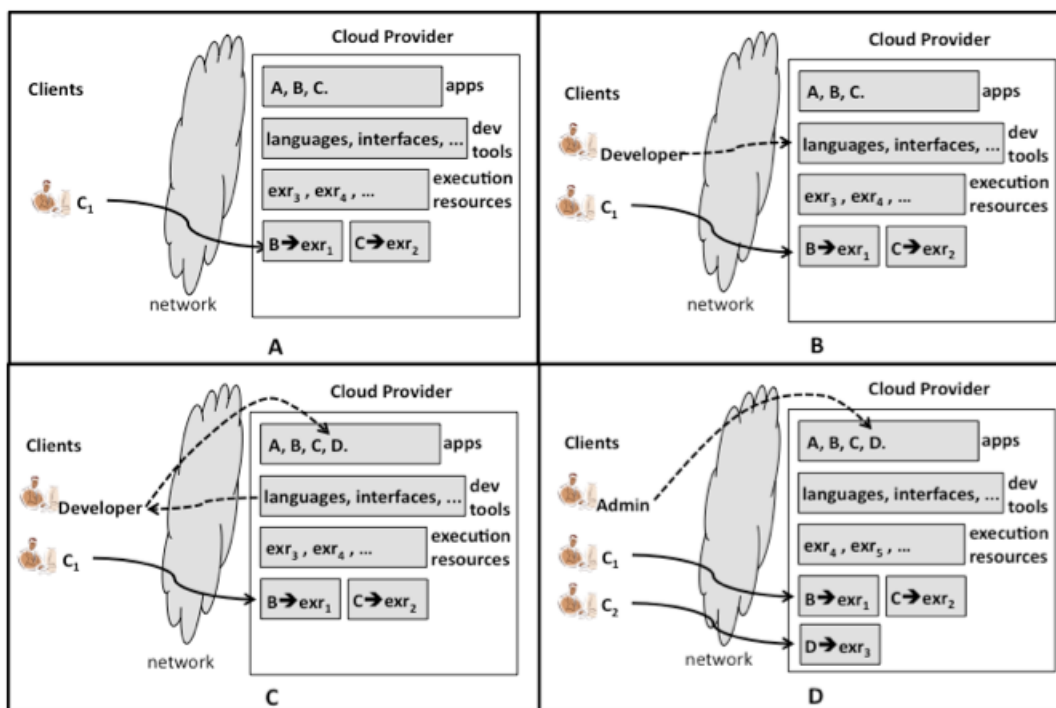


Figure 2.11 Provider/Consumer Interaction Dynamics [31]

Figure A shows the PaaS provider that has a current inventory of three applications deployed (apps), set of development tools (dev tools), and a set of execution environments (extri). There are also depicted two active applications, $B \rightarrow \text{exr}_1$ and $C \rightarrow \text{exr}_2$ indicating that applications B and C are using separate execution resources. The figure B shows the developer client accessing the development tools of the provider. The example of such tools could be programming languages, compilers, interfaces, testing tools, and tools for deployment of an application. In figure C it can be seen how the developer uses the tools. Thus one may download tools and use them locally in the developer's infrastructure or can access those tools in the provider's infrastructure as well. In each case the result is a new application D, as depicted in the figure, which is deployed onto the provider's infrastructure. In figure D can be observed an administrator configuring the new application that has been made available, as well as a new client, C2 using that new application [31]. Summarizing the figure 2.11 it depicts various ways of usage of PaaS cloud identified by NIST.

As it is illustrated on the figure below, the cloud provider has control over the more privileged, lower layers of the software stack (also has control over networking infrastructure such as LANs and routers between data centers). Thus it also shows how control and management responsibilities are shared [31].

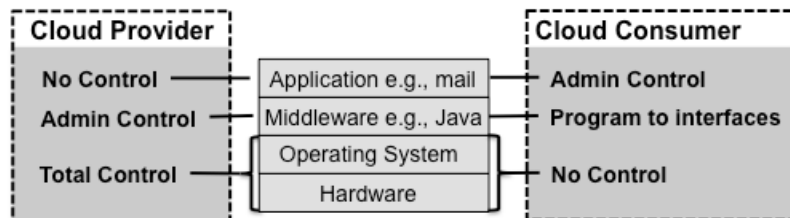


Figure 2.12 PaaS Component Stack and Scope of Control [31]

The provider makes programming and utility interfaces available to the consumer at the middleware layer. Thus those interfaces provide the execution environment within which consumer applications run and provide access to needed resources such as CPU cycles, memory, persistent storage, data stores, data bases, network connections, etc. The provider determines the programming model, i.e., the circumstances under which consumer application code gets activated, and monitors the activities of consumer programs for billing and other management purposes. Once a consumer has used the facilities of the PaaS cloud to implement and deploy an application, the application essentially is a SaaS deployment and the consumer has administrative control over the application subject only to the provider supporting the consumer according to the terms of use [31].

Some of the benefits and issues identified are described in continuation based on the source [31].

2.2.2.1 Benefits

- **No additional software.** Working on the provider's environment is very convenient as very often it is just the browser that is needed for most of the operations.
- **Centralization of data.** From the consumer's point of view in PaaS model, management and data are centralized. As such the PaaS provider can supply professional management of the data, including compliance checking, security scanning, backup, and disaster recovery. When these services are provided off-premises PaaS management of data gives protection against the possibility of a single catastrophe destroying both the consumer's facility and data. For on-site private and community PaaS clouds, the benefits of centralized management are similar, however there is less resilience against catastrophic losses unless consumers explicitly plan for those contingencies.

- **Ready to use development environment.** In case of outsourced and public PaaS clouds, consumers need not become involved with the management of a provider's infrastructure.
- **Pay for what you use model.** Outsourced and public PaaS clouds allow a consumer to begin using an application without the up-front costs of equipment acquisition, but potentially with a recurring usage fee.
- **Alleviated Scalable Application Development and Deployment.** Organizations can develop and deploy enterprise applications and maintain centralized control over their operation and the data that is processed with them. Application development frameworks in PaaS normally provide design patterns that support a high level of scalability, which enables well-written applications to operate smoothly through large fluctuations in demand. In on-premises scenarios, scalability will be limited to the resources provided by consumer data centers. Nevertheless in outsourced scenarios more resources may be available at the providers' facilities and, particularly in the case of public cloud, well-written PaaS applications can be rapidly deployed to large amounts of consumers and provide very large quantities of data and processing services.

2.2.2.2 Drawbacks

Similarly to SaaS, PaaS clouds perform a more application-level logic at provider facilities than traditional computing solutions. Thus PaaS shares some of SaaS' concerns.

- **Possibility of information disclosures.** For example, the very presence or absence of message traffic, or the sizes of messages sent, or the originating locations may leak information that is indirect but still of importance to some consumers
- **Network Dependency.** In case of network failure outsourced PaaS platforms become non-operational as there is no connection with them in such case.
- **PaaS clouds are not portable.** This is a concern particularly when platforms require proprietary languages and run-time environments.
- **Vendor lock-in.** In many cases of PaaS happens (e.g. Google App Engine) that uploaded application to PaaS cloud is not retrievable from the providers' servers. It is also called a vendor lock-in, i.e. once a company deploys its software onto the cloud it becomes dependent on that cloud provider.
- **Event-based Scheduling.** PaaS applications can be event driven with the events composed of HTTP messages. This kind of design is cost effective (absent an

outstanding request, few resources are consumed), however it poses resource constraints on applications, such as they must answer a request within a time interval or they must continue a long-running request by queuing synthetic messages that then can be serviced. Moreover, tasks that execute rapidly in a local application not necessarily offer equivalent performance in a PaaS application.

- **Security Engineering of PaaS Applications.** Unlike the case of an application that can potentially run in an isolated environment using only local resources, PaaS applications access networks intrinsically. Moreover, PaaS applications must use cryptography in an explicit way, and must interact with the presentation features of common Web browsers that provide output to consumers.

2.2.2.3 Main Providers

Since the next chapter is fully dedicated to PaaS, where more detailed description of main PaaS cloud providers will be given, below is a short list of the few leading players on the market.

- Google App Engine
- Windows Azure
- Appistry
- Force.com
- Heroku

2.2.2.4 Case Study

The following case study proves that migration to PaaS can be quite beneficial to the company.

2.2.2.4.1 Menumate

Menumate is a provider of point of sale hardware and software for the hospitality industry across Australasia. Menumate has decided to take advantage of the Force.com PaaS to migrate over time a series of legacy applications used in the business.

Some of these applications are [41]:

- **License Key Generation** - The Menumate software uses license keys to activate the features that the customer has paid for.

- **Enhanced Case Management** - A lot of the support cases Menumate were dealing with were orders for consumables. To handle this they had a separate DOS based application that would allow the user to build up an order and create an invoice.
- **Label Printing** - Another legacy application was for creating freight labels for sending consumables and hardware to customers.

Daniel Fowlie and Abhinav Keswani are Directors of development house Trineo, the company responsible for boutique development for Menumate. Fowlie stated that the use of the Force.com platform has allowed Menumate to centralize, modernize and integrate an otherwise disparate in-house software toolkit.

Keswani said that he thinks that a more conventional development approach would require significant infrastructure, connectivity, security and would introduce uptime considerations - whereas the Force.com platform inherently provides these non-functional requirements - allowing Menumate and Trineo to focus purely on developing the needed functionality. Additionally, utilizing a PaaS approach has meant Trineo could take advantage of both existing integrations and automated deployment tools - another example of PaaS easing the development process [41].

Using PaaS, Trineo have been able to migrate over time previously mentioned legacy applications used in the business. PaaS cloud brought following benefits to those applications.

- **Case of Key License generation** - PaaS allowed Menumate to quickly port this code to Force.com where the license keys are linked to the customer record in the Salesforce.com CRM. This allows Sales and Support staff to quickly see the status of licenses.
- **Case of Enhanced Case Management** - because of PaaS Menumate now can add products to a support case and automatically send an invoice to their accounting software using an existing integration product.
- **Case of Label Printing** - Utilizing the PaaS technology enabled for freight labels to be printed directly from the customer record.

Utilizing a PaaS development environment has resulted in the creation of these applications being significantly faster than would otherwise be the case. In some examples, in the absence of PaaS, the cost of developing the application would have been prohibitive [41].

2.2.3 IaaS (Infrastructure as a Service)

The basic idea of IaaS is the outsourcing of servers for disk space, database and/or computation time instead of having a separate datacenter within the company. With an infrastructure like IaaS what is given is a solution based on virtualization where it is paid for resource consumption.

Infrastructure as a Service	
The consumers	<ul style="list-style-type: none"> System administrators
Consumer Acquires	<ul style="list-style-type: none"> Access to virtual computers Network-accessible storage Network infrastructure components such as firewalls, and configuration services
Fees Calculation	Normally the fees are calculated basing on cpu hour, data GB stored per hour, network bandwidth consumed, network infrastructure used per hour, or value-added services used

Tabla 2.6 IaaS General Features

Below can be seen a simplified view of the interactions within an IaaS cloud.

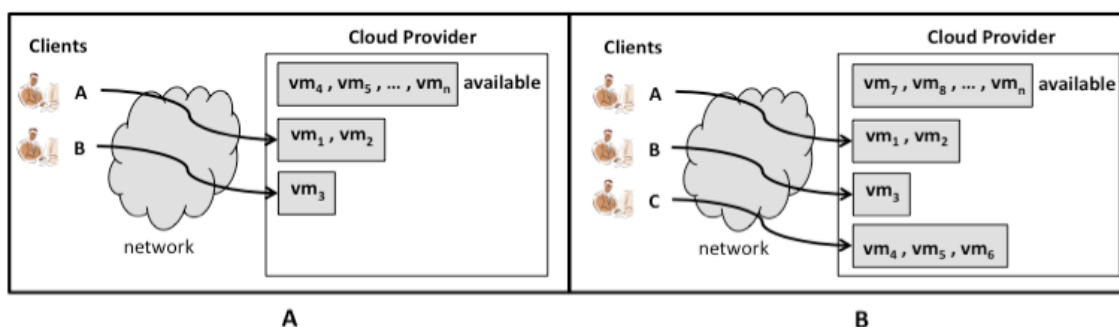


Figure 2.13 Provider/Consumer Interaction Dynamics [31]

The figure 2.13 A Shows clients that are interacting with an IaaS cloud over a network. The provider has several available virtual machines that he can allocate to clients. In the figure, client A has access to vm_1 and vm_2 , and client B has access to vm_3 . The provider retains vm_4 through vm_n , where it is presumed that n is larger than the number of vms any client is expected to request [31]. Another situation is shown on the figure 2.13 B, where just after a new client C has requested and acquired access to three more vms. At this stage, client C has access to vm_4 , vm_5 , and vm_6 , and the provider now retains only vm_7 through vm_n [31].

The above is extremely simplified schema of how IaaS cloud really works. Thus figure 2.13 only depicts virtual machine allocation (by a provider) and interaction (by a consumer). Practical

IaaS cloud systems also provide persistent data storage and stable network connectivity, as well as they must track resources with economic costs, and bill those costs to consumers [31]. In order to represent how control and management responsibilities are shared, the IaaS cloud component Stack with scope of control is shown below.

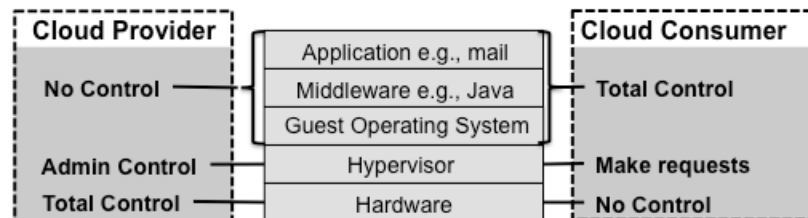


Figure 2.14 IaaS Component Stack and Scope of Control [31]

The cloud provider controls the most privileged, lower layers of the software stack. As depicted in the figure above the provider maintains total control over the physical hardware and administrative control over the hypervisor layer e.g. Xen⁵. Thus the consumer can make requests to the cloud to create and manage VMs but these requests are honored only in case they conform to the provider's policies over resource assignment. Via hypervisor, the provider will normally supply interfaces for the networking functions that the consumers can use in order to configure the virtual network within the provider's infrastructure. The consumer maintains the complete control over the guest operating system functionality in each of virtual machines, and all the software layers above. This structure gives very significant control over the software stack to consumers that have to take responsibility to operate, update and configure these computing resources for security and reliability. As such in this sense the approach of IaaS is very different from SaaS and PaaS clouds where most of those issues are handled transparently for consumers [31].

2.2.3.1 Benefits

Although proprietary cloud providers do not release technical information about their system architectures, three Open Source systems i.e. NASA Nebula, Eucalyptus, Ubuntu Enterprise Cloud (all based on the Eucalyptus source code) provide detailed technical information about specific system architectures. Furthermore are described main benefits of IaaS pointed out by the source [31].

- Like in other service models savings in Up-front costs, and access to cloud services over the open Internet.
- **Freedom of choice.** A very important aspect of administrative access to a VM is that consumer can run almost any software that he wants, including a custom operating system.

⁵ Xen is a Hypervisor providing services that allow multiple computer operating systems to execute on the same computer hardware concurrently.

- **Rapid and effortless hardware employment.** In public and outsourced IaaS clouds the ability to quickly rent and then release large numbers of VMs or other cloud resources is provided. This gives a consumer the possibility of quickly setting up large networks of VMs running software chosen by consumer to solve large problems without incurring the expense of purchasing and maintaining the necessary hardware.
- **Compatibility with local environment.** Due to the fact that IaaS clouds allow consumers to install and run operating systems of their choice, a high level of compatibility can be maintained between legacy applications and workloads in an IaaS cloud. Also, many user-facing applications can be run in an IaaS cloud by virtual desktop technology.

2.2.3.2 Drawbacks

As in other service models IaaS cloud shares similar concerns in regards to network dependence, and browser dependency. The following are the issues related exclusively with IaaS cloud pointed out by the source [31].

- **Legacy Security Vulnerabilities impact.** Most of IaaS systems give its users a possibility to create and retain virtual machines in various states e.g., running, suspended and off. An inactive VM can become out of date with important security updates; whenever such out-of-date VM is activated it may become compromised.
- **Virtual Machine Sprawl.** IaaS clouds expose consumers to all of the security vulnerabilities of the legacy software systems allowed by consumers to run in the provider's infrastructure.
- **IaaS provider authenticity verification.** The user's browser will most likely use public key cryptography to establish a private link to the cloud provider. Nevertheless, it is consumer that is in charge of checking the identity of the cloud Website in order to check if the private link is not with an imposter.
- **Robustness of VM-level Isolation.** Cloud consumers must be isolated from each other except when they choose to interact. Normally an IaaS cloud uses a hypervisor (which is a software layer), in combination with hardware support for virtualization (e.g., AMD-V and Intel VT-x), to split each physical computer into multiple virtual machines. Isolation of the virtual machines depends on the correct implementation and configuration of the hypervisor. Hardware virtualization provided by hypervisors has become a widely used technique for providing isolated, computing environments, but the strength of the isolation in the presence of sophisticated attackers is an open research question.

- **Features for Dynamic Network Configuration for Providing Isolation.** In order to prevent unwanted interactions among consumers, the cloud network must prevent a consumer from observing other consumer's packets. Furthermore it has to reserve enough bandwidth to ensure that each consumer has the expected level of service. The allocation a Virtual Machines typically is a matter of a few minutes, and the corresponding network configuration must be performed just as quickly. Various techniques for logical view of network's topology, such as Virtual Local Area Networks (VLANs) and overlay networks, can be quickly reconfigured. Thus they (and perhaps support in hypervisors as well) have to be configured carefully in order to prevent interference between networks belonging to different consumers.
- **Data Erase Practices.** Virtual machines access disk resources maintained by the provider. When a consumer releases such a resource, the provider must ensure that the next consumer that rents the resource does not observe data residue from previous tenants. Strong data erase policies (e.g., multiple overwriting of disk blocks) are time consuming and may not be compatible with high performance when tenants are changing. Data replication and backup practices also complicate data erase practices.

2.2.3.3 Main Providers

In this part are described major IaaS cloud providers and their solutions.

2.2.3.3.1 Amazon

Established in July 2002, Amazon Web Services provide online services for other web sites or client-side applications. Most of these services are not exposed directly to end users, but instead offer functionality that other developers can use in their applications. Amazon Web Services' offerings are accessed over HTTP, using REST and SOAP protocols. All services are billed based on usage, but how usage is measured for billing varies from service to service [38]

In 2004 an engineer at Amazon presented a paper proposing how the company could make a profit on the infrastructure required to run the Amazon.com store. As a result, Amazon EC2 was built by a team in Cape Town, South Africa [38].

Amazon, which is the leader and one of the pioneers of cloud computing achieved that position because of the EC2 (Amazon Elastic Compute Cloud), which allows customers to set up and access virtual servers through a simple Web interface. Amazon EC2 allows customers to quickly scale capacity as their computing requirements change.

The services offered by AWS in regards to IaaS are depicted in the following table.

AWS Types	Offered Products
Computation Capacity	Amazon Elastic Compute Cloud (EC2) Amazon Elastic MapReduce
Database	Amazon SimpleDB Amazon Relational Database Service (RDS)
Computation networks	Amazon Virtual Private Cloud (VPC)
Storage	Amazon Simple Storage Service (S3) Amazon Elastic Block (EBS) AWS Import/Export
Messaging	Amazon Simple Queue Service (SQS) Amazon Simple Notification Service (SNS)
Monitoring	Amazon Cloud Watch

Table 2.7 Services/tools provided by Amazon Web Services

It should also be noted that EC2 provides scalable virtual private servers using Xen.

2.2.3.3.2 Joyent

The Joyent SmartMachine is different from traditional and virtual machine architectures mainly because it goes further in abstraction of the hardware. Thus it presents the hosted application with access to a pool of resources through an already existing operating system rather than requiring installation of an operating system onto virtualized hardware that divides a piece of real hardware. Hence, the pool of resources can be the size of the physical capabilities of a single piece of hardware.

Joyent SmartMachines employ a very lightweight container-based virtualization rather than Xen/ESX hardware virtualization. This provides a wide range of performance enhancements [39]:

- it frees up a large pool of memory that would otherwise be used to run individual operating systems
- it eliminates the need to create emulated hardware
- it eliminates a virtualization layer, which introduces latency

Joyent provides preconfigured SmartMachines optimized for the most common elements in the application architecture. Application servers, a flexible Unix development environment that comes preconfigured with Apache, Nginx, MySQL, PHP, Ruby on Rails, and JAVA pre-installed [39].

2.2.3.3.3 Layered Technologies (Layered Tech)

Layered Tech is hosting and cloud computing service provider offering Virtual Machines that provide high-performance, high-availability computing resources that are not confined to a single, vulnerable server. Leveraging Layered Tech's Virtual Machine, customer's site is no longer tied to one server but is implemented across many nodes, allowing them to easily scale in order to meet their changing business requirements [39].

2.2.3.3.4 Terremark

Terremark is a traditional hosting company, which entered the cloud computing space with its Enterprise Cloud offering. The Enterprise Cloud is a web-based managed platform that gives customers the power to dynamically allocate computing resources for mission-critical apps in minutes. An Enterprise Cloud service gives customers control over the pool of compute resources, allowing them to deploy server capacity on demand. Pre-configured server templates are available for Microsoft Windows, Linux, and Sun Solaris operating systems [39].

2.2.3.3.5 GoGrid

GoGrid is an established IaaS cloud provider. The GoGrid platform offers Web-based storage and the ability to deploy Windows- and Linux cloud servers on demand through an intuitive web interface. With GoGrid Cloud Hosting customers only pay for the deployed RAM and outbound data transfer that they actually use [39].

Key features

- Standard Windows and Linux Cloud Servers
- Add Dedicated Servers to your GoGrid VLAN
- Create, Save, and Deploy Custom Server Images
- Scale Up or Down, in Real-Time

2.2.3.3.6 VMware

VMware offers private and public cloud computing. The Private cloud computing has been designed to ensure security and compliance that the business demands by deploying a private cloud infrastructure inside your firewall. It is developed on the industry's most trusted virtualization platform – VMware vSphere. The infrastructure allows customers to virtualize all their assets in IT infrastructure and offer IT as a Service through policy-driven management

that automates routine operational tasks and leads to efficiency and savings across the business.

The public cloud offers customers the freedom of open standards and interoperability of applications. VMware is a great choice for organization looking to expand IT capacity or migrate applications to the public cloud. A common management and infrastructure platform ensures visibility of pooled resources along with the elasticity to provide the highest service-levels for applications [39].

2.2.3.3.7 AT&T

AT&T Synaptic Hosting is designed to give companies on-demand access to scalable network, storage and computing capabilities. The solution provides companies with a self-service approach for using IT solutions delivered by AT&T, and allows them to quickly address demands for scaling the computing capacity according to their business requirements.

AT&T Synaptic Hosting allows users to [39]:

- Launch and scale their application quickly.
- Burst IT capacity in real time to handle traffic spikes.
- Pay only for the IT capacity they need.
- Deliver high uptime via built-in security and redundancy.

2.2.3.3.8 Rackspace

The Rackspace Cloud is composed of three main services: Cloud sites, a platform for building Web sites; Cloud Files, a storage service; and Cloud Servers, service that provides access to virtualized server instances. Rackspace Cloud Servers are ideal for companies that have sites with big fluctuations in traffic, as they can easily add or remove Cloud Servers on demand. Customers can choose from a variety of popular Linux Distributions: Ubuntu, Debian, Gentoo, Centos, Fedora, Arch and Red Hat Enterprise Linux [39].

2.2.3.4 Case study

The following is a case study that could serve as a reference when considering migrating to an IaaS cloud.

2.2.3.4.1 Oil & Gas SME [37]

This case study points out the potential benefits and risks associated with the migration of an IT system in the oil & gas industry from an in-house data center to Amazon EC2 from a broad variety of stakeholder perspectives across the enterprise.

