

**Functional Design**

*Project: Virtualisation*

|  |  |  |  |
| --- | --- | --- | --- |
| **Client: Plaintech** |  | **Filename:** | Functional Design v1.0.docx |
| **Project: Virtualisation** |  | **Version:** | 1.0 |
| **Author: Willem Westerhof, Pieter Dieleman, Eddy van der Steen, Kjell Zijlemaker, Rodney Lanuzga** |  | **Date:** | 3-10-2014 |

# Documentproperties

## History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Changes(concept/final)** | **Client** | **Author(s)** |
| 0.1 | 30-9-2014 | Creation of file with basic Itopia setup | Plaintech | Willem Westerhof |
| 0.2 | 30-9-2014 | Added chapter 1,2,3, and parts of 4 | Plaintech | Willem Westerhof |
| 0.3 | 02-10-2014 | Added chapter content | Plaintech | Willem Westerhof, Kjell Zijlemaker, Rodney Lagunza |
| 0.4 | 03-10-2014 | Added chapter content | Plaintech | Willem Westerhof, Kjell Zijlemaker, Rodney Lagunza |
| 1.0 | 3-10-2014 | Grammer check and layout | Plaintech | Willem Westerhof |

## Approval

This document needs the following signatures of approval:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Role** | **Autograph** | **Document Date** | **Version** |
| Arjen Jansen | Project manager |  | 3-10-2014 | 1.0 |
| CEO/CFO | Client |  | 3-10-2014 | 1.0 |

# Table of Contents

Inhoud

[Documentproperties 2](#_Toc400123880)

[History 2](#_Toc400123881)

[Approval 2](#_Toc400123882)

[Table of Contents 3](#_Toc400123883)

[Introduction 4](#_Toc400123884)

[Context 5](#_Toc400123885)

[MoSCoW 6](#_Toc400123886)

[Business processes and use cases 7](#_Toc400123887)

[case 1: 7](#_Toc400123888)

[Case 2: 8](#_Toc400123889)

[Case 3: 9](#_Toc400123890)

[Case 5: 11](#_Toc400123891)

[Case 6: 12](#_Toc400123892)

[Case 8: 14](#_Toc400123893)

[Case 9: 15](#_Toc400123894)

[System architecture and Major components 17](#_Toc400123895)

[The Infrastructure 17](#_Toc400123896)

[Logical Data structure 22](#_Toc400123897)

[Logical Data Diagram 25](#_Toc400123898)

# Introduction

In this document you will find a description of the functionalities supplied by the proof of concept. This functional design will describe different situations and use cases using modelling languages. Note that all models are based on the request for proposal as supplied by Plaintech as well as the requirements stated in the PID. If changes are made to these requirements a new version of this document will be published by Itopia.

# Context

Plaintech UK is an organization with its headquarters in Birmingham. The company currently has a Windows-only platform with no options to extend their platform for more diversity. Because of this, the company has great limitations for extending their platform. As a result, Plaintech UK wants to extend to a platform that supports both Linux and Windows hosts. This will be serviced by a Linux-based virtualization platform.

Plaintech currently host approximately 50.000 hosts. These customers are regular and business users and have no limitations in buying new hosts per customer. The customers are connected through the internet and have a connection to the Access Router at Plaintech. Here, the hosts are connected through the server network with all the virtualization servers available. The exact number of servers is unknown, but has to support the load of 50.000 hosts. Plaintech also has the corporate application server in this network to provide applications for the employees.  
Plaintech’s primary goal is to move their current physical (Windows-only) platform to a new Linux-based virtualized platform that allows for both Windows and Unix hosts.  
This platform must consist of a Linux based operating system that will support the KVM. Also, the infrastructure needs to be updated for the needed security, load balancing and speed for the customers.