The case study organization is a UK based SME that provides IT solutions for the Oil & Gas industry. The company employs around 30 employees with offices in the UK and the Middle

East. The structure of the company is based on functional divisions such as Administration, Engineering and Support, where Engineering department is the largest.

The company (company C) also owns some offshore assets in the North Sea oilfields. In order to manage their offshore operations by monitoring data from their assets on a minute-by-minute basis the company needed a data acquisition system. Company's assets rely on the production facilities of other major oil company (company A). Thus the data comes onshore via that major oil company communication links. Since the analyzed company (Company C) does not have capabilities to develop their own IT systems, they outsourced the development and management of the system to other IT solutions company (company B) with a small data center. The overview of the system consisting of two servers is shown on the figure below.

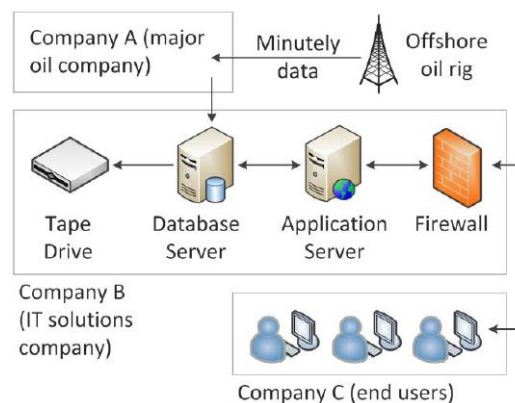


Figure 2.15 System Overview [37]

This case study investigates how the same system could be deployed using the cloud solution offered by Amazon Web Services.

Below is an overview of this scenario, where Company B deploys and maintains the same system in the cloud.

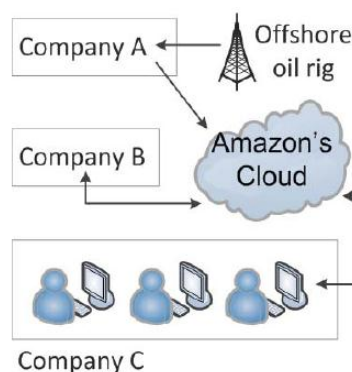


Figure 2.16 System deployed in the cloud [37]

This case study involved fieldwork at Company B's offices between May to July 2009.

Company C paid £104,000 to Company B for the system in 2005, £19,400 of which was for the system's infrastructure; the rest of the costs were for system development and deployment. In addition, Company C pays £43,000 per year to Company B for system support and maintenance, £3,600 of which is for the running costs of the system infrastructure. Amazon EC2 provides an option of using either small or large server instances depending on the amount of CPU power and RAM required. The system could initially run on two small instances, as the application and database server do not seem to be under a heavy load. However, this could be changed for large instances if the performance is found to be unacceptable. This would not have been possible using the existing approach since all hardware must be purchased before the system is deployed, and cannot easily be changed afterwards.

Table 1 shows a comparison of the costs of the system infrastructure, the amounts have been rounded to the nearest £10. The following specifications were used to calculate the costs of running the system on AWS: two Microsoft Windows On-Demand instance (AWS do not offer reserved instances for Windows) in Europe running 730 hours per month (i.e. 24x7); 20GB data transfer in; 20GB data transfer out; 200GB EBS storage (i.e. amount of effective storage on existing servers), 100 million EBS I/O request; 30GB EBS snapshot storage (for daily backups); 10 snapshot GET requests (in case backups need to be retrieved); 30 snapshot PUT requests (for daily backups).

Period	Amazon Server Instances			Company B
	2 small	1 small + 1 large	2 large	
1 Month	£200	£390	£590	£620
1 Year	£2,400	£4,680	£7,080	£7,440
5 Years	£12,000	£23,400	£35,400	£37,200

Table 2.8 Infrastructure cost comparison

Company B maintains a database of all the support calls they receive by telephone or email either externally from end users or internally from support engineers doing health checks. Since the system went live in 2005, 218 support calls have been made regarding the operation of the system. The majority of these calls were about software problems, however, it was found that many of the calls were related to the infrastructure.

Those calls could potentially have been eliminated if the system was deployed in the cloud since Amazon would be responsible for hardware related issues. This accounts for around 21% of the support calls but it should be noted that some additional calls might be introduced if the system was migrated to the cloud. These clouds related issues could include power outages at Amazon's data centers or network latency issues; however, the important point is that Amazon would deal with these issues. This could be seen as a big advantage for Company B's support department as it allows them to focus on software related issues, which are more important to the end users.

The analysis of the interview data suggests that the proposed cloud migration would have a positive net benefit from the perspective of the business development functions of the enterprise and the more junior levels of the IT support functions. A perceived zero net benefit was perceived by the project management and support management functions of the enterprise. A negative net benefit was perceived by the technical manager and the support engineer functions of the enterprise. Stakeholder impact analysis data suggests that there are numerous potential benefits but also risks associated with the migration of the system to the cloud. The benefits are the following:

- Opportunity to manage income & outgoings
- Opportunity to offer new products/services
- Improved status
- Removal of tedious work
- Improve satisfaction of work
- Opportunity to develop new skills
- Opportunity for organizational growth

And the risks:

- Deterioration of customer care & service quality
- Increased dependence on external 3rd party
- Decrease of satisfying work
- Departmental downsizing
- Uncertainty with new technology
- Lack of supporting resources
- Lack of understanding of the cloud

The system infrastructure in the case study would cost 37% less over 5 years on Amazon EC2, and using cloud computing could have potentially eliminated 21% of the support calls for this system. These findings seem significant enough to call for a migration of the system to the cloud but our stakeholder impact analysis revealed that there are significant disadvantages tied to the promised benefits. The disadvantages include risks to customer satisfaction and overall service quality due to diffusion of control to third parties; decreased job satisfaction due to changes in nature of work; and opening the organization to long term cost volatility in terms of cloud-usage and data transfer costs.

2.3 Summary

Although cloud computing seems to be a recent development, many argue that it already existed ever since dawn of large companies such as Google or Amazon. Nevertheless something has changed, and with no doubt it was a way that already existing infrastructure has started to be offered.

Summarizing the chapter, four deployment models and three service models were given a brief description. In the end the following was concluded.

- **Public Clouds** are currently the most popular deployment model of a cloud. Thus they seem to provide IT resources (applications, infrastructure, development environment) at the best price on the market, with all those resources residing on the providers' infrastructure which in fact are more secure and reliable than very often ones established on-premises (private).
- **Private cloud** computing is one of alternatives for deployment option and in some cases it can be a good practice to take advantage of them. Especially for organizations with enough scale, buying power, and expertise, private clouds offer the advantages of increased control, predictability, and security [24].
- **Community Cloud** can be a good alternative for the companies within the same capital group, although adoption of this deployment model has been very slow.
- **Hybrid clouds** are moving steadily to becoming more popular for mid-sized and large organizations as they provide benefits of public and at the same time of private clouds.

Summarizing the service models:

- **IaaS** is a perfect solution for the companies that do not want to take care of setting up all the infrastructure on-premises. Although such solution requires configuring the virtual machines, it usually is much easier than on literal hardware.
- **PaaS** is an answer to the ones that do not want to be involved in neither setting up the infrastructure nor configuring the virtual machines. Thus what the consumer gets is ready-to-develop environment.
- **SaaS** provides a customer with ready-to-use applications, where customer does not have to preoccupy about the infrastructure nor how the applications function. The only thing that is required from a consumer is becoming familiar with provided application.

By combining service model with a deployment model, cloud consumer can achieve any desired outcome accompanied by e.g. lower infrastructure costs, less staff or smaller end-user learning curve.

3 Comparison of Different PaaS solutions

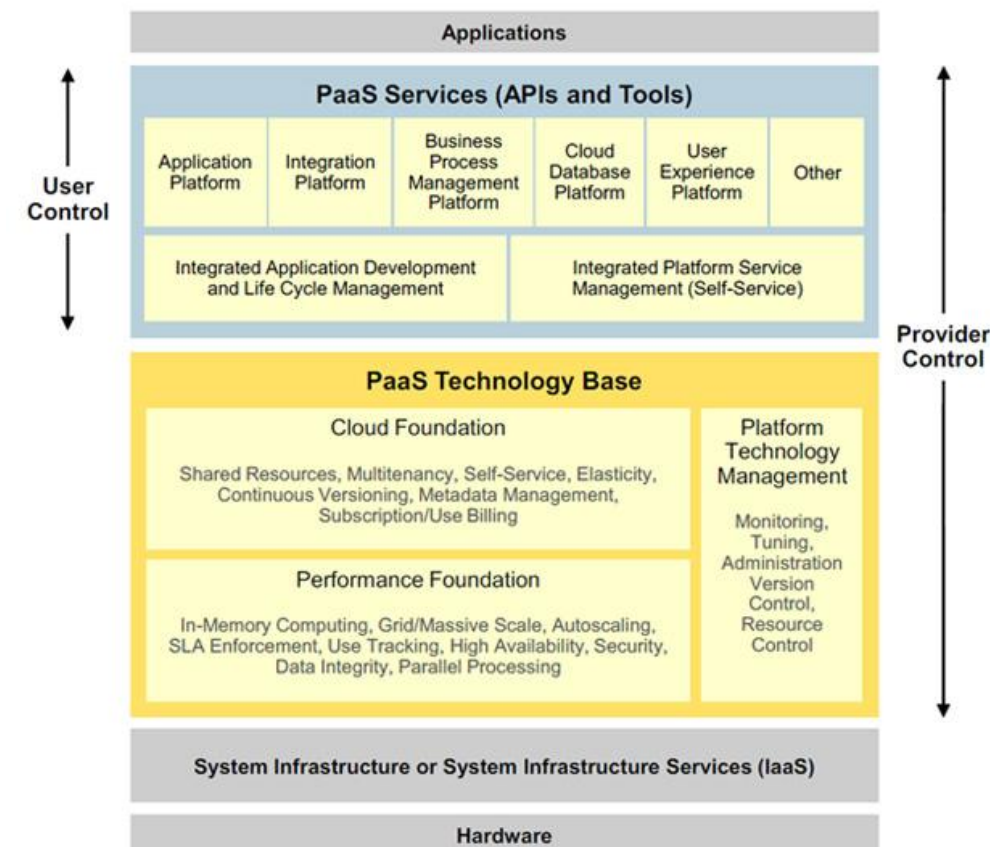
Deciding which cloud offers the best performance is difficult, mainly because no detailed, comprehensive performance comparison between cloud providers exists and the performance specifications that providers publish are usually vague.

In this chapter in order to allow for optimal comparison of different PaaS solutions, firstly a more detailed explanation of PaaS will be given, and then a brief major provider description along with a comparative table. There are many various PaaS providers, and seven of the main ones will be described in the following subchapter, following the PaaS architecture description.

3.1 PaaS Architecture

PaaS general architecture, according to Gartner, may be depicted as on the figure 3.1.

Looking at the figure 3.1 from bottom to top, beneath the PaaS layer, Gartner specifies an IaaS layer. Although there are many speculations whether a PaaS cloud has to be based on top of IaaS or not, for simplicity, it is assumed that the base for the PaaS cloud is an IaaS cloud (in this case). Thus looking at the big picture, the general flow could be explained by considering an AWS offering Elastic Beanstalk. In such scenario when a user deploys an application, AWS prepares an environment adequate for such application, i.e. for Java it runs an instance (or more) of EC2 with Apache Tomcat application server. Behind the scenes, Elastic Beanstalk handles the provisioning of a load balancer and the deployment of the WAR file to the instance (or more). When the application is deployed, all the features as the ones mentioned in the figure below in the PaaS Technology Base sector are taken care of by Amazon.



Source: Gartner (September 2011)

Figure 3.1 PaaS Architecture by Gartner [18]

As it can be observed the provider is fully in charge of the Technology base and PaaS services while the user has control over services offered by the provider. In the figure 3.1 in PaaS Services Sector - the only sector that can be controlled by the user – Gartner depicts blocks such as e.g. Application Platform, Integration Platform etc., thus these blocks represent

platforms where each of them could be provided as a service aPaaS and iPaaS respectively. Though it should be taken under advisement that what is being provided and typically referred to by PaaS is aPaaS.

The application infrastructure products that are enabled for cloud computing are referred to by Gartner as cloud-enabled application platforms (CEAPs). The same products are as well used by providers of public PaaS cloud and by independent software vendors (ISVs) providing cloud application services (SaaS). CEAPs are available from many software vendors (such as Rollbase, Cordys, LongJump and GigaSpaces), often in conjunction with a public PaaS offered by the same vendor [18].

The following are the description of the most important of the subset of PaaS services identified by Gartner.

3.1.1 iPaaS (Integration Platform as a Service)

An iPaaS is an integrated and collocated suite of cloud services providing a development and execution platform to support various types of integration projects. An iPaaS provides a range of capabilities to develop, execute, manage and govern integration flows (that is, custom-developed software and metadata implementing the integration logic needed to connect multiple applications and/or data sources). An iPaaS can be used to connect applications and deliver data in a many-to many fashion, across any combination of on-premises and off-premises applications, services, processes and data structures. It supports application and data integration within the same organization, as well as across multiple organizations. An iPaaS enables cross-organization integration in a B2B fashion that often involves e-commerce suppliers and customers, but may also include any form of multi enterprise integration and collaboration. An iPaaS provides classic application and data integration capabilities, including [19]:

- Support for multiple communication protocols and message/data formats
- Message and data transformation
- Routing
- Data delivery styles
- Data quality features
- Protocol conversions
- Adapters for multiple packaged applications, technology environments and data sources
- Service virtualization, orchestrations and community management

Additionally, what iPaaS typically offers is a self-service environment (enabling users to provision new users, deploy and monitor new integration flows, activate new integration capabilities and perform other administrative tasks). Moreover, iPaaS offerings may include a

collaborative environment (enabling different tenant organizations and service providers to share, buy and sell integration artifacts), governance capabilities (registry/repository, artifacts life cycle management, policy management and enforcement, and statistical data collection), managed file transfer features, extended B2B support and other cloud-related capabilities (security federation, usage tracking, administration, monitoring and management). All these capabilities are not necessarily provided in current iPaaS offerings, although several providers deliver significant subsets of them. As the market matures, more of these capabilities will be incorporated in more iPaaS offerings [19].

An example provider of iPaaS can be IBM WebSphere Cast Iron Live.

3.1.2 dbPaaS (Database as a Service)

Database platform as a service (dbPaaS) is engineered to run as a scalable, "elastic" service that is available on a cloud infrastructure. They are available as a one-to-many service, not necessarily relational and offer some degree of self-service. For example, Microsoft's SQL Azure, Google's Cloud SQL and database.com from salesforce.com are fully relational dbPaaS, while Amazon's SimpleDB and Google's BigTable are not relational and have different persistence models. Provided in a shared, multi-tenant environment with automatic and elastic scaling, dbPaaS is for use in simple through to complex transactions [18].

Database management systems (DBMSs) and data stores that run on cloud infrastructure, but are not purpose-built as a cloud service are excluded from dbPaaS. Most of the DBMS engines are available on a cloud infrastructure (for example, Amazon's EC2), but are not specifically engineered to take advantage of the cloud; this includes Amazon's RDS (available for MySQL or Oracle implementation), IBM's DB2, MySQL, Oracle and many others. These are offered as a hosted service, not as cloud services, since the data store software in question has no provisions for elasticity or other cloud capabilities [19].

The example providers of dbPaaS are Amazon's DynamoDB, salesforce.com database.com, Microsoft SQL Azure.

3.1.3 bpmPaaS (Business Process Management as a Service)

A BPM platform minimally includes [19]:

- A graphical business process and/or rules modeling capability
- A process registry/repository to handle the modeling metadata
- A process execution and state management engine and/or a rule engine

Other BPM-enabling technologies (BPMTs), such as those listed below, may also be part of the platform, but they are not required:

- Automated business process discovery (ABPD)
- Business activity monitoring (BAM) and business intelligence (BI)
- CEP
- Optimization/simulation

The term bpmPaaS refers to the delivery of BPM platform capabilities as a cloud service by a service provider. Cloud-enabled BPM (CE-BPM) platform refers to a BPM platform product that has the capabilities to address the five attributes of cloud computing. A bpmPaaS and CE-BPM platform usage scenarios include integrating business processes that span applications, as well as supporting the development and integration of new process-centric applications. In some cases, a bpmPaaS may be considered to also be an iPaaS or an aPaaS [19].

Example providers of bpmPaaS cloud are: AppPoint AppsonAzure or Intalio PaaS.

Although the external appearance of one PaaS offering might differ from that on figure 3.1 of another, and although the functional responsibilities of different PaaS offerings may be different as well — all PaaS offerings require both the performance and the cloud foundation technologies [17].

3.1.4 Cloud Foundation

The following are the features contained within the Cloud Foundation block of the Gartner reference model.

- **Shared resources.** It is a complex task to share physical computing resources (such as memory, CPU cores, threads, execution priorities, and database and network gateways) between independent application instances, while retaining full logical isolation between them. The more sharing there is, the more challenging the isolation; however, the more effective the use of resources is [17].
- **Multitenancy.** Tenants are represented in software as logical application instances. There can be multiple instances of the same application, instances of different applications or both, that are managed in a shared environment (see "Understanding Tenancy: Salesforce.com Versus Google.com"). A multitenant system keeps track of activities by tenant; enforces resource sharing between tenants; and provides tenant-segregated resource allocation, security, SLA enforcement, billing, logging, backup, error recovery and more [17].
- **Elasticity.** Elastic PaaS allocates (with some degree of automation) the optimal amount of computing resources to an application instance, based on changing demand. It has sufficient resources to meet spikes in demand, but will activate minimal resources during a "quiet period." [17].

- **Self-service.** Most cloud services, including PaaS, are associated with some degree of self-service. This includes self-service provisioning, configuration, some monitoring and administration, and other functions — all of which are offered through a dedicated Web portal [17].
- **Continuous versioning.** A PaaS-based software service may be used by thousands of organizations around the world, and PaaS must be designed for 24/7 operations. Yet as with any software, it must make version upgrades and patches. A well-designed PaaS is enabled for continuous unobtrusive versioning in which updates are deployed into a running system in a manner that does not interrupt the operations of the running applications, tenants and transactions [17].
- **Use-based billing framework.** A multitenant PaaS includes the feature of tracking of activities and resource consumption by tenant. It is essential to the implementation of elastic allocation of resources: The system must know, which tenant runs out of capacity or utilizes excessive capacity before it can make normalizing adjustments. Tracking of tenant resource use is also used as the foundation for billing. The more advanced PaaS offerings will recognize the subtenants — tenants of their software vendor (ISV) partners — and will provide a framework for the ISVs to establish a subtenant billing process, based on the PaaS-provided tracking data [17].

3.1.5 Performance Foundation

The following are the features belonging to the Performance Foundation block of the Gartner reference model.

- **In-memory computing.** The in-memory data grid (once referred to as "distributed caching") provides resilient in-memory storage of massive amounts of data. This allows some entire databases to be contained in memory, eliminating disk input/output (I/O) from the critical path of most transactions, thus dramatically improving performance, scalability and throughput of the platform. Note that all current platform technologies have been designed to optimize against the bottleneck of the disk I/O. Eliminating this constraint is likely to lead to the emergence of all new platform architectures [17].
- **Parallelism.** Most modern servers are multicore, and most modern computing takes place over a grid of physical or virtual servers. Both multicore and grid architectures are best utilized through parallel computing in which a single task is partitioned into threads that can execute simultaneously on separate cores or servers, greatly improving the elapsed duration of a task and system scalability [17].

- **Grid clustering.** This is the ability for a platform to reside over a large pool of physical or virtual servers (grid) so that the loss of any server at any time might be handled and new servers may be added to the grid at any time — all without any detrimental effect on the running application [17].
- **Bidirectional policy-based autoscaling.** For any given application, to utilize a variable and optimized footprint of computing resources, the underlying platform must track the application execution (in isolation from other application instances that may be deployed in the same environment). It must also decide what the optimal footprint is at any given time, based on demand and based on a policy, and add or remove resources from the application space in real time and without any detrimental effect on the running application [17].
- **Continuous (high) availability.** This refers to uninterrupted availability of sufficient computing resources, which ensures the uninterrupted operations of applications, despite hardware and software failures and regional disasters (such as power loss) [17].
- **Security.** The application and its data are protected from intrusion by other applications sharing the same cloud environment [17].
- **Data and transaction integrity.** The consistency of an application's data must be protected in the context of multiple applications' transactions potentially accessing shared database instances at the same time [17].

3.2 PaaS Providers Description

In this part it was decided to evaluate the following providers and their solutions:

Vendor	Solutions to be compared
Microsoft	Windows Azure
Amazon	AWS Elastic Beanstalk
Google	App Engine
SalesForce.com	Force.com
SalesForce.com	Heroku
Vmware	CloudFoundry
RedHat	OpenShift

Table 3.1 PaaS offerings to be compared

3.2.1 Windows Azure

This platform-as-a-service offering from Microsoft, provides a strong full-featured platform for new cloud applications development, and migration of already existing enterprise application to the cloud. As of the time of this writing, Windows Azure is a strictly public offering, thus there also exists a Windows Azure Platform Appliance for on-premise use that is already in use by a short list of partners (i.e. Fujitsu, Dell, HP, eBay). Microsoft claims that Azure offers a support for programming languages such as Java, node.js, Python, .net, PHP, although in order to develop in Java one would need to create a Virtual Machine, which is simply an IaaS offering not a PaaS. Moreover, PHP developers typically use development tools (such as Zend Framework) and runtime services (such as Apache, MySQL) that are not part of the Azure platform.

Azure system consists of a number of elements. The Windows Azure Fabric Controller provides an auto-scaling and reliability and it manages memory resources and load balancing. The .NET Service Bus registers and connects applications together. The .NET Access Control identity providers include enterprise directories and Windows LiveID. Finally, the .NET Workflow allows construction and execution of workflow instances [10].

Some of usage examples are the following.

Company	T-Systems Multimedia Solutions, Germany
Number of employees	1000
Technology	Windows Azure, SQL Azure
Benefit	Reduced Costs(60%), Faster time-to-market, Increased Scalability, Simple & Efficient Development(50% faster)

Company	Fujitsu Services, United Kingdom
Number of employees	11400
Technology	Windows Azure, SQL Azure, SQL Azure Reporting, AppFabric
Benefit	Reduced costs (60%), Faster time-to-market (60%)

3.2.2 Google App Engine

App Engine is a pure PaaS cloud targeted exclusively at traditional web applications, enforcing an application structure of clean separation between a stateless computation tier and a stateful storage tier. The virtualization and the elasticity that are so visible in the IaaS model are almost completely invisible here. But they are a big part of the picture behind the scenes. One of the selling propositions of this model is its automatic elasticity in the face of capacity requirement changes. The App Engine programming languages are Python, Java and Go!. It works best for web applications and relies on the assumption of a request-reply structure, which assumes long periods of no CPU utilization (such as, human think time). Consequently, Google can and does severely ration CPU time for each request. App Engine's automatic scaling and high-availability mechanisms, and the proprietary Datastore data storage (built on BigTable) available to App Engine applications, all rely on these constraints. But if your application fits within those constraints, there is probably no faster and cheaper way to build an application that scales automatically and runs on the largest cloud on the planet.

App Engine is a strictly public offering of Google, although Google provides a Secure Data Connector. The Secure Data Connector (SDC) ensures that private data is securely accessible to the Google App Engine application. Thus this is called as mentioned in the previous chapter, a cloud bursting⁶.

3.2.3 AWS Elastic BeanStalk

According to Amazon, Elastic Beanstalk is an easier way to quickly deploy and manage applications in the AWS cloud. It allows to simply upload the application and for all the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring will be automatically handled. Amazon states that with Elastic Beanstalk, one retains full control over the AWS resources powering his application. If a client decides that they want to take over some (or all) of the elements of their infrastructure, they can do so seamlessly by using Elastic Beanstalk's management capabilities. In order to ensure easy portability of the application, Elastic Beanstalk is built using familiar software stacks such as the Apache HTTP Server for PHP, IIS 7.5 for .NET, and Apache Tomcat for Java. There is no

⁶ Description from <https://developers.google.com/secure-data-connector/docs/1.0/tutorials/appengine>

additional charge for Elastic Beanstalk, the payment is only for the AWS resources needed to store and run the applications [9].

Amazon avoids the term PaaS when describing its Elastic BeanStalk offering, this is understandable from the PR point of view as many PaaS providers take advantage of AWS IaaS offerings. Although Elastic BeanStalk is something more than just PaaS cloud, as it allows altering the infrastructure resources, it can be easily considered as PaaS cloud.

3.2.4 Force.com

In combination with the Salesforce.com service, this PaaS cloud allows developers to create add-on functionality that integrates into the main Salesforce CRM SaaS application.

Force.com offers two ways of creating applications that can be deployed on its SaaS platform i.e. a hosted Apex or Visualforce application.

Apex is a proprietary similar to Java language that can be used for creating Salesforce applications. Visualforce, though, is an XML-like syntax with purpose of building UIs in HTML, AJAX, or Flex to overlay over the Salesforce hosted CRM system [10].

3.2.5 Heroku

Heroku is PaaS offering acquired by Salesforce.com, as such it is their second offering of PaaS cloud next to Force.com. Originally it was a platform for instant deployment of Ruby on Rails Web applications, although with time Salesforce.com broadened its offering by Node.js, Clojure, Java, Python, and Scala. Heroku background infrastructure is EC2 from Amazon. Thus those servers are invisibly managed by the platform and never exposed to users [10].

3.2.6 CloudFoundry

VMware is aggressively developing an open source PaaS offering- Cloud Foundry -that is to be another PaaS offering from VMware next to vFabric Cloud Application Platform (leading on-premise PaaS cloud solution for Java applications). At the time of this writing, CloudFoundry was in its Beta release, and as such it is offered to developers free of charge. Moreover it is possible for developers to download and run locally as the source code is accessible to anyone via a very lenient Apache 2 open source license. VMware also plans to integrate all of vFabric's commercial offerings into its open source PaaS. Cloud Foundry is built on a portable layer, so developers can run it on whatever cloud they like, using their choice of developer frameworks and application interfaces services, it also supports multiple application frameworks and languages including Spring, Grails and Scala for the Java community, Rails and Sinatra for the Ruby world, and node.js.

According to the VMware executives, the company does not want CloudFoundry.com to become a major PaaS force to equal Google App Engine, Red Hat's OpenShift or Microsoft's Azure. Instead, it is aiming for the Cloud Foundry code distribution to be adopted widely and hosted on other infrastructures [12].

3.2.7 OpenShift

Red Hat's OpenShift PaaS provides developers with a range of open source languages, tools, application frameworks, and data management support for building, testing, running and managing their applications. Red Hat claims that OpenShift is the first PaaS to run CDI (Contexts and Dependency Injection) applications, which has Java EE 6 support. Based on Red Hat's Deltacloud API, OpenShift PaaS will enable developers to write applications that can run on any Red Hat Certified Public Cloud, including Rackspace and Amazon. OpenShift PaaS comes in two flavors: Free Shift (free) and Mega Shift (paid). Supported technologies are Perl, Ruby, PHP, Python, and Java EE6, the OpenShift environment also supports the "Do it yourself" or "DIY" application type. Using this application type, a client can run just about any program that speaks HTTP [20]. Current Red Hat Certified Cloud Provider Partners include Amazon, Fujitsu, IBM and Savvis. At the time of this writing, OpenShift runs on RHEL running on top of VM instances from Amazon's IaaS offering, in particular EC2.

3.3 Comparative Table

The before described PaaS cloud offerings are very broad, and differ in many aspects. As such in order to facilitate the comparison of those major provider offerings it was decided to create a comparative table presented below.

	Possible Deployment	Programming Language Frameworks	Developer Tools	Target Use	Persistence Options	Backend Infrastructure Providers
App Engine	Public, Hybrid (Secure Data Connector)	Python, Java, Go!	Eclipse-based IDE, Development Console	Web Applications	BigTable, Cloud SQL	Own data centers
Force.com	Public	Apex	Eclipse-based IDE, Webbased console	Enterprise applications	Own object database	Own data centers
Windows Azure	Public, Hybrid, Private	.NET, node.js, Python, PHP	Azure tools for Microsoft Visual Studio	Enterprise and Web applications	Table/BLOB/queue storage, SQL services	Own Data Centers
Heroku	Public	Ruby on Rails, Java, Node.js, Scala, Clojure	Command-line tools	Web applications	PostgreSQL, Amazon RDS	Amazon EC2
Amazon Beanstalk	Public	Java, .NET, PHP	AWS Management Console, AWS Toolkit for Eclipse,	Web applications	Amazon RDS, Amazon DynamoDB, Amazon SimpleDB	Own Data Centers
Red Hat OpenShift	Public, Hybrid, Private	Java, PHP, Python, Perl, Ruby, Node.js	JBoss Developer Studio, RHC Client Tools	Web Applications	MySQL Database, MongoDB NoSQL Database, PostgreSQL Database	Amazon EC2
CloudFoundry	Public, Hybrid, Private	PHP, Python, .NET, Spring Java, Play, Rails and Sinatra for Ruby, Node.js, Groovy, and Grails	VMC-command-line interface, Extension to Eclipse or SpringSource Tool Suite (STS), Micro CloudFoundry	Web Applications	vFabric Postgres, MongoDB, Redis	Own Data Centers, vSphere

Table 3.1 Comparative table of major PaaS cloud offerings

3.3.1 Possible Deployments

All three possible deployment types were described in the previous chapter i.e. public, hybrid, and private. Thus the table 3.1 points out the applicable types for each of the providers' offerings. App engine is a public offering with a capability of cloud bursting, which allows for taking advantage of Google PaaS cloud also for hybrid environment.

On the other side Amazons' Beanstalk at the time of this writing does not enable VPC that would allow for a step towards hybrid deployment, currently this option is only available for EC2- Amazon's IaaS cloud. The same as in case of the latter offering Salesforce.com's Force.com and Heroku platforms are strictly public offerings where Force.com has a very interesting new feature that allows for interconnecting it with Google App Engine and it is called "Force.com for Google App Engine". What it does is it basically lets developers create web and business applications that span both salesforce.com and Google's cloud computing platforms, and take advantage of the key features of both. Force.com for Google App Engine

provides Java and Python libraries that allow accessing the Force.com Web Services API from within Google App Engine applications [15]. Very late

CloudFoundry and OpenShift allow for any type of their deployment as both are open source and can be either provided on company premises or can serve as a base platform for PaaS offerings of the 3rd party provider (e.g. CloudFoundry based AppFog).

Microsoft's Windows Azure provides any possible deployment type, although excluding on-premise private cloud that is not yet available for commercial use, though some big customers such as Fujitsu are already trying out this Microsoft Solution. Windows Azure offers hybrid deployment through its Service Bus Relay. It facilitates this by enabling for secure exposing of Windows Communication Foundation (WCF) services that reside within a corporate enterprise network to the public cloud, without having to open up a firewall connection or requiring intrusive changes to a corporate network infrastructure [16].

3.3.2 Programming Language Frameworks

Although Google App Engine offers support for three programming languages, only Python can be perceived as the one fully supported (it was the first language offered by the platform). Thus Java has many limitations, Go! at the time of this writing is still in its beta version. More on Java support can be found in the next chapter.

Force.com provides its proprietary language Apex, a Java-like programming language and Visualforce, an XML-like syntax for building user interfaces in HTML, Ajax or Flex. In comparison to Force.com Heroku gives a wider range of supported languages with the first supported one being Ruby on Rails, as the platform itself is also written in that language.

Windows Azure offers support for many different environments, including Java, although as it was already mentioned Java is not exactly a PaaS offering. Of course the main offering of Microsoft's PaaS cloud is its .Net support, though Microsoft is aggressively widening its offer with support for new languages.

VMware calls its CloudFoundry an open PaaS. The purpose of that is that VMware is aiming to provide a platform supporting all languages, running on any infrastructure. For now open PaaS from VMware seems to be only a promise for future as for now it does not provide support for all programming languages and frameworks.

Amazon's Beanstalk runs Apache HTTP Server for PHP, IIS 7.5 for .NET, and Apache Tomcat for Java.

The RedHat PaaS cloud offering OpenShift besides support for Perl, Ruby, PHP, Python, and Java EE6, supports the "Do it Yourself" or "DIY" application type, as it was already mentioned. By using this application type, one can run just about any program that "speaks" HTTP [20]. This works in the following way, the OpenShift execution environment is a carefully secured Red Hat Enterprise Linux 6.2 on x64 systems. Thus, OpenShift can run any binary that will run

on Red Hat Enterprise Linux 6.2 x64. The way that OpenShift DIY interfaces the application to the outside world is that creates a HTTP proxy specified by the environment variables `OPENSHIFT_INTERNAL_IP` and `OPENSHIFT_INTERNAL_PORT`. All the application has to do is bind and listen on that address and port. HTTP requests will come into the OpenShift environment, which will proxy those requests to the application. The application will reply with HTTP responses, and the OpenShift environment will relay those responses back to one's users [20].

3.3.3 Developer Tools

Microsoft, Amazon and Google provide a plug-in for IDE, which definitely makes the deployment and application development much easier. All of the PaaS providers mentioned in the table 3.1 provide tool that enable for easy development and testing local environment.

3.3.4 Backend Infrastructure

As shown in the table 3.1 some of PaaS cloud vendors work on their own infrastructure, while the other ones use AWS EC2 instances that enable them to provide their proper PaaS cloud service. In the case of RH OpenShift this solution is temporary as RedHat offering is in its very early stage, and their aim is to base OpenShift on their IaaS offering CloudForms in the future.

3.3.5 Persistence Options

The table 3.1 also shows different data-storage options. It can be observed that most of the offered database solutions are NoSQL databases. Thus the cause of that lies in the very nature of NoSQL database, which is more responsive than the traditional SQL Database, and allows for better scalability, which in case of cloud computing is crucial. Thus NoSQL type of database seems to be a better solution for large-scale global Internet applications [13].

Another benefit is that there are often no schemas associated with a table, which allows the database to adopt new business requirements [13]. Many projects over the years change their requirements. As this is rather hard to handle with traditional databases, NoSQL allows easy adoption of such requirements. Amazon could be a good example of such process. Thus they store a lot of data on their products as they are offered of many different types - such as personal computers, smartphones, music, home entertainment systems and books - they need a flexible database. This is a challenge for traditional databases. With NoSQL databases it is easy to implement some kind of inheritance hierarchy - just by calling the table "product" and letting every product have its own fields. Databases such as Amazon Dynamo handle this with key/value storage [13]. In order to demonstrate some of the NoSQL database features, the most popular databases of this type will be compared.

- **Amazon SimpleDB** - Amazon SimpleDB is a highly available and flexible non-relational data store that offloads the work of database administration. Developers simply store and query data items via web services requests and Amazon SimpleDB does the rest.
- **Google's BigTable** - Big Table is a compressed high performance database built on Google File System (GFS).
- **MongoDB** - MongoDB (from "humongous") is a scalable, high-performance, open source, document-oriented database. Written in C++.

CHARACTERISTICS	BIGTABLE	SIMPLEDB	Mongo DB
Programming language	C++, Python.	Erlang.	C++
Data Item	Multi-version with time stamp	Multi-value attribute	Multi-value attribute, BSON objects.
Schema	Column-families	No schema	No schema
Operation	Single-table scan with various filtering conditions	Range queries on arbitrary attributes of a table	Range based partitioning
Consistency	Single-row transaction	Eventual consistency	Eligible (any)
Scalability	Highly Scalable	Comparatively less Scalable	Highly Scalable
Purpose	Designed to scale massive amount of data	Designed to scale massive amount of data	Designed to scale massive amount of data
Database model	Column-oriented	Domain based	Document-oriented
Data storage	Distributed storage of structured data	Centralized storage of structured data	Either centralized or distributed storage of structured data.
Dimension	Single Dimension	Multi Dimension	Multi Dimension.
Integrity	Problem of Referential Integrity	Data integrity is not guaranteed	guaranteed through journaling and replication
Type	Reconfiguration is automatic	No need of reconfiguration	Reconfiguration is automatic
Usage	User doesn't need to learn any syntax and it is user friendly	User need to learn syntax and provides inconsistency	User needs to learn syntax
Cost	Structured data storage on Bigtable less than in SimpleDB	Structured data storage on SimpleDB costs more than in Bigtable	N/A
Features	Data Import, Export and back up are fast	Data Import, Export and back up are comparatively slow	There exists a tool to export and import data but it still has some bugs, geospatial indexing

Table 3.3 NoSQL database comparison

Flawless execution of many Web applications requires rigid data consistency. Although the properties of the Cloud like high scalability and availability make it an excellent platform to

host Web content, measurable cloud database services offer comparatively only weak consistency properties. The application needs consistency suitable for SimpleDB. Bigtable is the best fit for scalable and suitable storage of huge data [14]. Thus MongoDB seems to be a decent open source option.

3.4 Private PaaS

According to Gartner, 84% of enterprises will begin their cloud journey behind the firewall. While most enterprises have considerable interest in public cloud, viewing it as a significant part of their future, they are constrained by compliance, budget access, sunk-cost infrastructure and existing IT bureaucracies.

Beyond the fact that private clouds give organizations more control, the whole idea of the private cloud is to create a flexible IT environment where the cost of managing that environment scales up and down as required [18]. That is not easy to accomplish when companies are paying licensing fees for commercial application software based on the class of processor they are deployed on or the number of users who might be accessing an application during a specified time period. It may be the reason for so much interest these days in custom application development and open source software in the cloud.

The initial objective of many private cloud projects was to consolidate control over the proliferating networks of virtual machines (VM) and, in the more complex cases, improve hardware resource utilization through the dynamical allocation of VMs. The private PaaS objective is to [18]:

- Improve the agility of the enterprise computing infrastructure by expanding the logic of sharing computing resources to software stack higher levels.
- Improve resource utilization through more-advanced elasticity and multitenancy
- Reduce costs through these and other advanced technology patterns available with PaaS architecture.

In a private PaaS, multitenancy is applied primarily to multiple applications acting as tenants and competing for shared resources (in contrast to the public PaaS, where the tenants are often IT organizations). It is important to note that deploying a traditional application server, DBMS, ESB or another middleware product over private IaaS most of the time does not constitute a viable private PaaS. According to Gartner the expected benefits can only be achieved by using a cloud-enabled application infrastructure technology such as one that extends the resource sharing and optimization up the stack into the application container and application infrastructure. Placing a pre cloud middleware product over managed virtualization technology most of the time amounts to simply hosting that product on VMs, with minimal or no sharing of any resources between tenants [18].

Implementation of a private PaaS requires that the application infrastructure technology is inherently cloud-enabled i.e. It must embed support for multitenancy or, at least, be dynamically and elastically horizontally scalable. In the first case, multitenancy and elasticity

are implemented in the application infrastructure layer. In the second case, the elasticity is implemented in the IaaS layer by dynamically allocating VMs based on established policies, and the application infrastructure expands or contracts to follow the changing underlying resources, avoiding influencing the performance of the running applications [18]. In order to start a private cloud project Gartner recommends to focus on private IaaS, managing and optimizing the network of VMs, run traditional application infrastructure and traditional applications in the first stage of the private cloud to become aware of the costs and benefits of cloud computing incrementally, avoiding major discontinuities [18].

Then one should start experimenting with private PaaS by deploying horizontally scalable application infrastructure over elastically managed pools of VMs (shared-hardware PaaS architecture). Some of the traditional application servers and DBMSs are or will be available in the elastic clustering mode [18]. One should give preference to cloud-enabled application infrastructure products that have one or, even better, multiple public PaaSs using them. This will set all up for future expansion or transition to the public cloud with a minimum impact. Before moving to public or private PaaS, one should make sure that the IT environment is well-versed in SOA administration as adjusting to using cloud application resources is a lot like building SOA services [18].

3.5 Summary

Nowadays, the cloud industry is changing from one day to another, and that it is going to keep on changing for quite some time. Thus it is even more difficult to choose a platform that will fulfill the entirety of one's needs. When making the choice, it first has to be decided whether the solution one is looking for is the private, hybrid or public cloud. Although the choice is not easy, and there are many pros and cons to each of the deployment types, it seems that more and more companies will move steadily toward an enterprise hybrid cloud computing model as it can offer the best from both worlds: the security, quality of service, and control of an on-premise (private) cloud, combined with the agility and pay-per-use economics of off-premise (public) clouds.

Companies often fear vendors lock-in although it is a price that one has to pay for the benefits that come from using the cloud.

Public PaaS seems to be a very beneficial option as it also includes great business benefits such as time to market and small investment in comparison with the private PaaS. Out of many public PaaS offerings one should consider the ones with adequate features, for example:

- Possible programming languages
- Good Pricing
- Good development environment
- Easiness of application deployment
- Weak vendor lock-in
- Reliability

For the purposes of this thesis the programming language is Java, as at Everis, most of the projects are developed in that language. When it comes to pricing, the Google App Engine seems to be the most economic choice. It also allows for easy deployment and has a very useful plug-in for Eclipse that simplifies the process of development. Although an application developed for Google App Engine will need a bit of changing - depending on the application's complexity – when moved to another platform. When it comes to the reliability, App Engine is designed in such a way that it can sustain multiple datacenter outages without any downtime. This resilience to downtime is shown by the statistic that the High Replication Datastore saw 0% downtime over a period of a year. Moreover, all billed High-Replication Datastore App Engine applications have a 99.95% uptime SLA. Basing on those briefly explained advantages and the ones to be explained in the following chapter, the Google App Engine was chosen as a platform that will be given a deeper look.

4 Google App Engine

The Google App Engine was announced on April 2008. Since then the objective of this platform has been to enable developers to run and develop their software somewhere in the Google cloud. Being able to do so, anyone may feel as if they were Google employees having access to the entire scalable Google infrastructure.

One of the most important features of the Google App Engine is the fact that it is free for moderate levels of use. Every person that possesses a Gmail account can have a number of ten free applications running on the Google infrastructure and in case of one of them becoming very popular and the traffic going above the allowed levels of the free account, one can pay to use more of Google's resources. As the application scales, all the hardware, data storage, backup, and network provisioning for the customer are taken care of by Google's engineers. The payment for Google's resources is likely to be way lower than maintaining the same resources by the customer themselves. Google focuses on providing hardware and network, while the customer focuses on development and the user community around his application [4].

A general Google App Engine architecture is represented on the figure below.

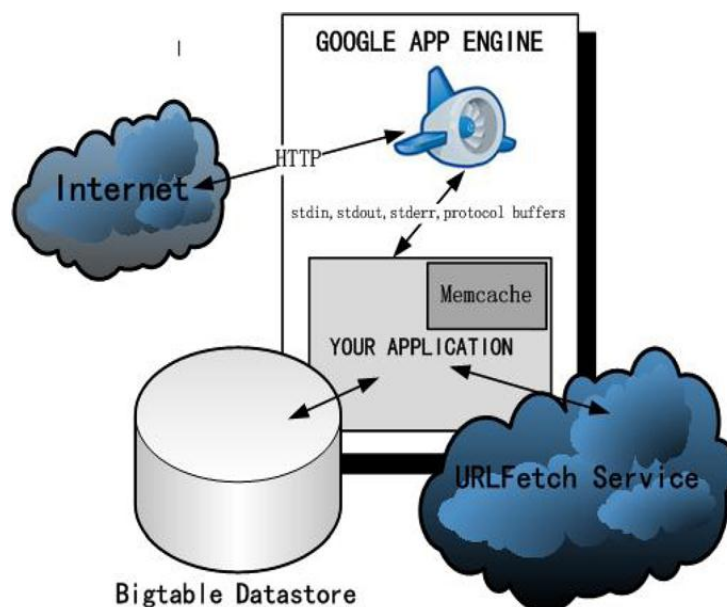


Figure 4.1 General view of Google App Engine Architecture [4]

The above figure represents a generalized view of GAE architecture, where Datastore, Services and Internet connection via HTTP protocols are the factors on which the GAE application is based on.

The following chapters are based on the Java Implementation.

4.1 Technology Support

The only three languages that are supported at the time of writing this thesis by GAE are Python, Java and Go!, which is still in its experimental release [1].

JAVA

Out of three supported languages only Java will be described in detail, as most of the projects developed at Everis are built in Java.

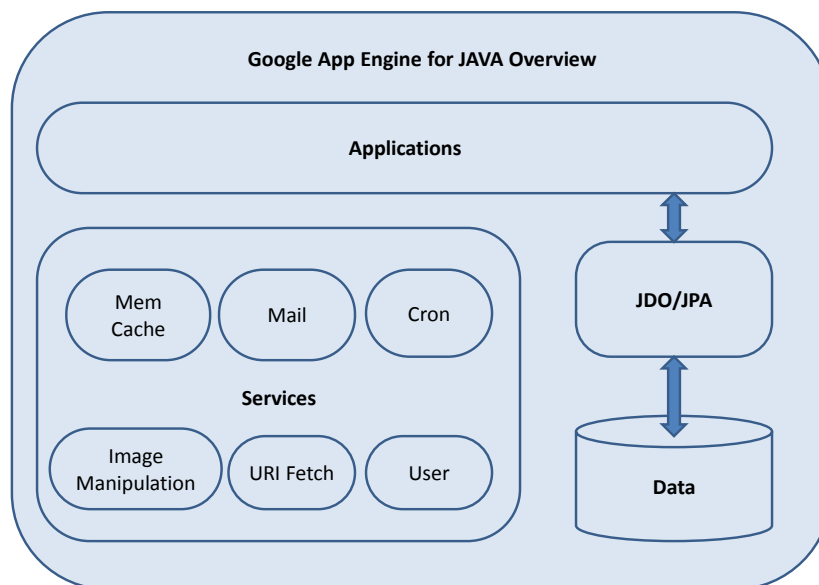


Figure 4.2 General view of Java Runtime environment on Google App Engine

Google App Engine executes Java applications using the Java 6 virtual machine (JVM). The App Engine SDK supports Java 5 and later, and the Java 6 JVM can use classes compiled with any version of the Java compiler up to Java 6.

The Java Servlet standard is a base for web applications in App Engine. There have to be app's servlet classes, JavaServer Pages (JSPs), data files, static files, along with the deployment descriptor (the web.xml file) and other configuration files provided in a traditional WAR directory structure. The requests in the App Engine are served by invoking servlets according to the deployment descriptor.

As for the runtime environment it is also worth mentioning that JVM runs in a so-called "sandbox". A sandbox can be seen as a secure container isolating threads from each other. A more detailed explanation from Google is cited below [1].

"The JVM runs in a secured "sandbox" environment to isolate your application for service and security. The sandbox ensures that apps can only perform actions that do not interfere with the

performance and scalability of other apps. For instance, an app cannot spawn threads in some ways, write data to the local file system or make arbitrary network connections. An app also cannot use JNI or other native code. The JVM can execute any Java bytecode that operates within the sandbox restrictions.”

Since the support for Java it has been possible to develop applications for GAE also in other languages that have to be JVM-compatible, such as PHP using Quercus, Ruby using JRuby, Groovy or Scala. All of those JVM-based languages are subject to the same App Engine sandbox restrictions as Java [4].

For specific information on classes that are GAE compatible there even exists “The JRE Class White List⁷”, which specifies all of the JRE classes that will run on Google App Engine.