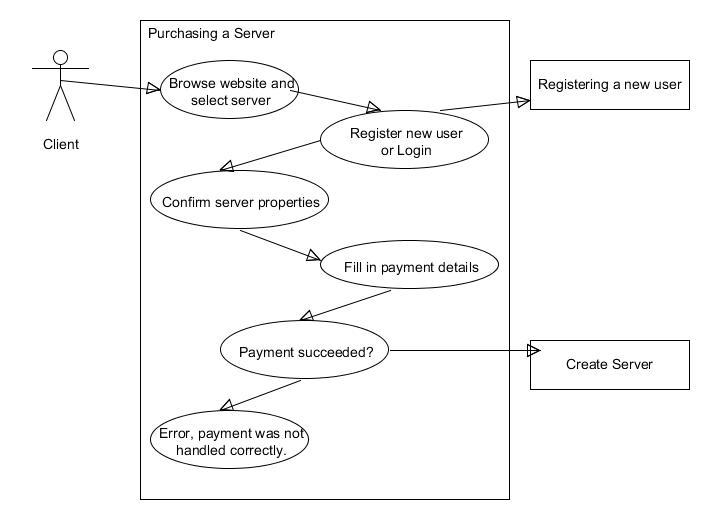
Plaintech would also like to know the total cost saving that they should achieve when accepting the project proposal. Therefore, there will be document where we will raise this matter.

Because all the above things must be changed completely, it’s also important that we’ll know exactly what the cost will be, and if there will be a radical change in  the management of the infrastructure.

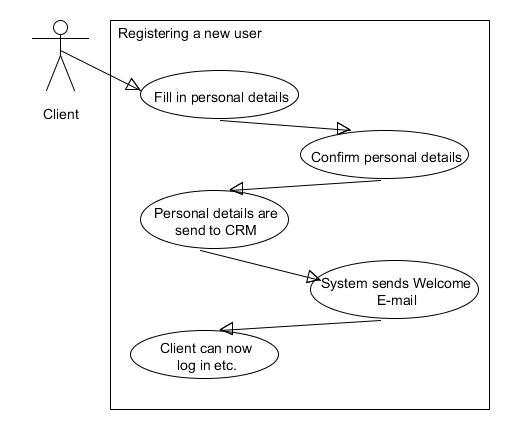
# MoSCoW

The proof of concept will be detailed by a MoSCoW approach.   
  
Functional:  
Must have:  
All website functions can be used correctly.  
The website has an admin control panel to acces the CRM system.  
Serverside scripts that create and manage virtual machines as specified by the user or admin.  
  
Should have:  
Security across the entire platform.  
A virtual network setup that is scalable to the full scale implementation.  
A system that automatically creates daily backups.  
  
Could have:  
High availability.  
Off-site backup  
  
Won’t have:  
-  
  
Nonfunctional:  
Must have:  
Website has to look attractive.  
Website is user friendly.  
Scripts and code must be commented.  
Code must compliant with the itopia standards.  
  
Should have:  
Creating or managing a virtual machine has to be responsive.  
  
Could have:  
-  
Won’t have: **-**

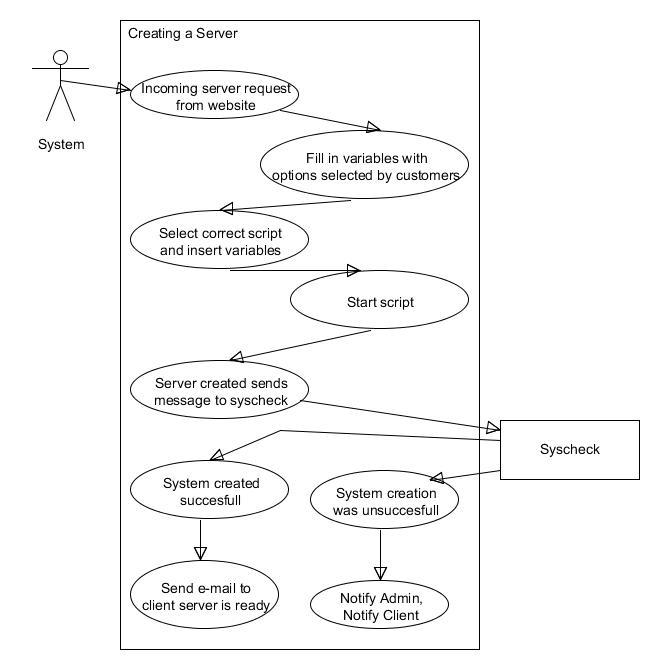
# Business processes and use cases

To make sure all functionalities are covered. We will provide a number of Diagrams here. These diagrams provide specific situations or functionalities. Each individual diagram is explained in a small description following the diagram.  
  