Some of the GAE compatible Java frameworks and individual components are listed below [1]:

- Apache Struts
- Spring MVC
- Java Data Objects (JDO)
- Java Persistence API (JPA)
- Java Server Faces (JSF) 1.1 - 2.0
- Java Server Pages (JSP) + JSTL
- Java Servlet API 2.4
- JavaBeans™ Activation Framework (JAF)
- Java Architecture for XML Binding (JAXB)
- Java API for XML Web Services (JAX-WS)
- JavaMail
- XML processing APIs including DOM, SAX, and XSLT
- Since the release of Google Cloud SQL Hibernate is operational

It should be noticed that while Google does not support the entirety of the Java EE specification, many of its individual components are supported, although for some of them in order to work one may need a bit of a workaround.

There are many APIs and technologies that are not supported by Google App Engine yet. Thus the most important ones are listed below.

- Enterprise Java Beans (EJB)
- JAX-RPC
- Java Database Connectivity (JDBC)
- Java EE™ Connector Architecture (JCA)
- Java Management Extensions (JMX)
- Java Message Service (JMS)

⁷ <https://developers.google.com/appengine/docs/java/jrewhitelist>

- Java Naming and Directory Interface (JNDI)
- Remote Method Invocation (RMI)

App Engine also fully supports Google Web Toolkit (GWT), a framework for rich web applications that allow writing all of the applications' code—including the user interface—in the Java language.

The Java environment uses the following application server model: a request is routed to an app server; if necessary the application is started on the app server and invoked to handle the request to produce a response, and the response is returned to the client. The Java environment interpreter (the JVM) runs with sandbox restrictions, such that any attempt to use a feature of the language or a library that would demand access outside of the sandbox will fail with an exception. While using a different server for every request has scaling advantages, it is time consuming to run a new instance of the application for each and every request. Google App Engine mitigates startup costs by keeping the application in memory on an application server for as long as possible and reusing servers intelligently. When a server needs to reclaim resources, it purges the least recently used application. All app servers have the runtime environment (JVM) preloaded before the request reaches the server, so only the application itself needs to be loaded on a fresh server [4].

4.2 Google Database

Google owns its proprietary database which is called Bigtable. Bigtable is a distributed storage system entirely built by Google, having other Google services such as Scheduler, GFS, Lock Service and MapReduce⁸ as its base. Its purpose is to manage structured data that is designed to scale to a very large size. Bigtable uses the distributed Google File System (GFS) to store log and data files.

Various Google projects store data in Bigtable. Some of those projects are: Orkut, Google Earth or Google Analytics. In total, over sixty Google projects have Bigtable as their base. These applications place various demands in terms of latency requirements and data size, even though Bigtable has provided all of them with a high-performance, flexible solution.

BigTable is not a relational database and it does not support joins or rich SQL-like queries. Basically speaking, it is a map of maps or hash of hashes. Each table is a multidimensional sparse map that consists of rows and columns, and each cell has a time stamp. The timestamps can be used for example to delete items older than date/time.

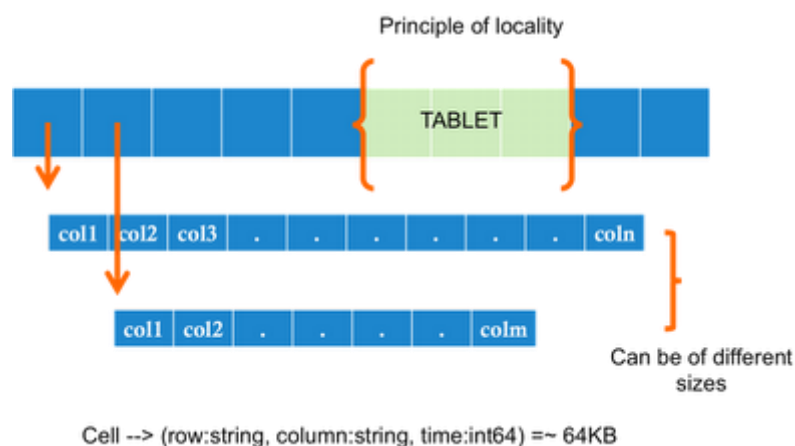


Figure 4.3 BigTable visualization

In order to manage the huge tables, Bigtable splits tables at row boundaries and saves them as tablets. A tablet's size is around 200 MB, and each machine saves about 100 tablets. This way tablets from a single table can be spread among many servers, also if one table is receiving many queries, it can shed other tablets or move the busy table to another machine that has less workload. Moreover, if a machine goes down, a tablet may be passed on to many other servers so that the performance impact on any given machine is minimal [5].

Also worth mentioning are Google proprietary compression techniques (BMDiff and Zippy). Those techniques are used whenever a machine -where immutable Sorted Strings Tables (SSTables) and logs are stored – runs out of system memory. Basic compactions involve only a

⁸ MapReduce is a programming model for processing large data sets, and the name of an implementation of the model by Google. MapReduce is typically used to do distributed computing on clusters of computers.

few tablets, while more complex compactions involve the whole table system and recover hard-disk space [5].

The locations of Bigtable tablets are stored in cells. The lookup of any particular tablet is handled by a three-tiered system [5].

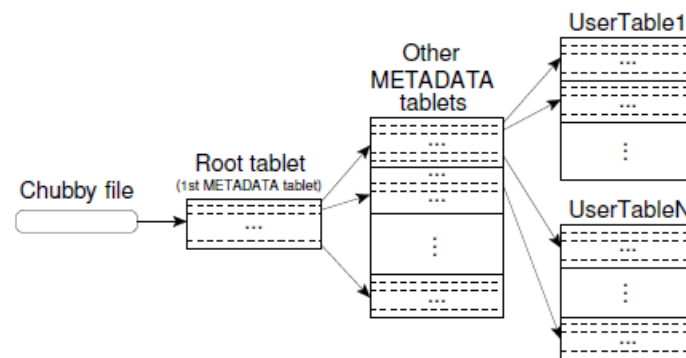


Figure 4.4 Three-tiered architecture for locating the tablet [5]

The 1st level is a file stored in Chubby⁹ file that contains the location of the root tablet. The root tablet contains the location of all tablets in a special METADATA table.

Each METADATA tablet contains the location of a set of user tablets. The root tablet is just the first tablet in the METADATA table, but is treated in a special way i.e. it is never split to ensure that the tablet location hierarchy does not have more than three levels. The METADATA table stores the location of a tablet under a row key that is an encoding of the tablet's table. Each METADATA row stores approximately 1KB of data in memory. With a modest limit of 128 MB METADATA tablets, the three-level location scheme is sufficient to address 234 tablets (or 261 bytes in 128 MB tablets). The client library caches tablet locations. If the client does not know the location of a tablet, or if it discovers that a cached location information is incorrect, it recursively moves up the tablet location hierarchy [5].

4.2.1 Datastore

Google Datastore for App Engine is a very small and only part of BigTable that Google shares with common users.

Google App Engine's abstraction for data is easy to comprehend, but it is not clear how to take advantage of its features. In particular, it is surprisingly different from the kind of database that most developers are familiar with, the relational database. It differs so much that Google does not even call it a "database", but a "Datastore".

Datastore overview according to Google [3]:

⁹ Chubby is a fault-tolerant system at Google that provides a distributed locking mechanism and stores small files. Typically there is one Chubby instance, or "cell", per data center.

“The App Engine Datastore is a schemaless object datastore providing robust, scalable storage for your web application, with no planned downtime, atomic transactions, high availability of reads and writes, strong consistency for reads and ancestor queries, and eventual consistency for all other queries. The Java Datastore SDK includes implementations of the Java Data Objects (JDO) and Java Persistence API (JPA) interfaces, as well as a low-level Datastore API.”

Summarizing the strengths:

- Managed by someone else (maintenance, backups, capacity, hardware failures).
- Schema-less. Committing structural changes to the data on-the-fly with no downtime.
- Scalability. Whether the application has 10 or 10 million users the performance will stay the same.

The main weaknesses:

- Transactions are very limited and the data model must be structured to use them in a particular way from the start.
- Many Java libraries are incompatible (no SOAP RPC implementation).
- Simple set of features

Datastore should be perceived as an object database, where each object is known as an entity. An entity has a unique key that identifies it thorough all the system. Fetching an entity by key is the fastest way of retrieving it. Data for the entity is stored in one or more properties. Each property has a name and one or more values. Each value is of one of several supported data types, such as a string, an integer, a date-time, or a null value.

The queries of Google’s Datastore operate on each of the determined entity types. They establish none or various filters for the keys and the property values. In case that one of the determined entities includes at least one value (possibly null) for each of the filter properties, and all of them coincide with query criteria the entity is returned as a result.

The Datastore queries are based on indexes, that is, a table with the potential resulting query in a desired order. Thus no further computations are needed to obtain the correct results of all queries, as they are immediately available directly from the indexes. Additional indexes usually can be defined in a datastore-indexes.xml file, also by annotations, and for some of the queries indexes are provided automatically. The development web server aggregates suggestions to the configuration file automatically when it encounters queries that do not have configured indexes yet [3].

The table below describes the eight indexable types supported by Datastore, they are listed in their relative order [4].

Data Type	Java Type	Ordering
Null value	Null	-
Integer and date/time	long(other integer types are widened), java.util.Date, datastore.Rating	Numeric(date/time is chronological)
Boolean	Boolean(true or false)	False then true
Byte string db.ByteString	datastore.ShortBlob	Byte order
Unicode string	Java.lang.String datastore.Category datastore.Email datastore.IMHandle datastore.Link, datastore.PhoneNumber datastore.PostalAddress	
Geographical point	datastore.GeoPt	By latitude, then longitude(floating-point numbers)
A Google Account	users.User	By email address, Unicode order
Entity key	datastore.Key	Kind (byte string), then ID (numeric) or name (byte string)

Table 4.1 How the Datastore value types are sorted

GAE provides also transaction capabilities. Many applications need data integrity guarantees when performing a set of various operations, over multiple units of data. Such a set of operations is called a transaction. Transaction can totally fail or totally succeed. Transactions are fully optional, and in order to make operations one does not need to use them.

4.2.1.1 Data Repository

On January 5th 2011 Google announced the availability of a new Datastore configuration, that is, the High Replication Datastore (HRD). Before that, Datastore was based on a Master/Slave replication topology, but since Google was struggling with its reliability they had to change the approach. The main benefit provided by HRD is the number of data centers that maintain replicas of one's data is increased by using the Paxos algorithm that synchronizes the data in real time across Google datacenters [3].

4.2.1.2 Datastore Interfaces

As already mentioned, Java SDK includes implementations of JDO, JPA, and low-level Datastore API. However, the Java SDK also supports other frameworks designed to simplify Datastore usage for Java developers.

Objectify is a very useful and simple interface that eliminates some of the complexities of JDO/JPA and Datastore API

Twig is an interface to persist the configurable objects that improves support for inheritance, polymorphism, and generic types. Like Objectify, it also helps to avoid complexities of JDO/JPA or low level Datastore.

Slim3 is a full-stack model-view-controller framework that can be used for a wide variety of App Engine functions, including the Datastore.

Although it may seem that the use of JDO/JPA interfaces or Datastore API should be encouraged (as it is the original implementation), the reality shows different, and the above-mentioned frameworks can be seen as an answer to those extremely complex and incomplete solutions provided by Google. Jeff Schnitzer the creator of Objectify asked about the problems with the JDO/JPA solutions said [6]:

“The biggest problem is that they're clumsy and unwieldy. JDO contains a bazillion annotations, configuration options, and query language structures to allow it to straddle every conceivable kind of datastore, from an RDBMS to a object databases. JPA is tailored specifically to the feature set of an RDBMS. The GAE datastore is something new... so JDO gets a whole new arsenal of special extensions and JPA just gets left out in the cold.

*In the context of GAE, 80% of JDO and JPA are meaningless cruft, and the GAE-specific features that you *do* need have tortured syntax. Do you really want to type things like @Extension(vendorName = "datanucleus", key = "gae.unindexed", value="true") everywhere? The syntax for batch gets - one of the most important operations on the datastore - is atrocious:*

```
Query q = pm.newQuery("select from " + Book.class.getName() + "
where :keys.contains(key)");
```

Using JDO/JPA on GAE effectively requires climbing not only the learning curve for JDO/JPA (and figuring out which 20% works), but also learning how the datastore works and how to bend JDO/JPA around it. It's just not worth it - and I say this as someone who has been working extensively with Hibernate and JPA since the days of Hibernate 1.0.

Aside from the general complexity issue, there are some specific problems with JDO/JPA on GAE. GAE has some interesting nuances (parent keys & entity groups, partial indexes, collection properties) that don't have direct mappings in either JDO or JPA. JPA is particularly troublesome; DataNucleus uses the bytecode enhancer, but JPA doesn't have a detach() method - so you can't serialize your entities, ever!”

Summarizing, Datastore is based on BigTable, not a traditional SQL-based RDBMS like MySQL or PostgreSQL. It is possible to use the low-level datastore API, JDO or JPA object-relational mapping interfaces provided. Many JDBC wrappers are available for the Datastore, and it is still possible to connect to in-memory database such as the H2 database engine or HSQLDB.

4.2.2 Cloud SQL

Cloud SQL is a brand new database solution for app engine applications. In opposite to Datastore, Cloud SQL is very similar to traditional MySQL Database. It is fully relational, moreover, it allows to import from and export to Google Cloud, current use cases. Below quoted, is the description from the producer [8].

“Google Cloud SQL is web service that allows you to create, configure, and use relational databases with your App Engine applications. It is a fully-managed service that maintains, manages, and administers your databases, allowing you to focus on your applications and services.

By offering the capabilities of a MySQL database, the service enables you to easily move your data, applications, and services into and out of the cloud. This allows for high data portability and helps in faster time-to-market because you can quickly leverage your existing database (using JDBC and/or DB-API) in your App Engine application.

To ensure that your critical applications and services are always running, Google Cloud SQL replicates data to multiple geographic regions to provide high data availability.” Google also lists the main features and restrictions of their SQL cloud [8].

4.2.2.1 Features

- Ability to host MySQL databases in the cloud
- Instances up to 10GB
- Synchronous geographic replication
- Import or export databases using mysqldump
- Java and Python compatibility
- Command-line tool
- SQL prompt in the Google APIs Console

4.2.2.2 Restrictions

- Size limit for individual instances is 10GB
- User defined functions are not supported
- MySQL replication is not supported
- The following MySQL statements are not supported:

```
LOAD DATA INFILE  
SELECT ... INTO OUTFILE  
SELECT ... INTO DUMPFILE  
INSTALL PLUGIN ...  
UNINSTALL PLUGIN  
CREATE FUNCTION ...  
LOAD_FILE()
```

Since June 12th 2012 Google charges for the use of Google Cloud SQL. There are provided two billing plans: Packages and Per Use¹⁰.

¹⁰ Pricing can be found at <https://developers.google.com/cloud-sql/docs/billing>

4.3 Integrated Services

What makes Google App Engine so versatile are its various APIs. For Java Runtime SDK 1.6.5 version the integrated services are the following [2]:

- App Identity
- Blobstore
- Google Cloud Storage
- Capabilities
- Channel
- Conversion
- Images
- Log Service
- Mail
- Memcache
- Multitenancy
- OAuth
- Prospective Search
- Search
- Task Queues
- URL Fetch
- Users
- XMPP
- Cron Service

4.3.1 App Identity

Along with the popularity of cloud computing grows the need of cloud's interoperability i.e. authentication and communication of an application hosted on one cloud service with other cloud application or service.

The Application Identity API seems to be one possible solution. It provides an application with public key infrastructure managed by Google. Every single App Engine application has its unique identity in the form of an email address and public/private key pairs, which are rotated every few hours. The API allows obtaining the current public key, signing an object with use of the private key, or to discover its own identity [2].

At the time of writing this description App Identity service is in its experimental version.

4.3.2 Blobstore

Blobstore gives the application a possibility to store bigger data objects called BLOBs (Binary Large Objects). Blobs are objects that are much larger than the objects allowed in the Datastore service such as videos or images. They are created through uploading a file via an HTTP request. Once a blob is uploaded using an opaque reference, called a blob key, can retrieve it. There are two ways of serving such blob, i.e. the complete blob value upon a user request or reading the value via a file-like interface. After their creation blobs cannot be modified, although they can be deleted. For each blob there is a single record stored in the Datastore, which provides more information about the blob i.e. content and when it was created. An application can read only a part of a Blobstore value at a time. The maximum size of such part is a bit less than 32 MB [2].

4.3.3 Google Cloud Storage

The Google Cloud Storage API allows reading and writing files to Google Cloud Storage. The previously described Blobstore API can also serve those files. Moreover, features that are provided by Google Cloud Storage while not being provided by Datastore nor Blobstore are [2]:

- Access Control Lists(ACLs) to manage data access
This feature allows controlling access to Google Cloud Storage objects and buckets¹¹. An ACL is made of one or more entries, and every entry grants permissions to a scope.
- POST policy support
This feature defines what the user can upload with a POST form. It also provides authorization for making sure that the form can upload files into the bucket.
- OAuth 2.0 authentication and authorization
Auth 2.0 is a relatively simple protocol used in many App Engine APIs. A developer can integrate with Google's OAuth 2.0 endpoints without too much effort.
- A RESTful¹² API
This feature means that Google Cloud Storage API is based on method information and scoping information to define the operations that are going to be performed.

Google Cloud Storage supports the following HTTP methods:

- *GET Service*—lists all of the buckets that you own.
- *PUT Bucket*—creates a bucket and changes the permissions on a bucket.
- *GET Bucket*—lists the contents of a bucket or retrieves the ACLs that are applied to a bucket.
- *DELETE Bucket*—deletes an empty bucket.

¹¹ Buckets are the basic containers that hold data.

¹² A RESTful API is a service implemented using HTTP and the principles of REST

- *GET Object*—downloads an object or retrieves the ACLs that are applied to an object.
- *PUT Object*—uploads an object or applies new ACLs to an object.
- *DELETE Object*—deletes an object.
- *HEAD Object*—lists the metadata for an object.
- *POST Object*—uploads an object by using HTML forms.

4.3.4 Capabilities

This API is useful for detection of outages and scheduled downtime for particular API capabilities. It allows predicting when a capability becomes unavailable and as such one can disable it before it has a direct impact on the users of the application.

4.3.5 Channel

This API enables to communicate directly with user browsers. It pushes notifications directly to the JavaScript on the client-side, eliminating the need for polling. This service makes it less complex to build real-time applications such as multi-player games or any collaboration centric app. It is built on the same Google infrastructure that powers Google Talk.

4.3.6 Conversion

A simple API that allows converting text to PDF. It also has an option to perform optical character recognition (OCR). At the time of writing it is still in its experimental release.

4.3.7 Images

The image API provides the possibility of manipulating the image data. It allows resizing, cropping, rotating or flipping images. There also exists a possibility to enhance a photograph via a predefined algorithm.

4.3.8 Log Service

This API provides access to the application logs, and request logs.

4.3.9 Mail

This API allows the application for sending email messages on behalf of the application's administrators and users with Google Accounts. It supports the JavaMail interface for sending email messages.

4.3.10 Memcache

High-Performance applications usually take advantage of a memory cache. In most cases, a memory cache uses the RAM of the cache machine, for very fast read and for write access to values. Due to the fact that memory is volatile the cache is not useful for long-term storage neither short-term primary storage for important data, though it can be excellent as a secondary system for quick access to data even from the Datastore. The App Engine memcache stores key-value pairs that can be set with a key and get the value by using the key. The maximum size of a value can be up to 1MB. A key can have up to 250 bytes, whether the API accepts even larger keys using a hash algorithm to convert them to the allowed 250 bytes. Besides the low-level API App Engine also provides the JCache interface, a standard proposed by JSR 107. Google recommends using low-level API as the JSR 107 is not a ratified standard and the JCache has a very limited usage [2].

The interaction with the service is as simple as working with the MemcacheService object. This object is obtained by simply getting it from MemcacheServiceFactory.getMemcacheService(). The basic operation of the memcache service are: put(), get() and delete(). Although memcache service does not support transactions as Datastore does, it includes methods for setting, fetching or deleting multiple items in a single batch call, which is faster than making one service call per item. The maximum size of such call is limited to one megabyte [2].

4.3.11 Multitenancy

Multitenancy is a software architecture that allows multiple client organizations/tenants to run the same instance of an application. Google achieves this functionality via Namespaces API, which allows for segregating data by using a unique namespace for each client. The Namespaces API facilitates data partitioning across tenants just by specifying a unique namespace string for each tenant. Instead of setting tenants explicitly for a specific request each one of them is specified in the namespace manager. As it is also integrated with Google Apps it allows using one's Google Apps domain as the current namespace. The API is also designed to be very customizable, with hooks into the code that can be controlled so the multi-tenancy can be set up in any chosen way.

In terms of Google App Engine namespace is the same as tenant.

4.3.12 OAuth

OAuth is a protocol that enables a user to obtain a permit to access a web application in his name for a limited time without having to share his private data (username and password) with the third party. That third party could be a web application or any other application capable of invoking a web browser for the user.

4.3.13 Prospective Search

Sometimes the website visitors are interested only in the data when newly entered records meet certain criteria. The Prospective Search API is a perfect tool in such cases.

The Prospective Search API allows registering a query on data that has not been entered into the system yet. As soon as a new record is added that meets the search criteria of the registered query, a specific task in the task queue is triggered to take action.

4.3.14 Search

The Search API gives the application a possibility of Google-like searches over structured data within the application. Full-text searches can be done through many different types of text, such as plain text or HTML. The result of such search is a ranked list of matching text and it can be customized.

4.3.15 Task Queues

Task queue API is a very powerful tool for background work. It allows for initiating a request that will make applications perform their work outside of that request. Furthermore, it allows for organizing the work into small tasks. Those tasks are then inserted into at least one queue and processed in FIFO order. A high-level task insertion into the queue can be illustrated as in the following diagram.

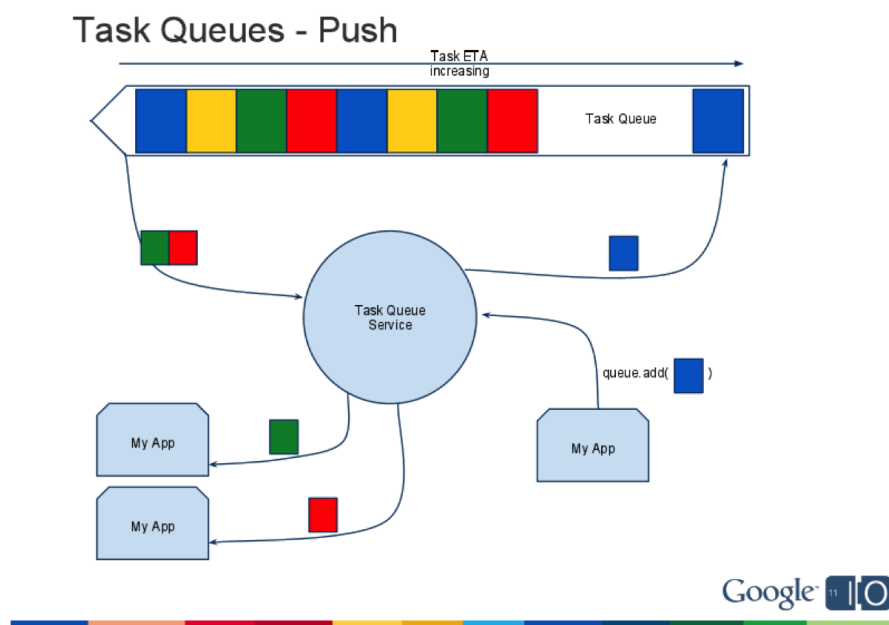


Figure 4.6 Conceptual presentation of inserting a task into the queue¹³

¹³Image from http://static.googleusercontent.com/external_content/untrusted_dlcp/www.google.com/en/_/events/io/2011/static/presofiles/putting_task_queues_to_work.pdf

There are two different queue configurations provided by GAE:

- **Push Queues**

Push queue processes the tasks depending on the processing rate that is configured in the queue definition. The lifetime of those queues is set automatically by the App Engine that also adjusts the processing capacity to match the configuration and processing volume.

- **Pull Queues**

Pull queues allow a task consumer to lease tasks at a specific time within a specific timeframe. They can be accessed internally as well as externally through the task queue REST API. Pull queues give more control over when tasks are processed. In order to use pull queues, the application needs to handle scaling of instances based on processing volume, and it also needs to delete tasks after processing.

4.3.16 URL Fetch

URL Fetch service allows Google App applications to communicate with other Internet hosts via HTTP and HTTPS requests. It supports `java.net.URLConnection` from the Java standard library.

4.3.17 Users

The purpose of the Users service is the authentications of users with Google Accounts, accounts on their own Google Apps domain, or OpenID identifiers. It allows the application for functionalities such as detecting whether the current user is signed in.

4.3.18 XMPP

This service allows applications to send and receive chat messages to and from any XMPP-compatible chat messaging service, such as Google Talk. Functionalities provided by XMPP Java API are the following:

- Send/receive chat messages
- Send chat invites
- Request a user's chat presence and status
- Provide a chat status

Incoming messages are handled similarly to web requests, request handlers.

4.3.19 App Engine Cron Service

The Cron Service gives a possibility of task scheduling. Such tasks can be scheduled for execution at a defined time or every given time (regular intervals). A cron job invokes a URL, using an HTTP GET request at a given time. The maximum runtime of such request can take up to 10 minutes. Free applications can have a maximum of 20 scheduled tasks, while the paid ones up to 100. Configuring a cron job is as simple as defining it in a cron.yaml file following the YAML¹⁴ format.

Some of the previously explained services that will be used in developing the pilot application are pointed out in the table below.

App Identity	
Blobstore	
Google Cloud Storage	
Capabilities	x
Channel	x
Conversion	
Images	
Mail	x
Memcache	x
Multitenancy	
OAuth	
Prospective Search	
Search	
Task Queues	
URL Fetch	x
Users	x
XMPP	x

Table 4.2 Services used in sample application

¹⁴ Data serialization standard for all programming languages

4.4 Tools

Among many other things Google provides a set of useful tools to maintain, upload, test or develop an application in offline App Engine environment.

- Development Server
- Google Plugin for Eclipse
- Local Unit Testing

4.4.1 Development Server

The App Engine Java SDK consists of a development web server with a purpose of testing the application without a need of uploading it to the online App Engine environment. The development server simulates the App Engine Java runtime environment with all of its services and Datastore included. Although the development server is supposed to function the same way as the App Engine Java runtime environment, it is still quite common that some functionality that will work offline in its destination environment will simply not work. Similarly to the App Engine Java runtime environment the Development server provides a console with most of the services useful for testing. When the development environment is running it is possible to access the Development console just by entering the URL (by default: `http://localhost:8888/_ah/admin`).

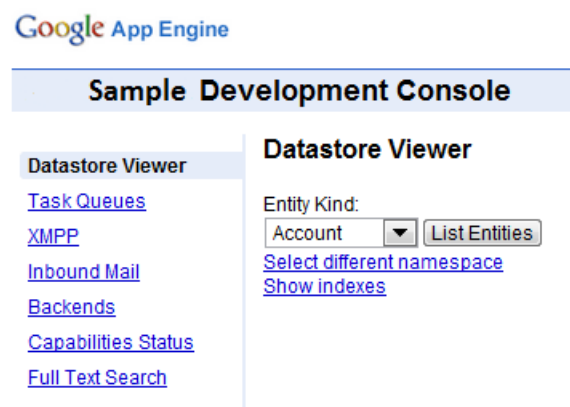


Figure 4.7 Main Console view

The above is a snapshot of a Development Console that appears after entering `http://localhost:8888/_ah/admin`. Functionalities that will be given a brief description one by one in this chapter are listed on the left.

4.4.2 Datastore Viewer

Datastore Viewer allows for most operations of its online version, although with some limitations.

- **Firstly**

There is a dropdown list with entity kinds. After selecting one of the entity kinds, the console displays all the entities of the chosen kind. Such table can be seen below.

SDK v1.6.5

Development Console

Datastore Viewer

Entity Kind: Account List Entities Select different namespace Show indexes Results 1 - 4 of 4

<input type="checkbox"/>	Key	Write Ops	ID/Name	accType	date	money	nickname
<input type="checkbox"/>	agx0b250b3N0dXRlc3RjDQsSB0FjY291bnQYJGw	10	37		Wed May 02 00:00:00 CEST 2012	33.0	test@example.com

Delete 1

Figure 4.8 View of the Datastore viewer

Development Console gives the option to delete any chosen entries. The Key, Write Ops, ID/Name are always existent.

- **Secondly**

Show Indexes option is also worth mentioning. After choosing it the indexes for the application are displayed with their status (which can be Error/Serving/Loading)

4.4.3 Task Queues

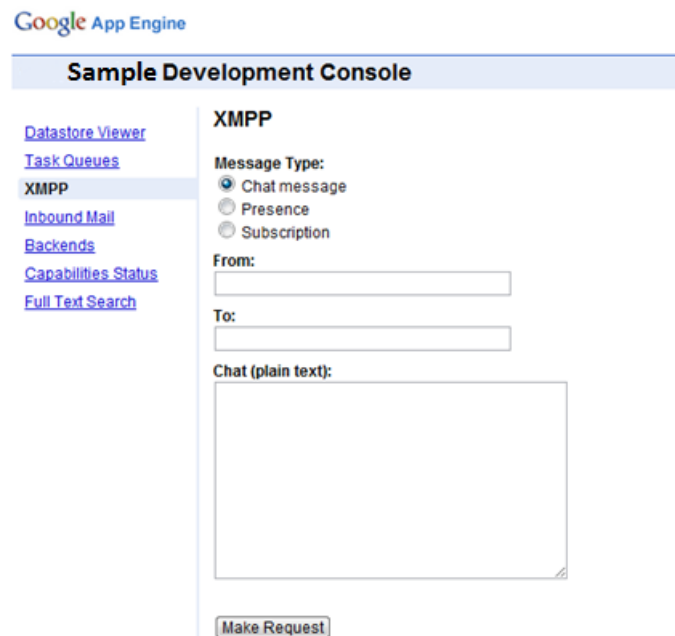
Task Queues is a useful tool that allows purging all tasks in case one of them does not succeed. For convenience there is always one default queue provided to all applications.

The Push Queue in Development Console has the following fields:

- Queue Name
- Maximum Rate(5 by default)
- Bucket Size
- Oldest task(UTC)
- Tasks in queue

4.4.4 XMPP

XMPP option provides a very simple interface that allows testing locally an application against functionalities related to XMPP service. That interface is shown below.



The screenshot shows the Google App Engine Sample Development Console. On the left, a sidebar contains links: [Datastore Viewer](#), [Task Queues](#), [XMPP](#) (highlighted), [Inbound Mail](#), [Backends](#), [Capabilities Status](#), and [Full Text Search](#). The main content area is titled 'XMPP' and includes the following fields:

- Message Type:** Three radio buttons: ☒ Chat message, ☐ Presence, and ☐ Subscription.
- From:** A text input field.
- To:** A text input field.
- Chat (plain text):** A large text area for entering the message content.
- Make Request:** A button at the bottom of the form.

Figure 4.9 XMPP console view

It can be observed that there are three message types. The interface looks a bit different for each of them.

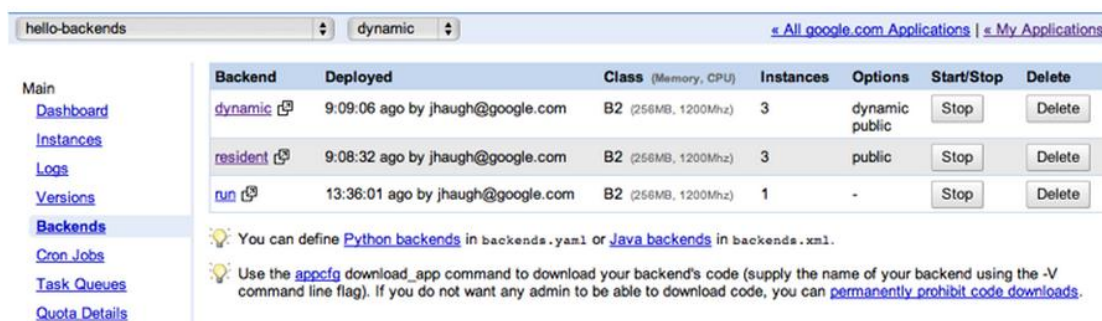
4.4.5 Inbound Mail

Inbound Mail option functionality is almost the same as XMPP's functionality. The interface differs from XMPP as its fields ask for the traditional email information.

4.4.6 Backends

If configured, Backends option when chosen, lists instances that are exempt from request deadlines and have access to more memory and CPU than normal instances.

By default the option does not list any data. An example of the backends listing can be found below.



hello-backends dynamic All google.com Applications | My Applications

Backend	Deployed	Class (Memory, CPU)	Instances	Options	Start/Stop	Delete
dynamic	9:09:06 ago by jhaugh@google.com	B2 (256MB, 1200Mhz)	3	dynamic public	Stop	Delete
resident	9:08:32 ago by jhaugh@google.com	B2 (256MB, 1200Mhz)	3	public	Stop	Delete
run	13:36:01 ago by jhaugh@google.com	B2 (256MB, 1200Mhz)	1	-	Stop	Delete

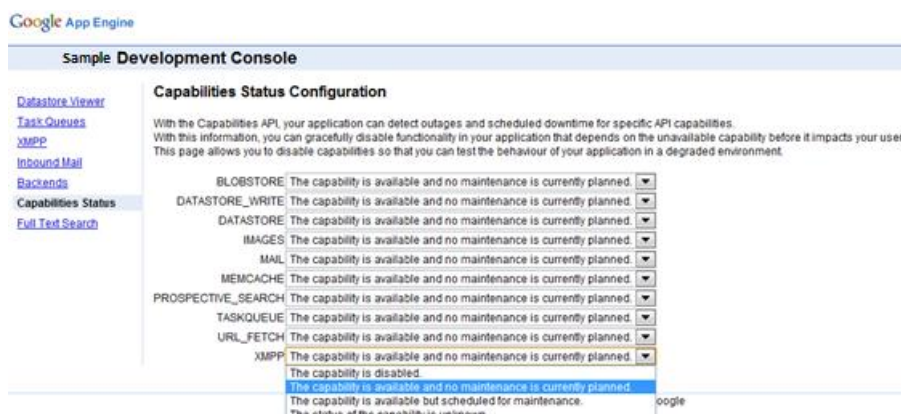
You can define [Python backends](#) in `backends.yaml` or [Java backends](#) in `backends.xml`.

Use the `appcfg download_app` command to download your backend's code (supply the name of your backend using the `-V` command line flag). If you do not want any admin to be able to download code, you can [permanently prohibit code downloads](#).

Figure 4.10 Example Backends listing console view

4.4.7 Capabilities Status

This option is perfect for checking the application against different services. Moreover, the capability status of each service is not limited to on/off only but it also has unknown or scheduled for maintenance statuses to choose from.



Google App Engine Sample Development Console

Capabilities Status Configuration

With the Capabilities API, your application can detect outages and scheduled downtime for specific API capabilities. With this information, you can gracefully disable functionality in your application that depends on the unavailable capability before it impacts your users. This page allows you to disable capabilities so that you can test the behaviour of your application in a degraded environment.

BLOSTORE	The capability is available and no maintenance is currently planned.
DATASTORE_WRITE	The capability is available and no maintenance is currently planned.
DATASTORE	The capability is available and no maintenance is currently planned.
IMAGES	The capability is available and no maintenance is currently planned.
MAIL	The capability is available and no maintenance is currently planned.
MEMCACHE	The capability is available and no maintenance is currently planned.
PROSPECTIVE_SEARCH	The capability is available and no maintenance is currently planned.
TASKQUEUE	The capability is available and no maintenance is currently planned.
URL_FETCH	The capability is available and no maintenance is currently planned.
XMPP	The capability is available and no maintenance is currently planned.
	The capability is disabled.
	The capability is available and no maintenance is currently planned.
	The capability is available but scheduled for maintenance.
	The status of the capability is unknown.

Figure 4.11 Capabilities status; console view

4.4.8 Google Plugin for Eclipse

Instead of setting up the Development Server manually, Google provides a very useful plugin for Eclipse that does that. When installed, this feature adds a new possibility of creating a Web Application Project. When the option of creating the Web Application Project is chosen, the user has a possibility of defining what SDK one wants to work on or whether to use Google Web Toolkit (GWT) or not. After following a couple of steps the entire development environment is set.

All the features previously described for the Development Server also apply for Google Plugin for Eclipse as it implements the environment with Development Server included. To test the

application on Development Server it is simply enough to choose run or debug Web Application from the project's options.

4.4.9 Local Unit Testing

App Engine provides testing utilities that use local implementations of Datastore and other App Engine services. Local Unit Testing is more of an approach than a tool. It is the documentation that gives the hints on how to write unit tests against various App Engine Services. The examples are written in JUnit.

4.4.10 Appstats

In order to maintain fast performance of an application it is very important to know whether the application is making superfluous RPC calls, should it cache data instead of making the same RPC calls to get the same data or should the requests be executed in parallel or serially. Those questions are much easier to answer with the aid of the Appstats library as it allows to trace all RPC calls for a given request and displays the time and cost of each call. Below the RPC timeline graph is shown.



Figure 4.12 Example console view for RPC timeline.

4.5 Limits, Quotas & Billing

Each API is free for a certain level of usage. In order to control that level Google established so called quotas i.e. size or amount of requests for each API. Quotas do not serve only the purpose of limiting application resources but they also allow controlling the usage of those resources. The table shown below represents the free quotas at the time of writing the thesis, as these quotas are subject to change. Below quotas are refreshed every 24 hours.

Service	Resource	Limit
Requests	Outgoing Bandwith	1.00 GBytes
	Incoming Bandwidth	1.00 GBytes
	Frontend Instance Hours	28.00 Instance Hours
	Backend Instance Hours	9.00 Instance Hours
Storage	Datastore Write Operations	0.05 Million Ops
	Datastore Read Operations	0.05 Million Ops
	Datastore Small Operations	0.05 Million Ops
	Datastore Stored Data	1.00 GBytes
	Blobstore Stored Data	5.00 GBytes
	Number of Indexes	200
	Prospective Search Subscriptions	10,000
	Logs Stored Data	1.00 GBytes
Search	Search API Calls	20,000
	Search Stored Data	0.25 GBytes
Mail	Mail API Calls	7,000
	Recipients Emailed	100
	Admins Emailed	5,000
	Message Body Data Sent	0.06 GBytes
	Attachments Sent	0.00 GBytes
	Attachment Data Sent	0.10 GBytes
URL Fetch	UrlFetch API Calls	657,084
	UrlFetch Data Sent	4.00 GBytes
	UrlFetch Data Received	4.00 GBytes
XMPP	XMPP API Calls	46,310,400
	XMPP Data Sent	1,046.00 GBytes
	Recipients Messaged	46,310,400
	Invitations Sent	100,000
	Stanzas Sent	10,000
Channel	Channel API Calls	46,310,400
	Channels Created	100
	Channel API Hours Used	200.00 Channel Hours
	Channel Data Sent	1,046.00 GBytes

Task Queue	Task Queue API Calls	100,000
	Task Queue Stored Task Count	1,000,000
	Task Queue Stored Task Bytes	0.49 GBytes
Deployments		1,000

Table 4.3 Quotas and Limits for chosen services

Whenever an application administrator decides to increase the billable quotas one can do that by enabling paid apps and setting a daily budget. Then the administrator is charged for the resources actually used, and for the amount of resources used above the free quota thresholds. The minimum cost of paid application maintenance is \$2.10/week [7].

Resource depletion

Whenever all of the allocated resources are consumed by an application, the resource becomes unavailable until the quota is replenished. Thus the application will most likely stop working if this happens. In such case, while attempting to consume the resource results in an exception that can be caught by the application and handled, such as by displaying a friendly error message to the user [7].

4.6 Summary

Without the wide range of integrates services Google App Engine would be much more complex in use and much less efficient. Additionally, Google releases new API's quite often. The first release of each one of them is the experimental version release for a couple of months or even longer. Thus when each feature is tested by thousands of users, Google commits all the required fixes and releases its stable version.

Moreover Google App Engine provides a very useful tool i.e. a table distinguishing platform and most crucial API's statuses.

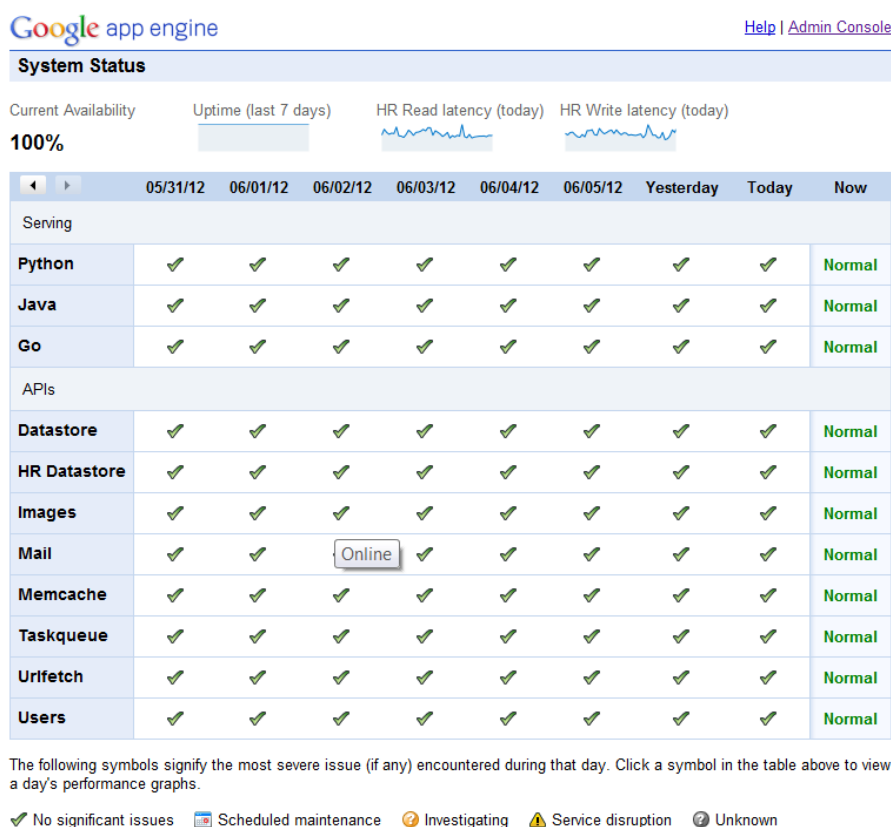


Figure 4.13 System status table

It should be observed that developers have a reason to worry that the applications will not be portable from App Engine which means a vendors' lock-in mentioned in previous chapters. Though in response, there are being developed a number of projects to create open-source back-ends for the various proprietary/closed APIs of app engine, especially the Datastore. Although these projects are at various levels of maturity, none of them is at the point where installing and running an App Engine app is as simple as it is on Google's service. So far the most interesting open source efforts are AppScale and TyphoonAE.

5 Pilot Application

This chapter describes the sample simple application developed for previously-described Google App Engine PaaS platform. The objective of this chapter is to demonstrate some of already explained Google's cloud features in working environment.

The development of the pilot has been performed following Everis COM Methodology for generic development. This methodology provides a set of phases and procedures to succeed in the creation of a generic development with desired quality.

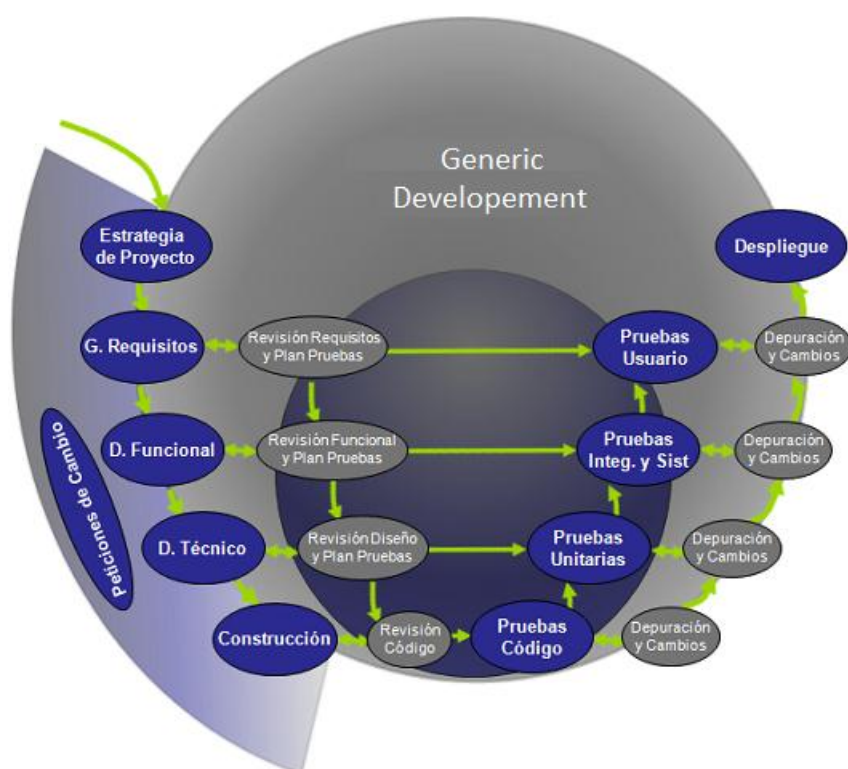


Figure 5.1 Everis COM methodology

5.1 Project Strategy

5.1.1 Objective and Scope of the Pilot Application

Main objectives of the pilot application are the following:

- To analyze functionalities of Google App Engine and assess whether the Google's platform is capable and ready for real project implementation.
- To examine some of the App Engine's services studied in the previous chapters.
- To find out which elements of Java are not compatible with Google's PaaS
- To assess the learning curve for the average developer willing to start developing on App Engine.

The scope of the pilot application is defined explicitly by the requirements defined in the following sections.

5.1.2 Technology and Motivation

After conducting an in-depth analysis of various PaaS solutions, Google App Engine seemed as a well-equipped platform for Java developers. Moreover, no fee is being charged for limited use and as such it freely allowed for testing different characteristics of Google's platform.

In order to examine some of the App Engine's functionality it was decided to develop a simple application in Java. The chosen technology was Java Server Faces framework with the PrimeFaces plugin.

Since Google App Engine disposes of many different services' APIs only few of them were chosen to test against, and they were pointed out in a table 4.2 in the previous chapter.

5.2 Requirement Management

5.2.1 Obtaining the Requirements

Defining the requirements for the demo application was carried out during the first three months of involvement in this project. As more research was carried out and more in-depth knowledge gained in regards to Google App Engine, the objectives have become clearer.

At the very beginning it was specified that the application should possess two main characteristics, i.e. it should have a look of a banking application and should allow to test against some of the App Engine's functionalities, and to explore specific features of Google's platform.

The functional requirements will be described through the use cases that specify the flow of the user-application interaction.

5.2.2 Non-Functional Requirements

The non-functional requirements will be described using the simplified Volere method template.