  
  
  


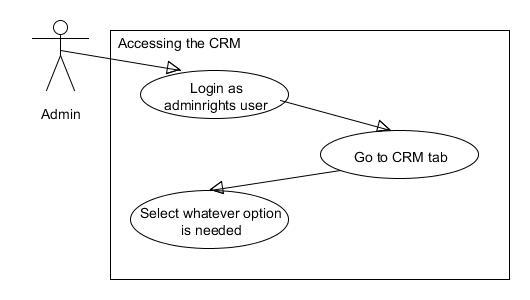
case 1:  
This use case shows the process of purchasing a server. IT also has 2 side steps toward other use cases. These use cases will be discussed further on in this chapter. First off the user browses through the Plaintech website and selects whatever specs he wants for his server. After the client registers or logs in he/she can confirm the given properties and fill in the exact payment details. If this payment is a success the create server use case is started. If payment is not successful an error occurs and server creation is not started.

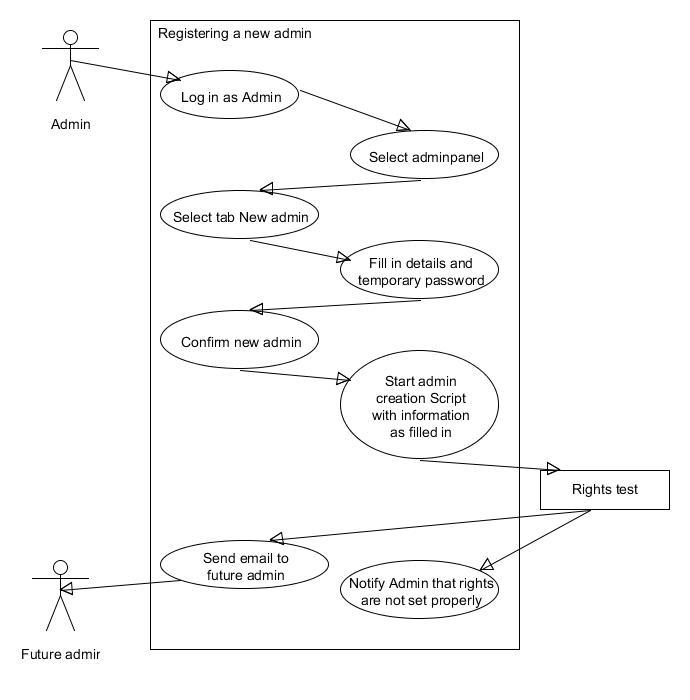


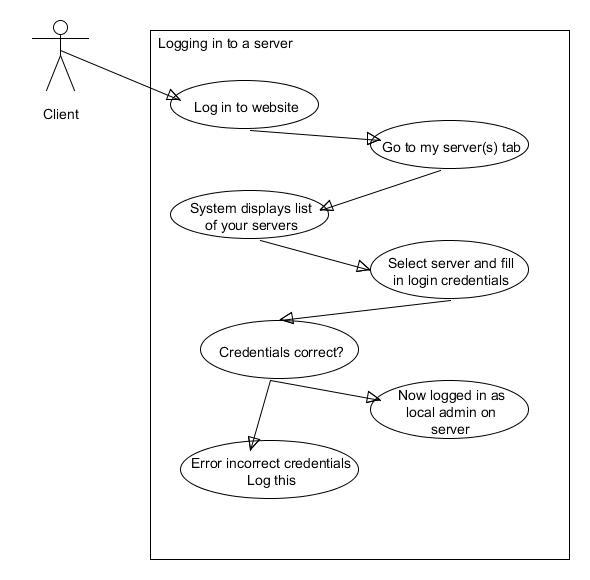
Case 2:  
As seen in case 1 a register or login is required to order a server. This use case describes the registering process. First the client fills in his or her personal details and confirms that these details are filled in correct. These personal details are then send to the CRM and noted in the Database, The client will now automatically be send an email with his/her login information and a welcome message. After this message is received the client can login to the website using his/her account name and case 1 continues.