Name	Name of the non-functional requirement		
Code	Identification number of the non-functional requirement		
Description	A brief description of the non-functional requirement		
Justification	A short description of why such requirement should be accomplished.		
User satisfaction	<p>Indicates the satisfaction level that the user will achieve if the requirement is implemented correctly.</p> <p>Scale ranges from 1(indifferent) to 5(extremely satisfied)</p>	Dissatisfaction of a user	<p>Indicates the dissatisfaction level that the user will get if the requirement is not delivered.</p> <p>The scale ranges from 1 (indifferent) to 5 extremely (dissatisfied)</p>
Condition of satisfaction	A minimal condition that has to be fulfilled by a requirement to be approved.		
Priority	Relative importance of a requirement (low, medium, high)	Date	Dated of creation or modification of the requirement

Table 5.2 Non-functional requirement description template

Name	Banking application appearance		
Code	NFR01		
Description	The application at first glance should look like a banking application		
Justification	One of the application's objectives is to provide some simplified banking functionality and as such it should also have a look of online banking system.		
User Satisfaction	3	User Dissatisfaction	5
Condition of satisfaction	There will be conducted tests in various browsers i.e. IE, Mozilla Firefox, and Google Chrome. The application has to be displayed correctly visually and functionally.		
Priority	High	Date	Created 01/05/2012

Table 5.2 Banking application appearance

Name	Ajax implementation		
Code	NFR02		
Description	The application should have Ajax functionality for better interaction with its users		
Justification	The application whenever possible should implement Ajax functionalities which will allow for not refreshing the entire page when not necessary		
User Satisfaction	3	User Dissatisfaction	3
Condition of satisfaction	The application does not refresh the entire website when not necessary		
Priority	High	Date	Created 01/05/2012

Table 5.3 Ajax implementation

Name	Filling out the fields aided with controllers		
Code	NFR03		
Description	Whenever possible, application should aid a user when filling out the fields with controls such as popup calendar, buttons, or any other control that could be of use.		
Justification	In order to make the task easier and avoid additional issues when filling out the fields it should be avoided to let user introduce data manually whenever possible.		
User Satisfaction	3	User Dissatisfaction	3
Condition of satisfaction	All possible fields aided with control buttons		
Priority	Medium	Date	Created 01/08/2012

Table 5.4 Filling out the fields aided with controllers

5.3 Functional Design

5.3.1 Use Case Diagram

In this subchapter use cases of the pilot application will be depicted on use case diagram and subsequently each one of them will be described.

The pilot application has only two actors, i.e. User and Google Administrator, where the Google Administrator is the representation of a Google's account management.

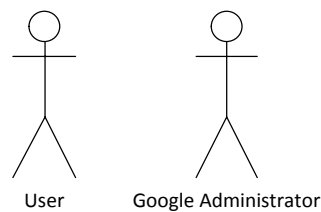


Figure 5.2 Actors involved in the use case of the pilot application

The application itself does not consist of any more-privileged actor than user, and as such there is no application administrator actor. The way that above-mentioned actors interact with the application is depicted below. User also represents a browser, and as such whenever any kind of interaction occurs between him and the system it should be understood that user is equivalent to a browser.

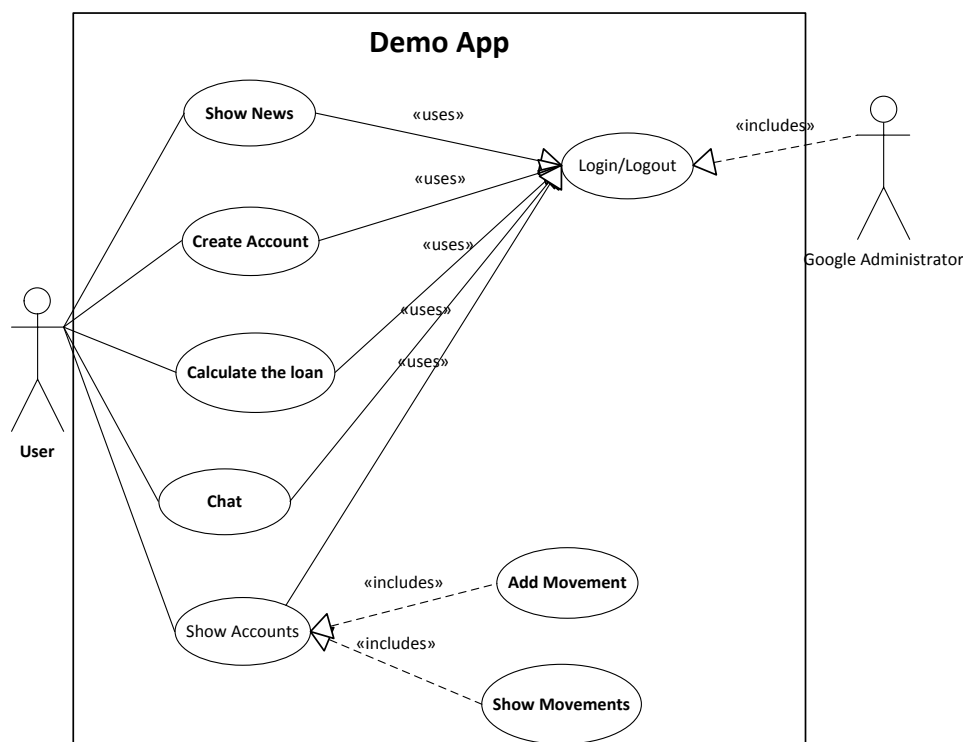


Figure 5.3 Use Case Diagram for the sample app

5.3.2 Functional Requirements

In order to describe the use cases it was decided to define a template based on Everis' COM methodology template and the Volere¹⁵ method template. Such defined template will be as follows.

Name	Name of the use case		
Code	ID of the use case		
Elaboration	Indicates whether the description is initial, base, or elaborated. Elaborated description is more detailed than the other types		
Description	Brief description of the use case		
Actor	Actor that activates or participates in the use case		
Trigger	Action or event by which the actor activates the use case		
User Satisfaction	Indicates the satisfaction level that the user will achieve if the requirement is implemented correctly. Scale ranges from 1(indifferent) to 5(extremely satisfied)	Dissatisfaction of a user	Indicates the dissatisfaction level that the user will get if the requirement is not delivered. The scale ranges from 1 (indifferent) to 5 extremely dissatisfied)
Pre-conditions	Conditions that have to be fulfilled in order to proceed with the use case		
Condition of satisfaction	Is the minimum condition or test case that the functional requirement has to fulfill to determine whether it is acceptable or not.		
Flow of Events	Describes the sequence of basic actions that are executed between an actor and the system		
Alternative Flows	Description of the sequence of alternative actions to the basic flow		

Table 3.5 Use Case description template

¹⁵ Volere Requirements Specification Template - <http://www.volere.co.uk/template.html>

Name	Login		
Code	UC1		
Elaboration	Base		
Description	User enters his credentials, and logs in the system.		
Actor	User, Google Administration		
Trigger	User wants to access the application.		
User Satisfaction	5	Dissatisfaction of a user	5
Pre-conditions	User must possess a valid Google account		
Condition of satisfaction	User can access the application		
Flow of Events	1.- Application is shown to the user		
Alternative Flows	*.a.- User creates a new Google account		

Table 5.6 Login

Name	Show Blog News		
Code	UC2		
Elaboration	Base		
Description	The application must let the user read the Everis blog within the application.		
Actor	User		
Trigger	The user clicks on the Home button of the application.		
User Satisfaction	3	Dissatisfaction of a user	3
Pre-conditions	User has to be logged in		
Condition of satisfaction	User can navigate and read the Everis blog within the application.		
Flow of Events	1. - Application displays the blog within its window.		
Alternative Flows	*.a.- Application shows the error (where its number depends on connection error with Everis blog site) *.b.- User logs out		

Table 5.7 Show Blog News

Name	Create Account		
Code	UC3		
Elaboration	Base		
Description	The application must let the user to create a new movement's account.		
Actor	User		
Trigger	User wants to create a new account and clicks on Account button		
User Satisfaction	4	Dissatisfaction of a user	4
Pre-conditions	User has to be logged in		
Condition of satisfaction	The new account has been created		
Flow of Events	1.- Application successfully creates a new account		
Alternative Flows	*.a. - Application shows the error that one of two fields are missing the input value. *.b.- User logs out		

Table 5.8 Create Account

Name	Calculate Loan		
Code	UC4		
Elaboration	Base		
Description	The application must let the user to calculate a loan given a loan amount, interest rate and number of months. It also gives possibility of sending the results at the email address.		
Actor	User		
Trigger	User wants to calculate loan rates		
User Satisfaction	3	Dissatisfaction of a user	4
Pre-conditions	User has to be logged in		
Condition of satisfaction	Application calculates the rates and sends them to the user's email address if decided to.		
Flow of Events	1.-Application displays the counted outcome such total interest, total payment and monthly payment. 2. - Upon request the user receives an email with calculated rates.		
Alternative Flows	*.a. - Application shows the error that one of two fields is missing the input value. *.b. - User logs out		

Table 5.9 Calculate Loan

Name	Chat		
Code	UC5		
Elaboration	Base		
Description	The application must let the user to chat with other currently logged in users.		
Actor	User		
Trigger	The user wants to login to the chat.		
User Satisfaction	4	Dissatisfaction of a user	3
Pre-conditions	User has to be logged in		
Condition of satisfaction	Application displays the list of available contacts and allows for chatting with them.		
Flow of Events	1. - User logs in to the chat with his Google account nickname. 2. - User can see other available users and can exchange text messages with them.		
Alternative Flows	*.a. - User logs out		

Table 5.10 Chat

Name	Show Accounts		
Code	UC6		
Elaboration	Base		
Description	The application must let the user view the accounts that are in his possession		
Actor	User		
Trigger	User wants to see his current accounts		
User Satisfaction	4	Dissatisfaction of a user	4
Pre-conditions	User has to be logged in		
Condition of satisfaction	User can view his accounts		
Flow of Events	1. - Application displays current accounts. For each account the specified fields are account number, account type, current amount and options		
Alternative Flows	*.a. - Application lets user know that there were no records found if that was the case. *.b - User logs out		

Table 5.11 Show Accounts

Name	Add Movement		
Code	UC7		
Elaboration	Base		
Description	The application must let the user add new movements contained within the account.		
Actor	User		
Trigger	User wants to add a new movement		
User Satisfaction	5	Dissatisfaction of a user	5
Pre-conditions	User must have at least one account		
Condition of satisfaction	User can add new movement		
Flow of Events	<ol style="list-style-type: none"> 1. Includes functional requirement “Show Accounts” 2. User chooses an option “Add movement” within an account of interest 3. Application adds a new movement after specifying movement details such as date, information, amount 4. Application updates current amount for the account 		
Alternative Scenario	<p>*.a. - Application prompts an error as one or all of the required fields were not filled out.</p> <p>*.b.- User clicks the Accounts button on the top menu to get back to Account view</p> <p>*.c.- User logs out</p>		

Table 5.12 Add Movement

Name	View Movements		
Code	UC8		
Elaboration	Base		
Description	The application must let the user view his previous movements		
Actor	User		
Trigger	User wants to see his previous movements		
User Satisfaction	4	Dissatisfaction of a user	3
Pre-conditions	User must have at least one account		
Condition of satisfaction	User can view his previous movements.		
Flow of Events	<ol style="list-style-type: none"> 1. Includes functional requirement “Show Accounts” 2. User chooses an option “See movements” within an account of interest 3. Application displays previous movements, with following fields specified: Movement, Date, More Info, Money, and Balance 		
Alternative Flows	<p>*.a.- User clicks the Accounts button on the top menu to get back to Account view</p> <p>*.b. - User logs out</p>		

Table 5.13 View Movement

5.3.3 Conceptual Data Model

The conceptual model represents a high level data structure of a system. The following is the conceptual model of the application.

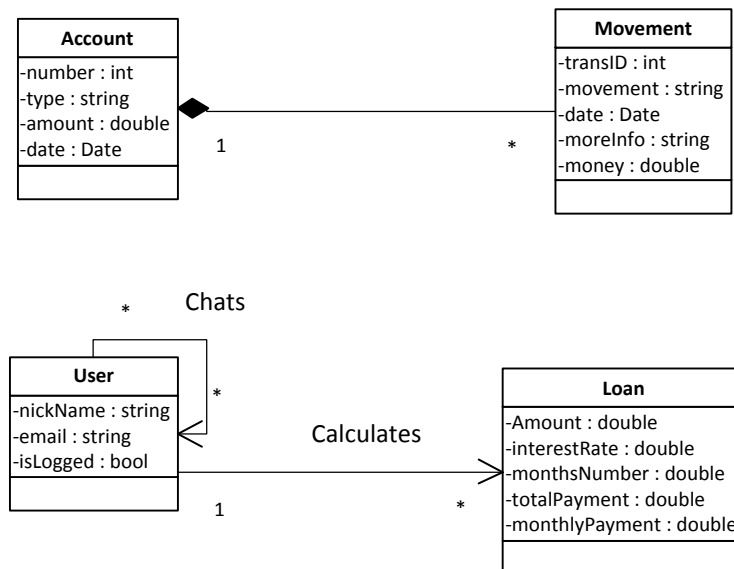


Figure 5.4 Conceptual schema of the application

The following subchapter will describe in detail each of the entities presented in the conceptual diagram shown above.

5.3.3.1 Conceptual Model Class Description

Each of the classes presented in the conceptual model will be briefly described below.

Name of the class	Account
Description	This class represents accounts created by the user.
Atributes	<ul style="list-style-type: none"> • Number – it is an automatically generated unique account number. • Type – it is an account type chosen by the user • Amount – amount of money/balance of the account • Date – Date and time when the account was created

Name of the class	Movement
Description	This class represents movements added by users
Atributes	<ul style="list-style-type: none"> • transID – unique identification number of the movement • movement – brief information about the movement • date – indicates the date when the transaction was input • moreInfo – more detailed information about the movement • money – amount of the transaction

Name of the class	User
Description	
This class represents a user	
Atributes	
<ul style="list-style-type: none"> nickname – nickname is the first part of the user's email. Email – user's email address of the user's account isLogged – indicates whether the user is logged in the system 	

Name of the class	Loan
Description	
This class represents loan calculated by the user	
Atributes	
<ul style="list-style-type: none"> Amount – amount of required loan interestRate – interest rate through which the loan is to be calculated monthsNr – number of months to pay off the loan totalPayment – entire amount (with interest) to be repaid monthlyPayment – monthly rate calculated 	

5.3.4 Navigation Map

In this part the navigation map of the pilot application is shown. The UX model is used to show how the content of the application will be structured and organized within different screens, and to show how the user will navigate through the screens when executing the use cases of the application.

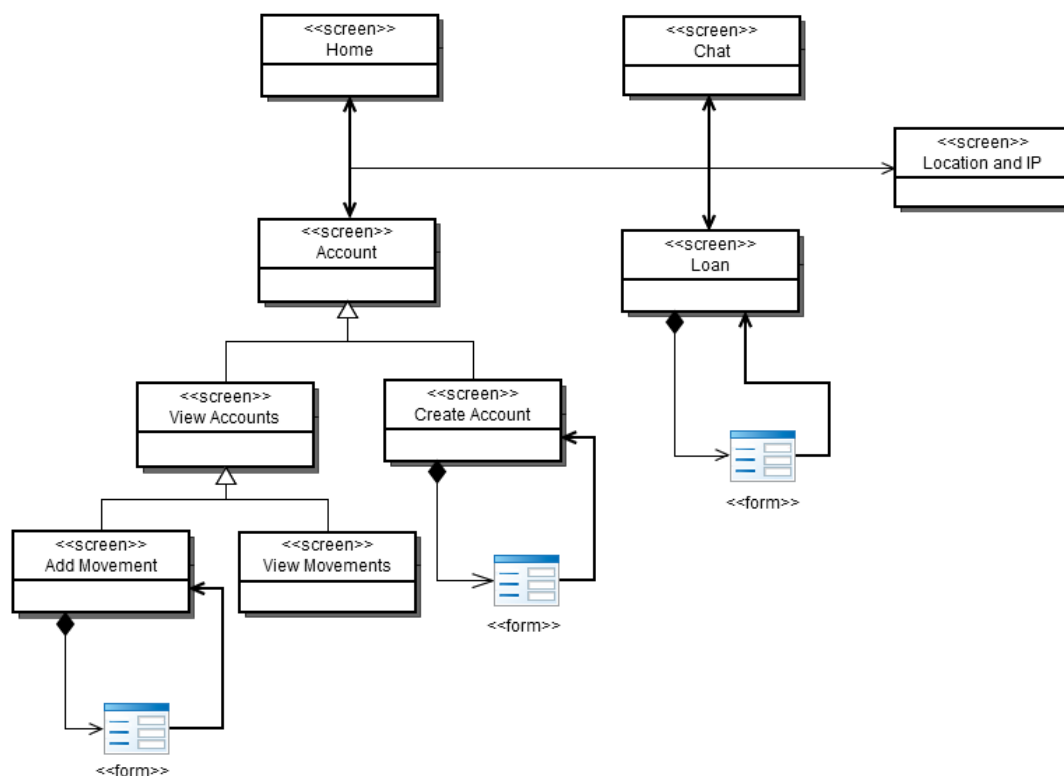


Figure 5.54 Navigation Map of the pilot application

5.3.5 User Interface Design

When designing the application's interface the only requirement was that it had to visually resemble the online banking application. As such it was decided to develop the application's structure in the following way.

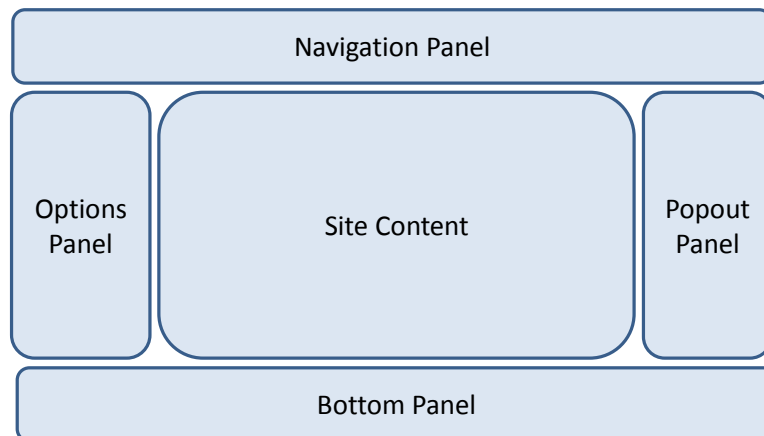


Figure 5.6 Structure of the application's interface

Beneath is presented a brief description for all the parts of the structure.

- **Navigation panel** – This panel contains of navigation buttons, which allow user to choose whatever one wishes to do at any time. Moreover, the logout link is also contained within that panel and it can be as well clicked on at any time.
- **Options panel** – Within this panel, user has three options, i.e. see the calendar with a current time and date, see their IP address and location, and login to chat.
- **Site Content** – This is the only part of the application that will change its contents. Any target outcome will be shown to the user on this part.
- **Bottom Panel** – This panel is simply for additional visual effects. A simple jQuery scroller was implemented for adding extra functionality.

5.3.5.1 User Interface of the Default Site of the Application

The default interface of the application and how it relates to the already described general interface schema can be observed below.



Figure 5.7 Default interface of the pilot application

The above snapshot shows the application after clicking on the Home button. As already mentioned the only dynamic part of the website is the content container in the middle. Here, the right pop-out panel was hidden for a better display of the blog.

It should be noticed that for all the cases when unfolding the tabs from “Options” menu they change color from blue to green.

5.3.5.2 User Interface of the Account Site

Below is presented the view after choosing an account view.

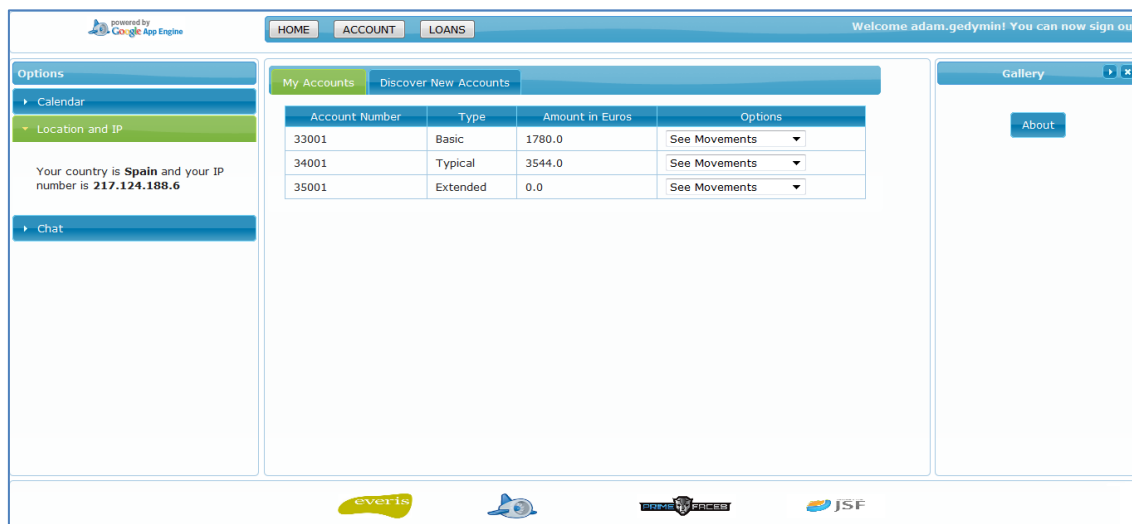
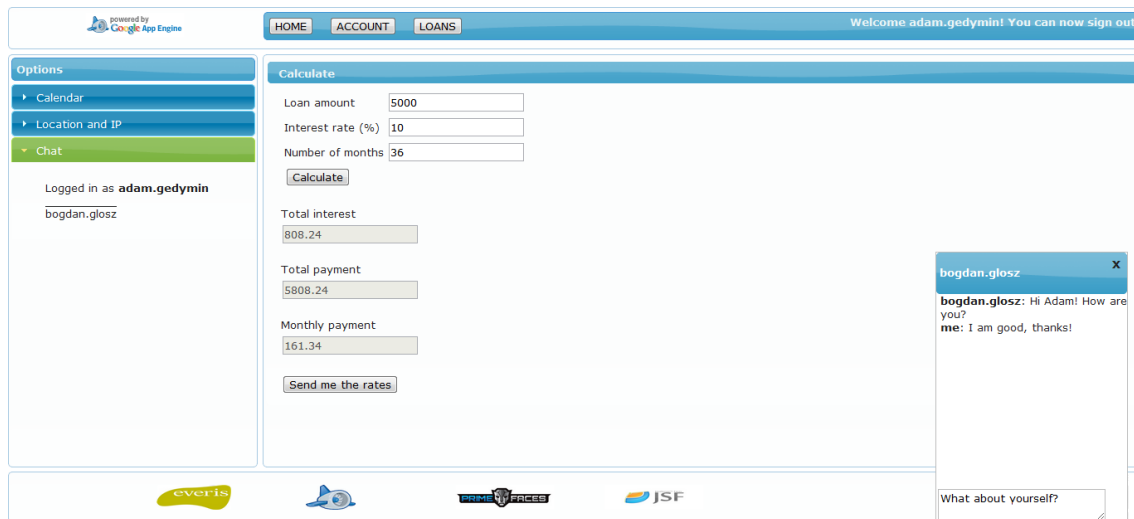


Figure 5.8 Interface of the pilot application after choosing Account view on the navigation panel and Location and IP from options

On the left Options panel it can be observed that “Location and IP” tab was unfolded so the user can consult his IP address and his location. This functionality like most of the others in this application has the only objective to test App Engine’s functionality (in this case URLfetch) and as such some superfluous behaviors (e.g. showing the country) are not avoided.

5.3.5.3 Chat and loan User Interface

Below is shown the user interface after entering the “Loan” tab and “Chat” from the options menu.