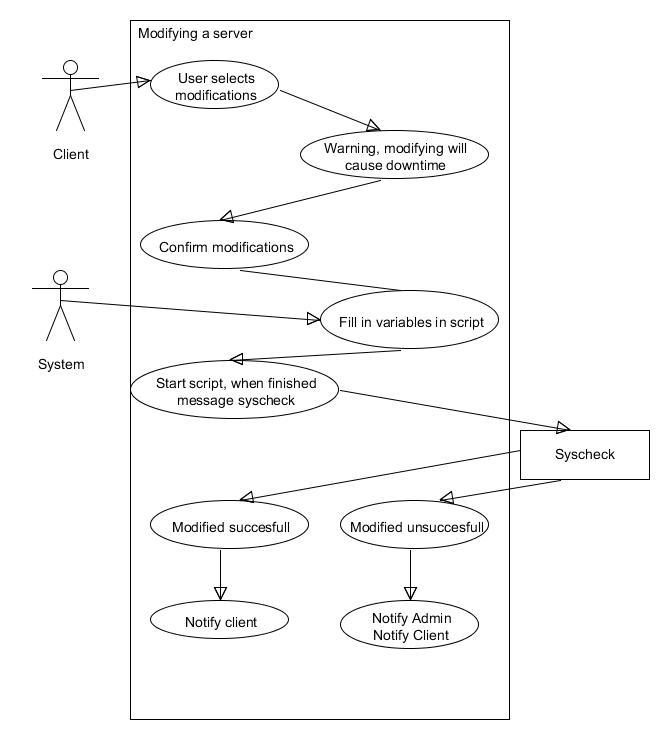
Case 3:  
As seen in case 1, when payment is received correctly this use case starts. The website sends a request toward our KVM server with the server details as provided by the user. These server details are filled in specific variables and inserted into the script. This script will then create the virtual server as requested by the client. As soon as the creation process is finished a message is sent to the Syscheck. This is a testing script that tests if the new server is accessible and fully functional. If the test is a success the syscheck messages the client that the server is ready and waiting. If the test is unsuccessful a Plaintech admin is notified with the exact error report. The client is also notified that his requested server will take a bit longer to be created due to technical issues. It is then up to the admin to debug the given errors and find out why server creation wasn’t successful.

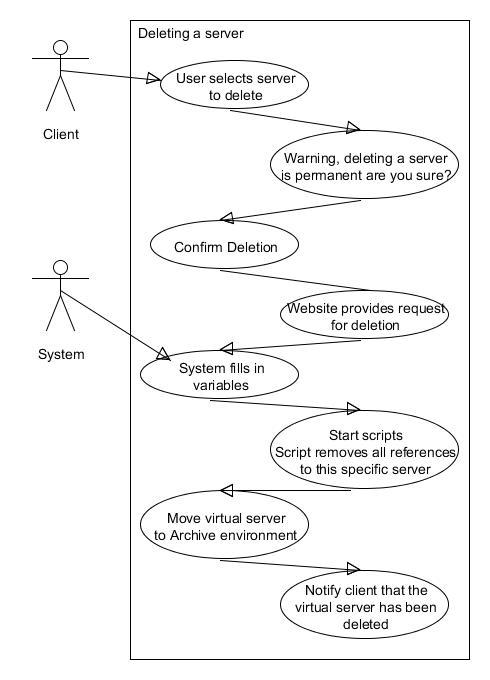
  
  
Case 4:  
In previous use cases we’ve talked about admins and the CRM. This use case shows the basic extra functionality that the admins have. When a admin logs in to the website using his admin account they get an extra CRM option. In this CRM tab there are a number of additional options which are privileged for admins only. New options can also be added to this CRM tab when necessary it is mainly used for gathering client information such as payment details, running servers etc.