The screenshot displays a web application interface. At the top, there is a navigation bar with 'HOME', 'ACCOUNT', and 'LOANS' tabs. A welcome message 'Welcome adam.gedymint! You can now sign out' is visible. On the left, an 'Options' menu is open, showing 'Calendar', 'Location and IP', and 'Chat' (which is highlighted). Below the menu, it says 'Logged in as adam.gedymint' and 'bogdan.glosz'. The main content area is titled 'Calculate' and contains input fields for 'Loan amount' (5000), 'Interest rate (%)' (10), and 'Number of months' (36). A 'Calculate' button is present. Below these inputs, the results are displayed: 'Total interest' (808.24), 'Total payment' (5808.24), and 'Monthly payment' (161.34). A 'Send me the rates' button is at the bottom of this section. On the right side, a chat window is open, showing a conversation between 'bogdan.glosz' and 'me'. The chat messages are: 'bogdan.glosz: Hi Adam! How are you?' and 'me: I am good, thanks!'. At the bottom right, there is a text input field with the placeholder 'What about yourself?'.

Figure 5.9 User interface after loan calculation while chatting

On the figure above in the left “Options” panel when unfolding the “Chat” tab, the user can see a list of people currently logged in to the same application, and chat with them. When a user clicks on the other user’s name, a chat window will pop out as can be easily observed in the right corner of the website. In the content area loan calculation feature appeared after clicking the “Loan” button.

5.3.6 Logical Architecture

For developing the pilot application the Model-View-Controller (MVC) pattern will be used. Since it was decided to develop the application in Java ServerFaces framework, the MVC pattern will have a bit different impact on the application than other MVC approaches.

- *Controller* - The controller in the JSF architecture is consisting of the front controller Servlet called FacesServlet, the centralized configuration file and a set of action handlers for the web application. The front controller Servlet is responsible for receiving every request and performs the steps according to the request processing lifecycle to create a response for the client. The event listeners respond to events generated from the JSF event model and manipulate the model or invoke other backend code for example.
- *Model* - The application model in a JSF environment is implemented as a set of server-side JavaBeans that hold values of the model, and define methods on these values. These JavaBeans may also be persisted through an underlying persistence layer and a database through the use of Java Data Objects (JDO), Enterprise JavaBeans or an object-relational mapping implementation like Hibernate.
- *View* - The main part of the view is the component tree that contains the stateful user interface components. Components can be rendered in different ways according to the type of the client. The view delegates this work to separate renderers, each taking care of one specific output type, HTML or WML for example.

Below is depicted a schema of a JSF MVC pattern.

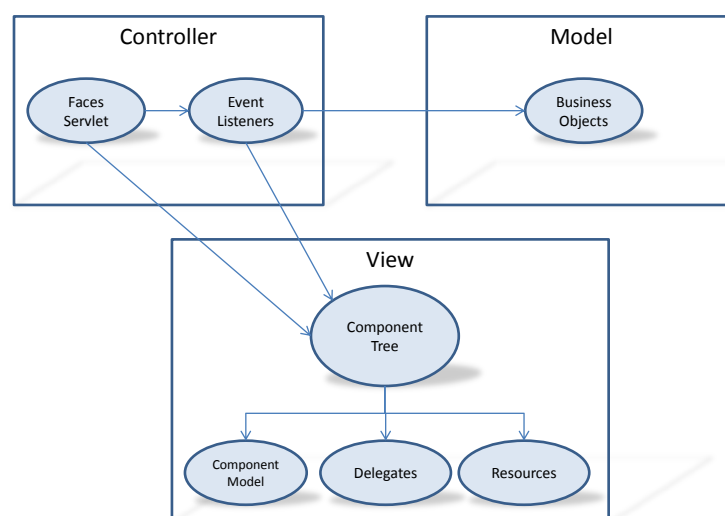


Figure 5.10 The MVC-Architecture of JavaServer Faces

The above depicted MVC pattern functions in the pilot application, the following way.

1. *Restore View* - In the first stage a request comes through the FacesServlet controller. The Restore View phase retrieves the component tree for the requested page if it was displayed previously. If the view does not already exist, the JSF controller constructs a new component tree and displays it for the first time.
2. *Apply Request Values* - the JSF implementation iterates over the component objects in the component tree. Each component object checks whether the request values belong to it and stores them. The components must be first retrieved or created from the FacesContext object, followed by their values. Component values are typically retrieved from the request parameters, although they can also be retrieved from cookies or headers.
3. *Process Validation* - At this stage, each component will have its values validated against the application's validation rules. The submitted string values are first converted to "local values," which can be objects of any type. Whenever a conversion or validation errors happen, an error message is added to FacesContext, and the component is marked as invalid. If a component is marked as invalid, the JSF implementation invokes the Render Response phase directly, displaying once again the current page so that the user has another chance to provide correct inputs. If there are no validation errors, JSF advances to the update model values phase.
4. *Update Model Values* - updates the actual values of the server-side model namely, by updating the properties of the backing beans. Only bean properties that are bound to a component's value are updated.
5. *Invoke Application* - the JSF controller invokes the application to handle form submissions. The action method of the button or link component that caused the form submission is executed.
6. *Render Response* -In this step response is encoded and sent to the browser. In this phase the JSF engine renders the UI components in the component tree persisted in the FacesContext.

5.3.7 Used Technologies

JavaScript – Scripting programming language created by Netscape, and it is typically used in website development. JavaScript was formalized in the ECMAScript language standard in the late 90's and is mainly used in the form of client-side JavaScript, implemented as part of a Web browser in order to give enhanced user interfaces and dynamic websites.

CSS – Cascading Style Sheets, it is a language used to describe the way that websites are displayed. CSS is mainly a list of rules that specify how the content of a website should be displayed by the browser.

JavaServer Faces 2.0 (JSF 2.0) – it is a Java Framework with the main objective of constructing the user interface much easier. The default technology used for handling the view is so called Facelet although JSF does not prohibit usage of other solutions (such JSP).

PrimeFaces v3.4 – JSF plugin that delivers many useful features for developing the user interface.

HTML - HyperText Markup Language is the main markup language used for displaying web pages and any other information that can be parsed by a web browser.

JQuery – It is a light library for JavaScript language. It allows for making visual effects with a relatively small amount of code.

AJAX (Asynchronous JavaScript and XML) - Ajax is an asynchronous technology. It means that data is requested from the server and it is loaded very rapidly without interference with web site behavior.

Objectify – It is a convenient interface to the Google App Engine Datastore

5.4 Technical Design

5.4.1 Sequence Diagrams

The following are the sequence diagrams that represent how objects interact among each other. In this chapter the most representative use cases will be depicted.

5.4.1.1 Show Blog News Use Case

The subsequent sequence diagram refers to Show Blog News use case, which is run by default when starting the application.

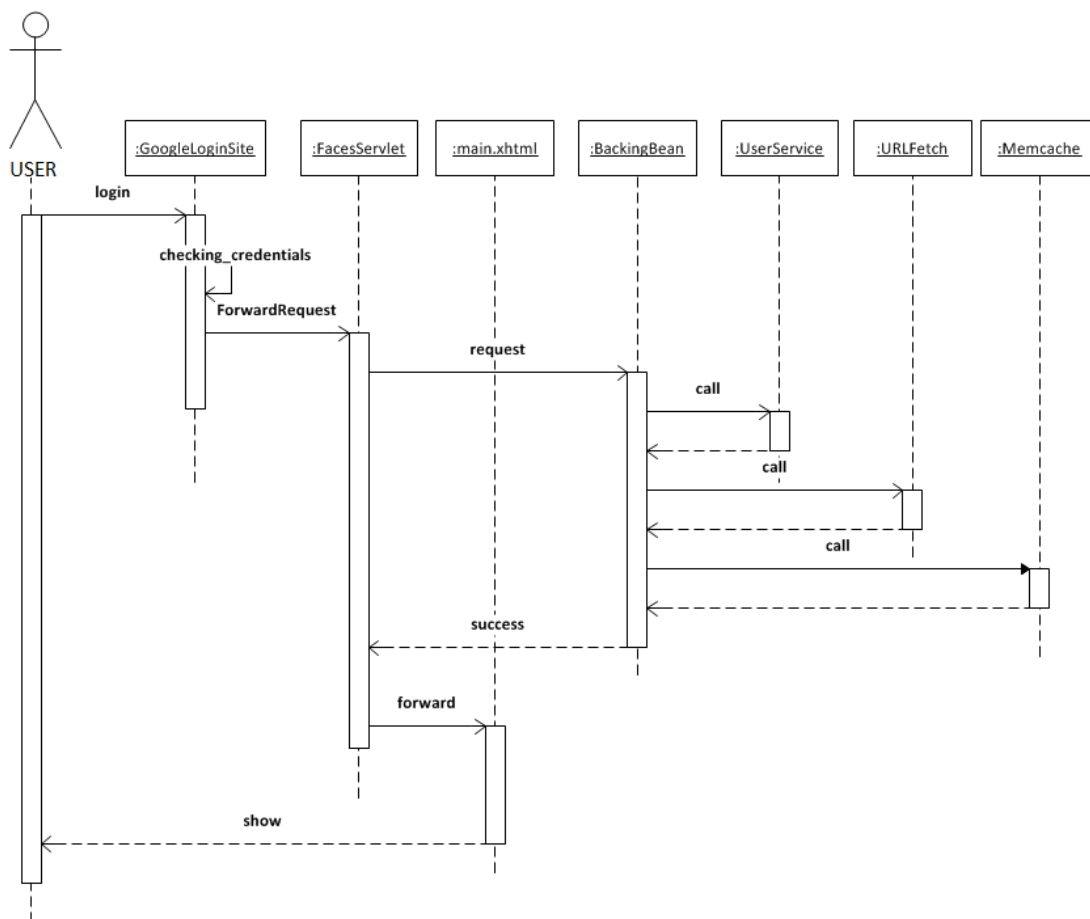


Figure 5.11 Sequence Diagram for Show Blog News use case

5.4.1.2 Chat Use Case

The below sequence diagram represents a general flow of how the chat works. Moreover, this diagram represents two actions triggered by the user, i.e. log in to chat and message with another user.

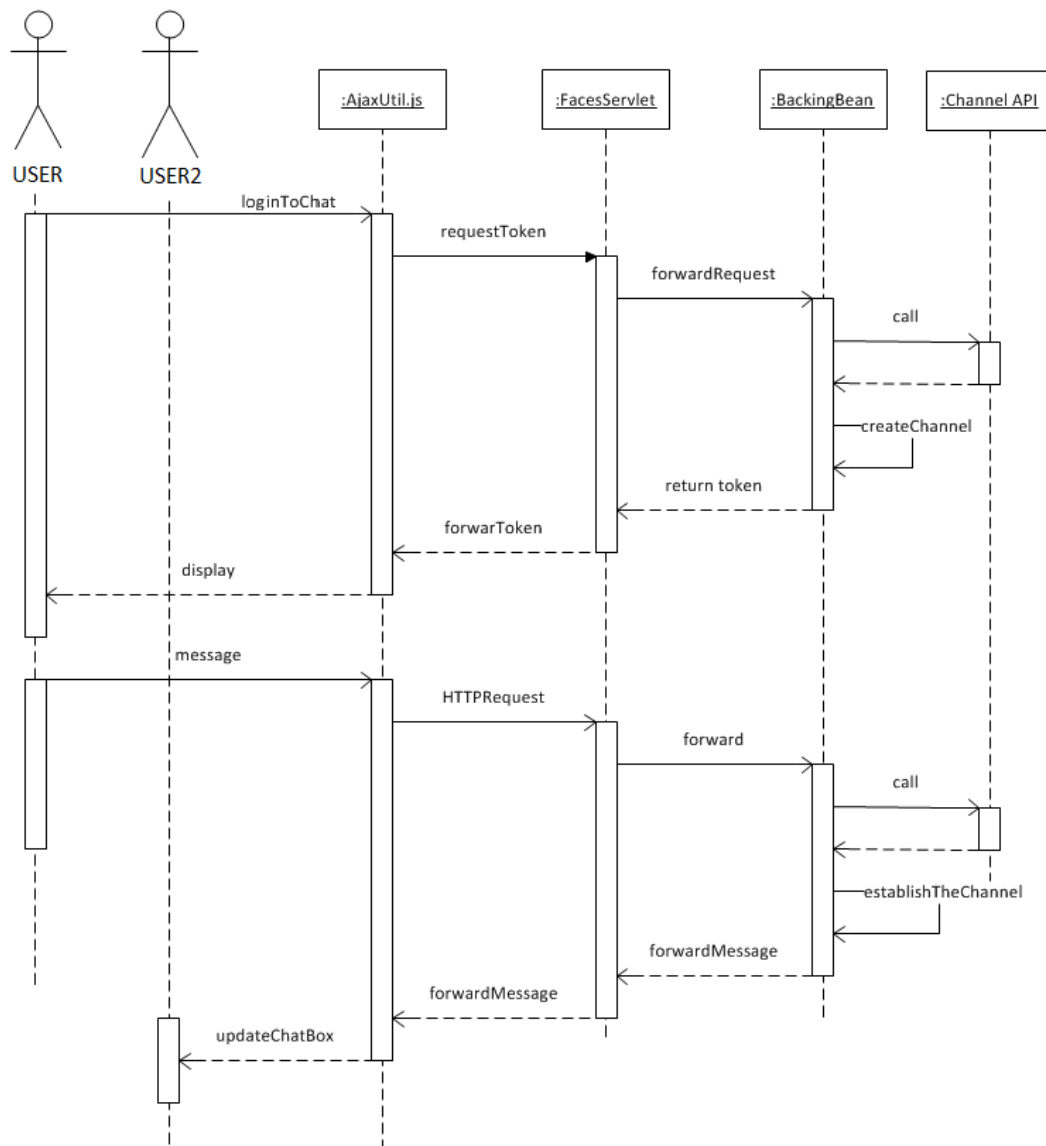


Figure 5.125 Sequence Diagram for Chat use case

5.4.1.3 Add New Account Use Case

The below sequence diagram depicts in a very simplified way, the flow of interactions between objects when a new account is created by the user.

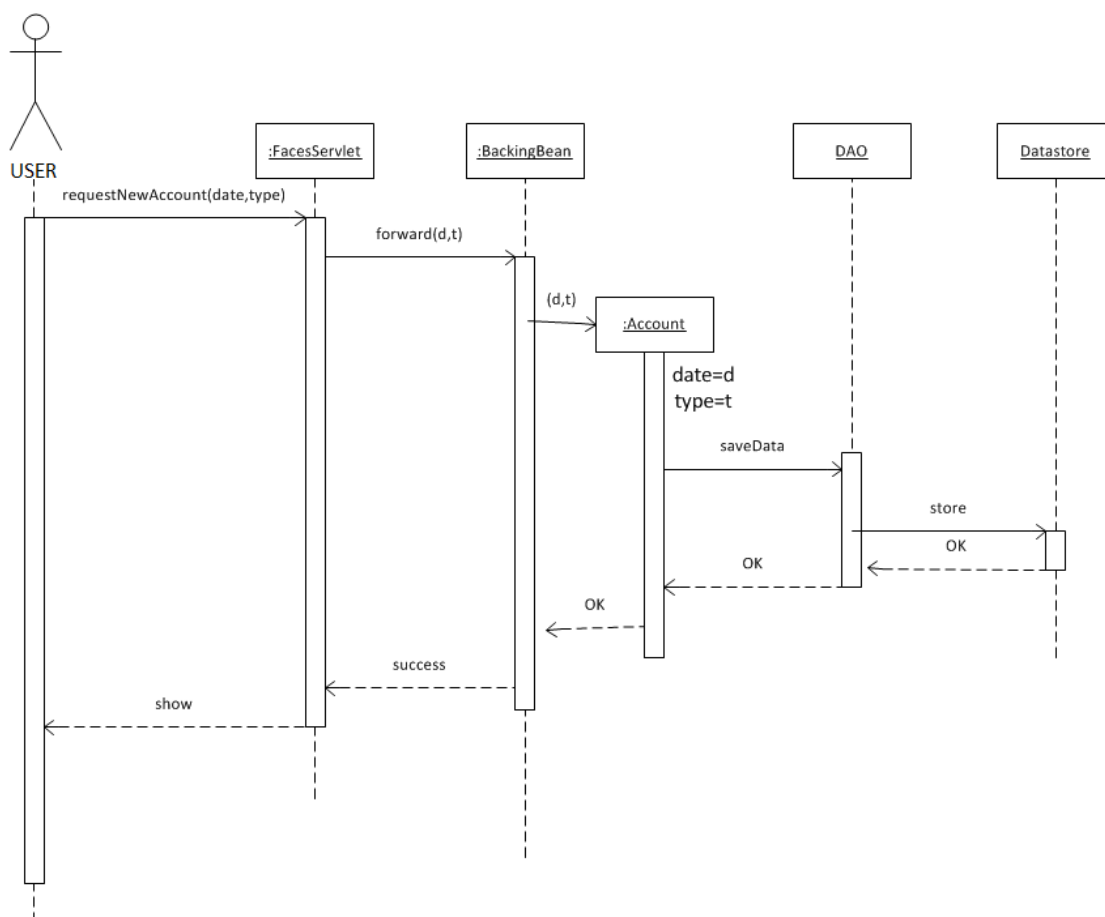


Figure 5.13 Add New Account sequence diagram

5.4.2 Pilot Application's File Structure

Below is the presented the file structure of the pilot application.

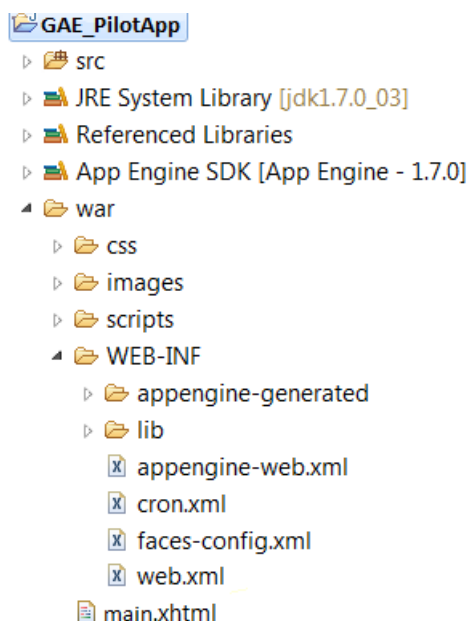


Figure 5.14 File structure of the pilot application

Each of the folders presented in the figure above will be given a brief description.

- src/
 - JRE System Library
 - Referenced Libraries
 - App Engine SDK
 - war/

This folder contains all the java classes. In terms of JSF those would be Backing Beans and Java Beans.

Here the typical and default Java libraries are being kept.

All the referenced manually added necessary jars are kept in this folder.

Libraries automatically aggregated by App Engine. All the libraries contained in this folder are necessary in order to successfully run an application within App Engine environment.

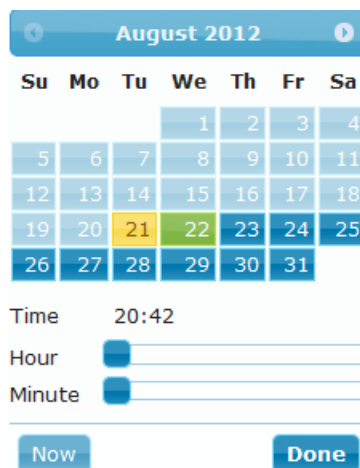
As in most Java deployments this folder (Web Application Archive) consists of others folders, and view part of the application i.e..xhtml and jsp files.

- war/CSS/ All the cascading style sheet files responsible for the web site aesthetics are kept in this folder
- war/Images/ All the icons and and images used in the application are kept here.
- war/Scripts/ JavaScript, JQuery and Ajax scripts responsible for application dynamics remain in this folder.
- war/WEB-INF This folder contains main configuration files such as web.xml (used for servlet mapping, enabling some GAE features etc.), appengine-congig.xml (specific GAE configurations), cron.xml(file defining cron jobs for GAE)

5.4.3 Specification of the User Interface

This paragraph will point out the components used for creating the user interface. All the elements that have been user were delivered either by JSF or Primefaces.

1. Popout calendar (DatePicker)



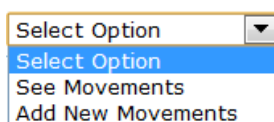
2. Menu with buttons



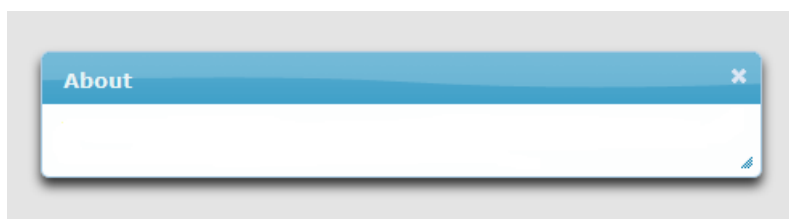
3. Unfolding tabs




4. SelectOneMenu



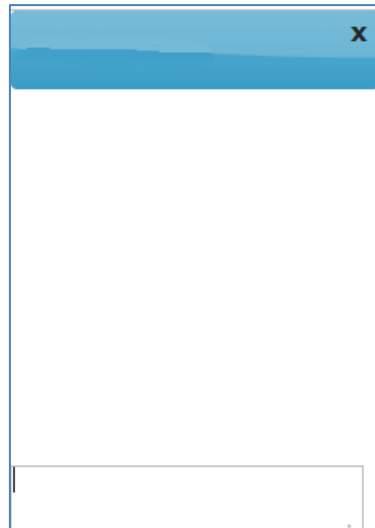
5. Alert Dialog



6. Validation Error

 Movement: Validation Error: Value is required.

7. ChatBox



8. Login Box

Sign in
Google

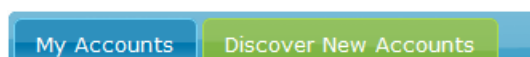
Email

Password

☒ Stay signed in

[Can't access your account?](#)

9. TabView



The above listed elements are the ones worth showing, although application consists of other more basic components.

5.5 Project Evaluation

This subchapter will evaluate the obstacles encountered during the application development.

5.5.1 Identified Obstacles

During the development of the pilot application, some issues were encountered. The ones worth mentioning are pointed out below.

- Running JavaServer Faces on Google App Engine
- Communication with Datastore
- Application Performance
- Serialization Issues
- Memcache API

The following subchapters will describe in detail encountered issues.

5.5.1.1 Running JavaServer Faces on Google App Engine

When it was decided to develop the pilot in JSF, there was no reason to believe that there could be a problem with implementing that framework on GAE, as there existed a walk-through tutorial on how to implement it. Apparently this was due to how JSF checks whether it can use `InitialContext` first by invoking it. Unfortunately, this provokes unusual behavior of GAE, since `javax.naming.InitialContext` is not one of the whitelisted JRE classes.

After quite some time of seeking how to resolve the problem, a solution was found. Someone had recompiled (unofficial) one of “JAR” files which was causing the problem. After implementing that solution everything worked well.

5.5.1.2 Communication with GAE Datastore

From the very beginning of developing the sample application it was found very interesting to work with non-relational database, as no previous experience with such type of databases was possessed. Although the concept of the Datastore itself seemed much easier than the one of relational databases, there was no approach that could be thought of for communicating among the Datastore and the application. Thus appropriate research had to be conducted.

Since GAE offers various approaches for communicating with database, initially it was thought to use one of them, especially low level API.

Below is the schema of such approach.

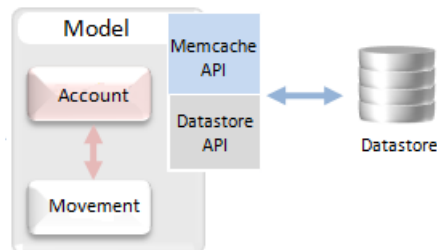


Figure 5.15 Overall Model-Datastore schema

Although after conducting an in-depth research, and realizing that those approaches are either incomplete, very complex or error prone it was decided to use Objectify.