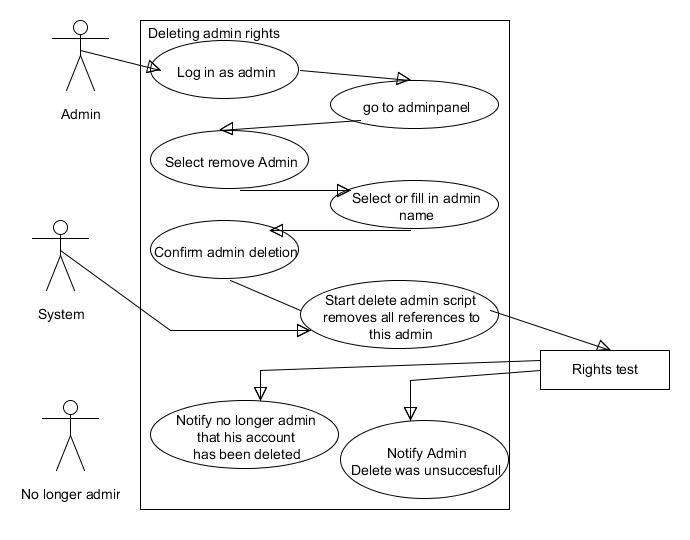
Case 5:  
Of course Plaintech will need more than one Admin, as well as the option to expand the number of admins. To do this we have created this use case. First a current admin logs in and selects the Admin Panel tab. The admin then selects the option to create a new admin and fills in the details of the future admin as well as a temporary password. He then confirms all information is entered correctly. The system then starts the Admin creation script with the filled in information and when this script is finished a rights test is started. This rights test is performed to check if the future admin indeed has the correct rights to access certain folders. If this test is a success the future admin is informed, if not the current admin is informed that rights are not set properly.  
  
  
  


Case 6:  
Let’s go back to the client’s point of view. The client has been informed that his virtual server is up and running and wants to access his server. This client first signs in to the Plaintech website and goes to the my server(s) tab. The system retrieves a list based on the client's username to display his servers. The client then selects a server and fills in his server login credentials. If these credentials are correct he is now logged in as local admin on his server, if these credentials are not correct the connection is refused and an error is given. These errors are logged for analysis purposes.

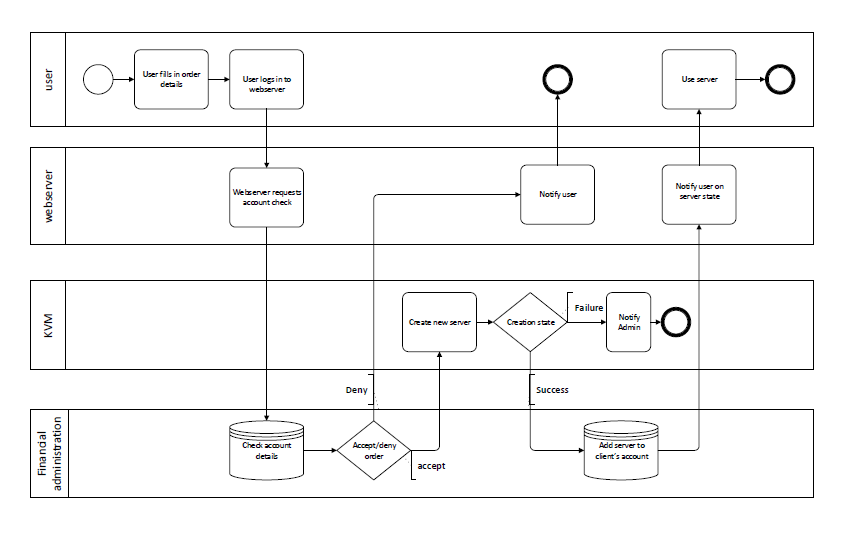
  
Case 7:  
Now let’s say our client’s business has grown quite a bit and he needs some more server capacity. The user then selects their server (as previously explained) and selects modifications and upgrades (or downgrades) his virtual components. A popup message will then warn the user that modifying the server causes downtime. After the user confirms that he wants the modifications the modifications will be filled in to a modify script. This script is then started and when finished a syscheck is started to make sure everything is in working order. If this modification is successful the client is notified by email if this modification is unsuccessful the admin as well as the client are notified. It is then up to the admin to make sure the clients server will become functional again.



Case 8:Of course some clients will also go bankrupt or leave Plaintech for some reason. If this happens the client can select the server he wants to delete, a warning is then given that deleting a server is permanent. After confirming this deletion the website sends a request for deletion to the system. This system fills in the specifics of the machine that needs to be deleted. After that scripts are started that remove all references to this specific server the server is moved to the archive environment. Here the raw data of the server is maintained for a while for archiving purposes. After this server is archived the client is notified that his server has been deleted.



Case 9:  
As time progresses some of the Plaintech admins will retire, find another job etc. To delete this persons admin account another admin goes to the admin panel and selects the remove admin option. He fills in or selects the admin name and confirms the deletion. A script is then started that removes all references from this admin. The rights test is again called to check if all admin rights for this person are deleted. If all admin rights are deleted the person who no longer is an admin will be notified that his account has been deleted. If for some reason the test is unsuccessful the deleting Admin is notified that the delete was unsuccessful.

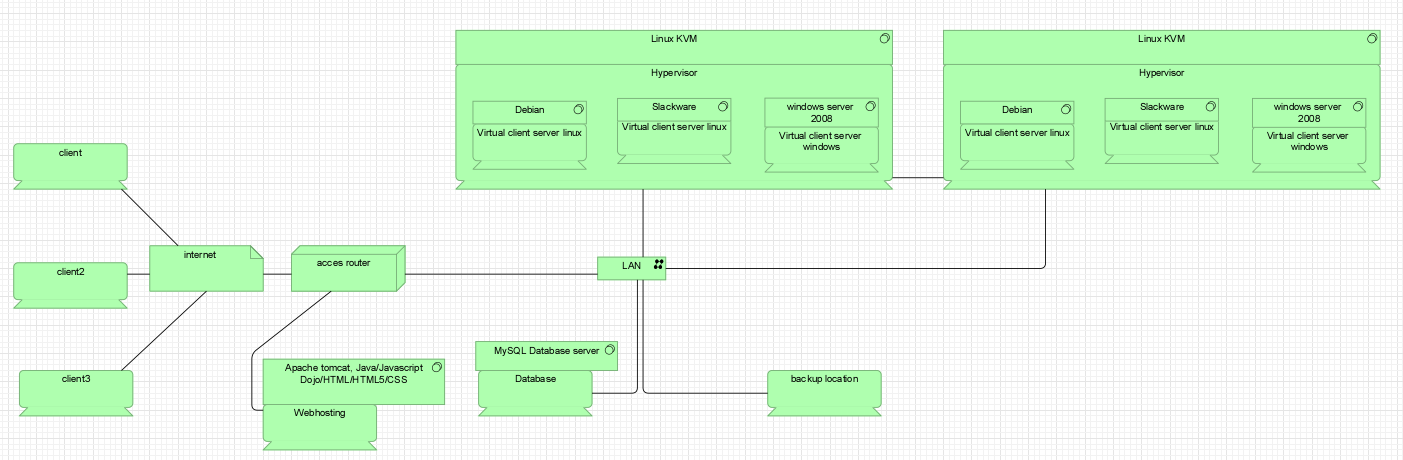
  
  
Case 10:  
The above model is a BPMN model. This model is added to this document to show what the different use cases add up to in general lines. It is divided in 4 lanes, the user lane which involves all user actions, the webserver lane which involves the actual website and it’s functions, the KVM lane which creates new servers and the financial administration lane which registers information and finances. This model gives a basic explanation of the primary process of plaintech which is selling virtual servers.

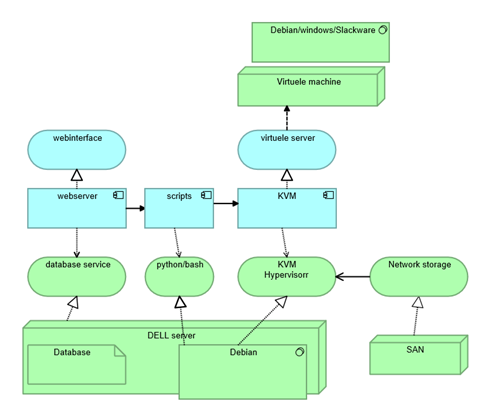
# System architecture and Major components

The system architecture consists of the entire hardware infrastructure for project virtualization.

In this part we will explain why we designed the infrastructure like this and what major components are available in this infrastructure.

## The Infrastructure

  
(hardware layer)

  
(software layer)

The infrastructure we created is based on the environment, quality attributes and specific requirements of Plaintech. The main function of the new infrastructure is that the customer will go to the Plaintech website and can then create and maintain their own virtual machine without any assistance of Plaintech employees.