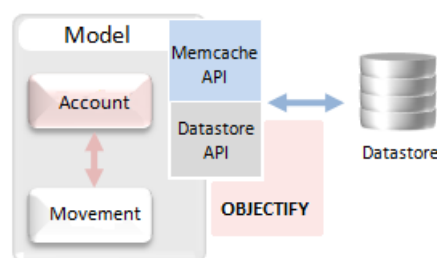


Figure 5.16 General Model-Datastore schema with implementation of Objectify

5.5.1.3 Application Performance

Once the application was deployed on GAE platform, there were observed some issues regarding its performance, i.e. the startup time was unexpectedly long. This issue at first seemed very odd as there was no information about any error in the GAE logs. Finally it was deduced that there should be at least one instance of application running, and as such a cron job invoking instance of an application was implemented. The only thing necessary to set the cron job was to create a cron.xml file and put in the WEB-INF folder, and within that file specify the following job.

```
<?xml version="1.0" encoding="UTF-8"?>
<cronentries>
  <cron>
    <url>/</url>
    <description>Refresh every 10 min</description>
    <schedule>every 10 minutes</schedule>
  </cron>
</cronentries>
```

5.5.1.4 Serialization Issues

During the process of development quite often the lines of code had to be rewritten due to GAE not allowing non-serializable objects. As a matter of fact every Java application running on GAE has to have its classes implement serializable and all types have to be serializable as well. Such issue was omitted couple of times by marking the property as transient (so it was skipped during serialization) and introducing lazy loading in the getter.

5.5.1.5 Memcache

While developing the pilot application it was not clear how to implement this API. After many struggles it was implemented as shown below.

The figure below represents the general flow of how Memcache API works within the pilot application.

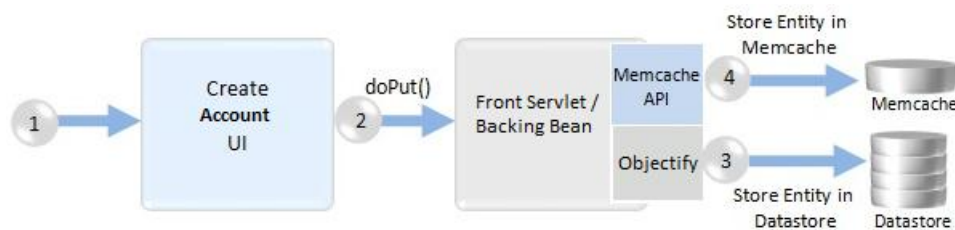


Figure 5.17 General Flow of Memcache on the Account example

Where,

1. Front Servlet is invoked once the user clicks on Create Account Button
2. doPut() method of the backing bean invokes Objectify to persist the data into Datastore
3. data is stored in the Datastore
4. The same data is stored in Memcache

5.5.2 Further Observations

5.5.2.1 Tests

Once the project development was done, it became important to check whether the quality of code - in terms of Google App Engine - was high. As it turns out App Engine offers “Local Unit Testing for Java”. It allows running tests that stay inside the development environment without involvement of remote components. App Engine provides testing utilities that use local implementations of Datastore and other App Engine services. Thus one can exercise their code's use of these services locally, without deploying the code to App Engine, by using service stubs. A service stub is a method simulating the behavior of the service. For example, the Datastore service stub allows testing the Datastore code without making any requests to the real Datastore. Any entity stored during a Datastore unit test is held in memory, not in the Datastore, and is deleted after the test run. Thus one can run small, fast tests without any dependency on Datastore itself [42].

5.5.2.2 Chat Implementation

It was found interesting how the chat function had to be implemented through the Channel API. Below is the detailed description of the role of the main components involved in process. The description was partially borrowed from the source [43].

JavaScript Client - User interacts with a JavaScript client within the website. The JavaScript client is mainly responsible for three things:

Connecting to the channel once it receives the channel's unique token from the server
 Listening on the channel for updates regarding other clients and making appropriate use of the data and changes in the webpage
 Sending update messages to the server

Server – The server is responsible for:

- Creating a unique channel for individual JavaScript clients
- Creating and sending a unique token to each JavaScript client so they may connect and listen to their channel
- Receiving update messages from clients
- Sending update messages to clients via their channels

Client ID - The Client ID is responsible for identifying individual JavaScript clients on the server. The server knows what channel on which to send a particular message because of the Client ID.

Tokens - Tokens are responsible for allowing the JavaScript Client to connect and listen to the channel created for it.

Channel - A channel is a one-way communication path through which the server sends updates to a specific JavaScript client identified by its Client ID. The server receives updates from clients via POST and then sends the messages to relevant clients via their channels. Channel Service allows you to manage two-way connections with clients.

Message - Messages are sent via POST from one client to the server. Once received, the server passes the message to the designated client via the correct channel identified by the Client ID.

Socket - The JavaScript client opens a socket using the token provided by the server. It uses the socket to listen for updates on the channel.

6 Planning and Economic Study of the Project

This chapter will consist of all the considered aspects at the time of workload distribution and tasks to accomplish. As well there is a subchapter detailing the estimation of the costs related to project.

6.1 Project Planning

There are two subchapters dedicated to description of the project planning i.e. initial planning and actual planning. These subchapters are a key to understand how the project was being developed, and what the initial assumptions were in comparison to the final planning.

6.1.1 Initial planning

The initial project planning was done with knowledge that it might be a subject to change as with time, once a particular subject is understood, number of milestones/subjects will grow.

On the next page the initial planning Gantt Diagram is presented. That diagram has a total duration of 111 days with the delivery due to 27th of August.

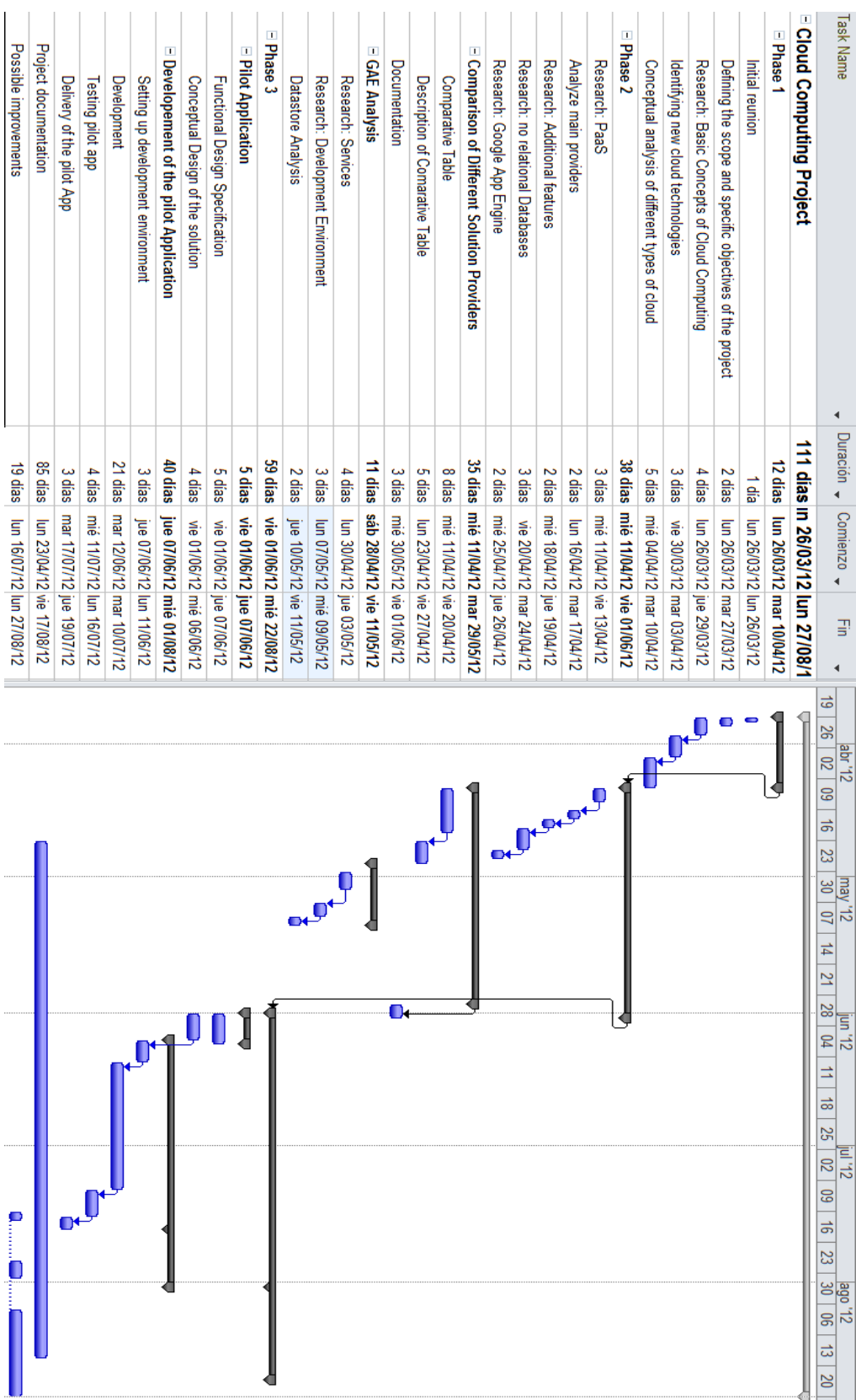


Figure 6.1 Initial project planning Gantt diagram

6.1.2 Actual planning

Once the project is done, it is possible to observe the real planning of the project. As expected from the very beginning, the real planning differentiates from the original, and those differences are due:

- *Lack of sufficient knowledge* – The duration and existence of some subjects could not be determined from the beginning as there was more in-depth knowledge necessary, and such knowledge was being achieved during the process of developing the project.
- *Importance of subjects* – Some of the tasks were not given enough attention for their proper development (Pilot Application), and some of them were given too much importance. Also in some cases tasks were removed or exchanged with other ones due to their insignificance to the project.

On the next page is presented the final planning Gantt diagram that has five days more than the initial diagram. Thus in order to finish the project it was necessary to extend the final date with five days which were given as a margin backup days and were not considered in the initial planning.

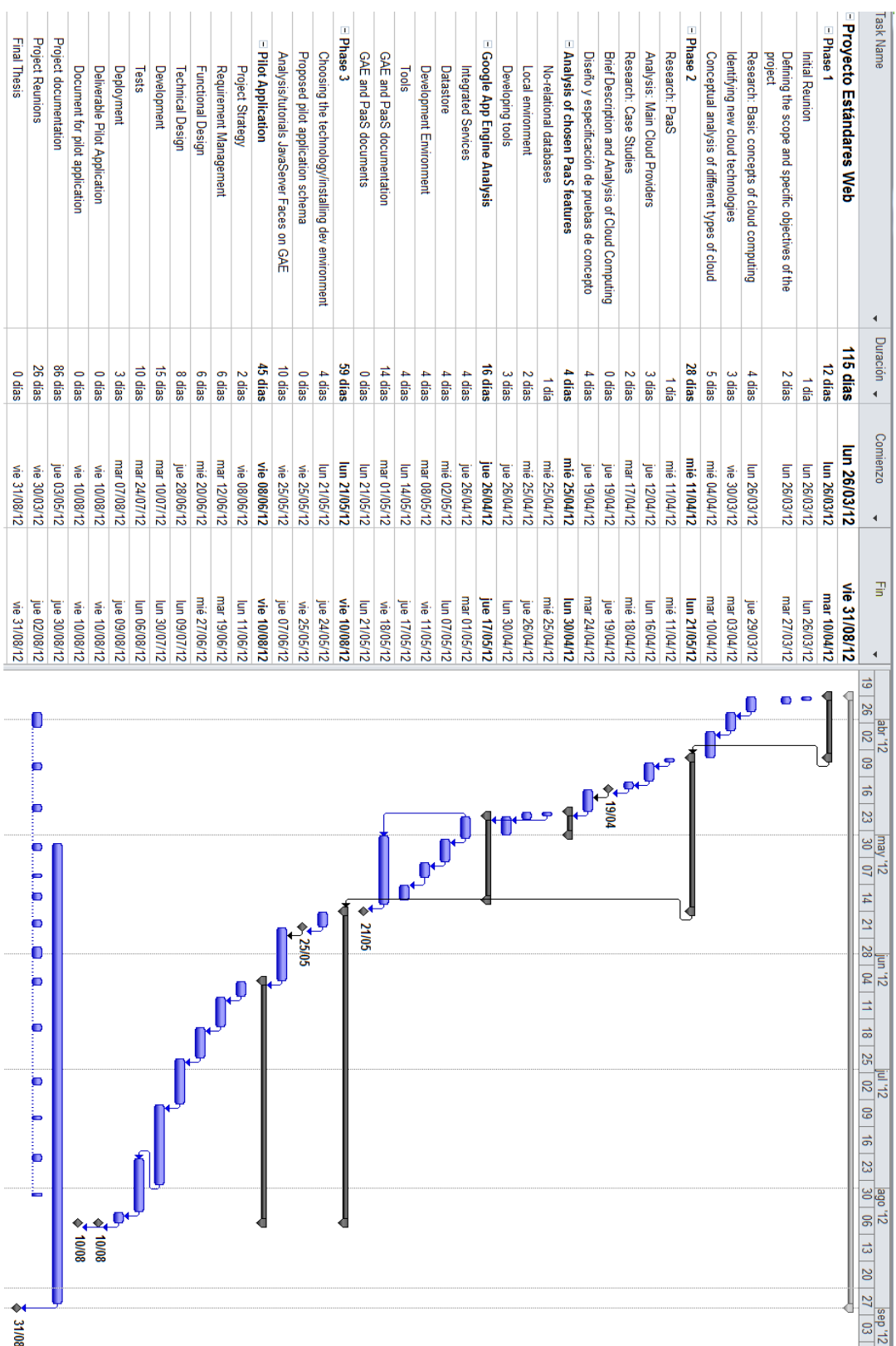


Figure 6.2 Final project planning Gantt diagram

6.2 Economic Study

This subchapter will estimate the costs according to the final planning of the project.

Firstly, in order to determine the salary of the intern, the value of one hour of work according to the agreement “universidad-empresa” of the UPC was considered. For the company the actual cost is of 7.5 €.

Secondly, it was considered that according to the agreement, Everis had to pay additional value of 14.7% of the total project cost to the university.

The following is the table with details corresponding to the resources or people employed in the project development.

Role	Hours	€/h	Total Cost in euros
Intern	920	7.5	6900
Consultant	23	30	690
Total			7590

Table 6.1 Costs of the resources

The subsequent table represents the costs corresponding to the agreement fees, and materials and services used during the project development.

Nature	Description	Total Cost in Euros
Agreement Fees	14,7% from intern cost	911,4
Materials and services	Office materials, basic services, printing	300
Total		1211.4

Table 6.2 Agreement Fees, and Material and Services costs

The company also provided a laptop for developing the project, and because of that the depreciation was considered. Below is the table with depreciation calculation.

Device	Initial Value	#Months	%depreciation	Depreciation Cost
Laptop	500	5	$(5 * 100)/36=13.8$	69

Table 6.3 Device depreciation

The last table represents the final total cost of the project basing on the previous calculation.

Nature	Cost in euros
Resources (Consultant, Intern)	7590
Contract Agreement Fees	911.4
Materials and services	300
Laptop Depreciation	69
Total	8870,4

Table 4.4 Final cost of the project

7 Conclusions

During the development of the project and the thesis itself, each of the objectives defined in the beginning was being accomplished progressively.

Following the initially determined structure, the subsequent tasks were successfully accomplished:

- A. There was given a description of Cloud Computing in general, its deployment models, and service models.
- B. Leading providers of Platform as a Service cloud were defined and compared.
- C. Google App Engine offering was described in-detail allowing for better understanding of Google's PaaS platform.
- D. There was developed a small pilot application in order to test against some of the Google's integrated APIs and discover the way they function.

From each of the above-mentioned points various important observations were concluded and presented below.

A.1 Cloud Computing is a new term based on already existing technologies. Thus the companies such Amazon or Google started offering what they already possessed but before it was for their proper use only.

A.2 Cloud computing is determined by its four deployment and three service models, which define the nature of the cloud, where private deployment is thought to be the most secure, though limited. Thus there is growing tendency towards the Hybrid deployment which merges the security of the private cloud with the unlimited resources of the public cloud.

B.1 After identifying and comparing the main providers of PaaS cloud it was deduced that PaaS is still in its early stage of development as many providers are offering PaaS via their IaaS offering. Thus this is kind of a workaround, not a solution. Moreover some implementations of PaaS are offered only in theory such as the IBM or Oracle offerings which offer just their pilot program and do not provide enough information and as such were possible to evaluate.

B.2 Current PaaS offerings offer diversity, but not necessarily stability as those platforms are still evolving and in many cases are a subject to change. At the time of writing the thesis it seemed that Google App Engine, BeanStalk, Heroku, and

Force.com were the most matured platforms at a time, though the rest of the offerings look promising.

C.1 Google App Engine integrates wide range of services which are easily implementable. Because of these services, GAE is not only another PaaS offering, but it also makes many development tasks much easier. Moreover, GAE offers variety of its tools, and its proper development environment that can become a necessity to a Java developer.

C.2 Google's PaaS cloud prime database is its Datastore which is a part of Google's proprietary non-relational Bigtable. Nevertheless it is not the only Google database offering as there is offered a relational (traditional) SQL database called Cloud SQL.

D.1 Sample pilot application developed for needs of this thesis have proven that different frameworks and technologies can be easily implemented within GAE environment. It also showed that monitoring the application is quite handy providing set of Google's tools.

D.2 Implementation of the pilot application gave an opportunity to evaluate services provided by GAE, and Datastore. The services under evaluation were found very easy to apply, and certainly those services can give an edge to a developer using them. Regarding the Datastore it was found very different from the relational (SQL) approaches, but for big projects it provides a better performance.

Nowadays Cloud Computing is still evolving. Nevertheless, it seems that now is the period when organizations should consider moving into the cloud as many companies gain competitive advantage due to migrating to the cloud already. The cloud as we know it may not be fully evolved, but it certainly achieved the stage of being cost-effective. Some organizations have doubts when it comes to implement a cloud, mainly because of the security, and private data exposure. Nowadays those two fears seem not necessarily applicable as most of the cloud providers spend bigger amount of money for securing the data than any other company ever will. Regarding a data exposure fear, normally providers offer very strict SLAs. In any case for those who still do not want to commit, there is possibility of choosing a private or even more performance-effective hybrid cloud. Whatever the choice, it seems that now is the time to join the cloud.

8 Bibliography

- [1] Google Documentation; Java Overview
<https://developers.google.com/appengine/docs/java/overview>
- [2] Google Documentation; APIs Overview
<https://developers.google.com/appengine/docs/java/apis>
- [3] Google Documentation; Datastore Overview
<https://developers.google.com/appengine/docs/java/datastore/>
- [4] Programming Google App Engine
Dan Sanderson, Programming Google App Engine, Editorial O'Reilly Media Inc, 2010
- [5] Google White Papers; BigTable
<http://research.google.com/archive/bigtable.html>
- [6] Datastore Frameworks: The interview
http://borglin.net/gwt-project/?page_id=604
- [7] Google Documentation; Quotas
<https://developers.google.com/appengine/docs/quotas>
- [8] Google Documentation; About Google Cloud SQL
<https://developers.google.com/cloud-sql/docs/introduction>
- [9] AWS Elastic BeanStalk
<http://aws.amazon.com/en/elasticbeanstalk/>
- [10] Cloud Computing. Principles and Paradigms
Rajkumar Buyya, James Broberg, Andrzej Goscinski, John Wiley & Sons, Inc., 2011
- [11] An overview of the cloud market vendor landscape
<http://www.vmware.com/files/pdf/cloud/VMware-Taneja-Group-An-Overview-Of-The-Cloud-Market.pdf>
- [12] VMware tests vSphere resilience with Cloud Foundry
<http://www.zdnet.com/vmware-tests-vsphere-resilience-with-cloud-foundry-3040093821/>
- [13] NoSQL – The Trend for Databases in the Cloud?
<http://cloudcomputing.sys-con.com/node/1615716>
- [14] Comparison of Cloud Database: Amazon's SimpleDB and Google's Bigtable
R.Shalini, Savita Goel, A.Subramanian, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 6, No 2, November 2011
- [15] Force.com for Google App Engine; Connecting the clouds
<http://developer.force.com/appengine>
- [16] How to Use the Service Bus Relay Service
<https://www.windowsazure.com/en-us/develop/net/how-to-guides/service-bus-relay/>
- [17] Gartner Reference Model for PaaS
Yefim V. Natis, Gartner, Inc., 28 September, 2011
- [18] Gartner; Hype Cycle for Cloud Computing
David Mitchell Smith, Gartner, Inc., 27 July 2011
- [19] Platform as a Service: Definition, Taxonomy and Vendor Landscape, 2012
Yefim V. Natis, Michele Cantara, Joseph Feiman, Gartner, Inc., 19 March 2012

- [20] A PaaS that runs anything HTTP: Getting Started with DIY Applications on OpenShift
<https://openshift.redhat.com/community/blogs/a-paas-that-runs-anything-http-getting-started-with-diy-applications-on-openshift>
- [21] The NIST Definition of Cloud Computing,
 Peter Mell, Timothy Grance, Recommendations of the National Institute of Standards and Technology, U.S. Department of Commerce
- [22] The Big Switch: Rewiring the World, from Edison to Google. W. W.
 N. Carr, Norton & Co., New York, 2008.
- [23] A dynamic VPN architecture for private cloud computing
 Wen-Hwa Liao, Shuo-Chun Su, Proceedings of the 2011 IEEE 4th International Conference on Utility and Cloud Computing (UCC 2011), IEEE Computer Society, 2011
- [24] The Cloud at your Service
 Jothy RosenberG, ARthuR MAteos, The Cloud at Your Service, Manning Publications, 2011
- [25] The Google-ization of Bechtel
http://www.networkworld.com/tundra/issues/110308_NWW2.pdf
- [26] Comparing Public-Cloud Providers
 Ang Li , Xiaowei Yang ; Kandula, S. ; Ming Zhang , IEEE Computer Society ,March-April 2011
- [27] NIST Cloud Computing Reference Architecture
http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505
- [28] Guidelines on Security and Privacy in Public Cloud Computing
<http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf>
- [29] Cloud Computing Security
<https://publications.theseus.fi/bitstream/handle/10024/31888/Cloudsecurity.pdf>
- [30] Community Cloud Computing
<http://www.springerlink.com/content/q6416rvj81n72303/fulltext.pdf>
- [31] Cloud Computing Synopsis and Recommendations
<http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf>
- [32] Cloud Computing. Principles and Paradigms
 Rajkumar Buyya, James Broberg, Andrzej Goscinski, John Wiley & Sons, Inc., 2011
- [33] BBVA banks on Google
<http://googleenterprise.blogspot.com.es/2012/01/bbva-banks-on-google-apps.html>
- [34] Google Apps for Business aporta a IRB Barcelona mucho más que ahorro de costes
https://docs.google.com/file/d/0B_wlnRj8bDj-NGExN2YzMDYtNTdhYS00NDQ3LWFfODctYmFIMWU3YjNiYjVh/edit?hl=en
- [35] Google Apps
http://en.wikipedia.org/wiki/Google_Apps
- [36] Salesforce.com
<http://en.wikipedia.org/wiki/Salesforce.com>
- [37] Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS
 Ali Khajeh-Hosseini, David Greenwood, Ian Sommerville, Proceedings - 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD 2010, p 450-457, 2010
- [38] Amazon Web Services
http://en.wikipedia.org/wiki/Amazon_Web_Services

[39] Introduction to cloud IaaS providers

<http://bizcloudnetwork.com/introduction-to-cloud-iaas-providers>

[40] Understanding PaaS

Michael P. McGrath, Understanding PaaS, O'Reilly Media, 2012

[41] Understanding the Cloud Computing Stack

http://broadcast.rackspace.com/hosting_knowledge/whitepapers/Understanding-the-Cloud-Computing-Stack.pdf

[42] Local Unit Testing for Java

<https://developers.google.com/appengine/docs/java/tools/localunittesting>

[43] Channel API Overview

<https://developers.google.com/appengine/docs/java/channel/overview>