To complete the processes that are required for the plaintech website we need quite a number of major components. A list of these major components and a short explanation of their functions can be found below.

- Client

o The user will be making use of a client, also called a workstation or device. This can be any kind of device, mobile, pc, tablet etc. This device is used to go to the plaintech website and start the process.

§ The Client uses the following major software component:

· The Internetbrowser (Internet Explorer, Google Chrome, Safari etc.)

- Internet

o The internet is vital for the infrastructure. Without internet access clients are unable to reach their servers or to go to the plaintech website. It is used to set up any kind of connection from a client device to the virtual server.

- Router

o The Routers will be the first entry point for the customer to access the plaintech website to create a virtual machine or to gain control over their virtual machine. The customer will go on the Internet and enter the web address of plaintech. The router will then redirect them to the webserver where the website is running. If they already have a virtual machine and the user wants to make a remote connection with their credentials, the router will redirect them to their specific server on the KVM machine.

- LAN

o The LAN will be the bridge for the whole infrastructure. The LAN makes sure that all the hardware in the plaintech network can communicate with each other.

- Server

o In the infrastructure we have separate servers, each of these servers has their own specific functions.

In the infrastructure we have the following servers:

- Webhosting server

- Database Server

- KVM Server

- High Availability Server

- Backup Server

o Webhosting server

§ The webhosting server will host the plaintech website. Here the user will be able to make a request to create a virtual machine or make a remote connection through a console to gain control of the virtual machine. Users can also browse the webpage as normal.

§ The Webhosting server uses the following major software component:

· Apache tomcat

Apache tomcat is open source software to help create a webserver.

The apache tomcat program is compatible with the programming languages that we use for the plaintech website.

The main software components that we use to create the website are:

- Java ee / Java Servlets

- Dojo

- Velocity

- HTML5

- Rest

o Database Server

§ The Database server is where all the data of the customers will be stored. Some parts of the website are also dynamically added from the database.

· The Database Server uses the following major software components:

o SQL Database

The SQL database is where all the information of the customers will be stored and managed.

o KVM Server

§ The function of the KVM server is to create virtual machines. When the user logs in on to the plaintech website he can fill in his specification for the virtual machine. this will trigger a script that will send the users request to the server to create a virtual machine. When the server finishes his task it triggers another script that will notify the user that the virtual machine has been created and the user will then receive his credentials to gain access to his virtual machine.

o High Availability Server

§ The High availability server’s function is to prevent downtime of the hosts that runs the Virtual machines. When there is a hardware or software problem or the main KVM server is in need of maintenance the HA Server will take over without causing any downtime.

· The KVM and High Availability Server uses the following Major Software Components:

- Debian

Debian is a Linux Operating System. The OS will run on our kvm server to run our services and manage our virtual machines.

For managing the virtual machine we use the toolkit Libvirt.

- Libvirt

Libvirt is a managing tool that will create, modify and delete the Virtual machine. When a user wants to create a Virtual machine a script will run that triggers Libvirt to create a virtual machine. If the user already has a Virtual Machine the user can modify his specification and libvirt will apply the modification to the virtual machine.

o Backup Server

§ The Backup server’s function is to make sure that the valuable data of all the servers will be stored in a separate location. It is their to make sure that if a server goes down or data becomes corrupted a save backup is there to revert back too.

- SAN

o The function of the SAN is to provide storage capacity for the virtual machines.

§ For example when the user wants to create a fileserver for his company he will need disk space. From the plaintech website the user can request this disk space. It will trigger a script to send the request to the server that will assign extra disk space to the Virtual machine. The server will then assign the requested disk space from the SAN to the virtual machine and it will look like the user has an extra hard drive or more hard drive space on their file server to store data.

# Logical Data structure

Source: <http://www.learndatamodeling.com/ldm.php>  
See last page for the logical data structure diagram. **Decription**

This logical data structure has three layers:

* **The Webinterface Customer management**, for the customers that already have a virtual machine and wants to manage them.
* **The Webinterface Customer buying,** for the customers that wants to buy new virtual machines at the website.
* **Data for external scripts**, for the getting the data collections for sending them to the scripts

At the first layer we have the **Virtual machines** and the **Virtual machines settings**. These are the most important for getting the right data to the customer, and for changing the settings indeed.  
The following fields are present in those tables:

**Virtual machines**

* **MachineID**, Unique ID for every virtual machine the customer has
* **Machine Name,** Unique name for every virtual machine the customer has
* **Machine State**, Indication if the virtual machine is running or not
* **Machine Type,** Information what kind of level the virtual machine is (SLA)

*The relationship between the Virtual machines and settings is many to one (one to many): there can be several several virtual machines, but only one setting per virtual machine*

**Virtual machine Settings**

* **SettingsID,** Unique ID for every setting, per virtual machine
* **CPUSpeed,** The total speed the CPU currently has
* **MemoryAmount,** The total memory the VM currently has
* **StorageAmount,** The total storage amount the VM currenty has
* **OSType,** The OSType the VM currently has
* **MachineID,** The Foreign key for every machine. Also, this is unique because every VM can only have one set of settings

In the middle, we have the Users table. This is in the middle because it will act on either the management, as the buying side of the configuration. The fields of this table are as follows:

**Users**

* **UserID,** Unique ID for indicating every customer that bought, or will buy, a new VM
* **FrontName,** Front name from the user
* **LastName,** Last name from the user
* **Username,** User can choose a new username. This will need to be unique
* **Email,** Unique email for sending mail to the user
* **Password,** Password that the user will choose to login with
* **UserType,** The type the user will be. Admin or normal user

*The relationship between the users and virtual machines is that a user can have several virtual machines for managing. For buying, the user can have one or more products, but those products can be part of several or more users to buy.*

The second layer in the data structure model is the Customer buying section. Here, customers can buy the VM’s and customize them before production. The tables and fields are as follows:

**UserVirtualMachines**

* **UserID,** ID for every user per machine (or more)
* **VirtualmachineID,** ID for every machine per user (or more)
* **VirtualMachineQuantity,** The amount of VM’s the user wants to buy

*The relationship between the virtual machine for the user and the cart is that one or more users can have one or more products to buy (many to many)*

**VirtualMachinesCart**

* **VirtualMachineID,** The unique ID for every virtual machine the user wants to buy
* **VirtualMachineType,** The type of machine (SLA) the user wants to buy
* **VirtualMachinePrice,** The total price per virtual machine the user wants to buy
* **VirtualMachineLevel**

*The relationship between the virtual machine cart is: the type of virtual machine can have one or more virtual machines (one to many)*

**VirtualmachinesTypeCart**

* **VirtualMachineTypeID,** The unique ID for the type of virtual machine
* **VirtualMachineCPU,** The type of CPU that the user wants to buy
* **VirtualMachineMemory,** The type of machine memory the user wants to buy
* **VirtualMachineStorage,** The type of storage the user wants to buy
* **VirtualMachineOS,** The type OS the user wants to buy
* **VirtualMachineID,** The foreign key, ID, for the virtual machine

*The relationship between the virtual machines cart and the settings is: The type of a virtual machine can have one or more virtual machines (one to many)*

**VirtualMachinesTypeSettingsCart**

* **VirtualMachineTypeSettingsID,** The unique ID for every type per VM
* **CPUSPeed,** The total speed that is customizable for the user
* **MemoryAmount,** The total amount of memory that is customizable for the user
* **StorageAmount,** The total amount of storage that is customizable for the user
* **OSType,** The type of OS that is customizable for the user
* **TypeID,** The foreign key, ID, from the type of VM so the user can fill in the details per VM type

*The relationship between the virtual machines type cart is: One set of virtual machine settings can have only one type of virtual machine (one to many)*

The final part of the data structure are the sets of data that will be handled by the scripts. The scripts will be responsible for the correct bridge of making the virtual machine in conjunction to the KVM.  
The scripts will handle the following datasets:

**AddUser, EditUser, DeleteUser**

* **UserID,** The ID will be unique and important for adding, editing and deleting the user
* **Username**
* **FrontName**
* **LastName**
* **Email**
* **Password**

**AddVM, EditVM, DeleteVM**

* **VirtualMachineID,** The virtual machine ID will be unique and is important for adding, editing and deleting the virtual machine
* **VirtualMachineType**
* **VirtualMachineSettings**

# Logical Data Diagram

