Special Topics in Computer Science (Cloud Computing)
Johannes Kepler University Linz
2012 WS

Manage Cloud through Mobile Devices

Final Project Delivery

Professors:

Ismail Khalil
Matthias Steinbauer

Team members:

Andrei Toader-Stanescu (k1157739)
Ti-Ho, Chang (k1256447)

Table of Contents

1 Application Name

Manage Cloud through Mobile Devices

2 User group and usage model

2.1 User group:

We focus on two groups of target users, cloud platform administrators and customers of cloud platform. From administrators' point of view, it's an extra possibility for keeping the cloud platform operates properly; from customers' point of view, it's a convenient and interesting function for one to access the cloud from the mobile device. Both target users can fetch and operates the cloud platform.

However, from the application point of view, both cloud platform administrators and customers of cloud platform are defined as one and the same entity, "user" or "end user" in the following cases.

2.2 Usage model:

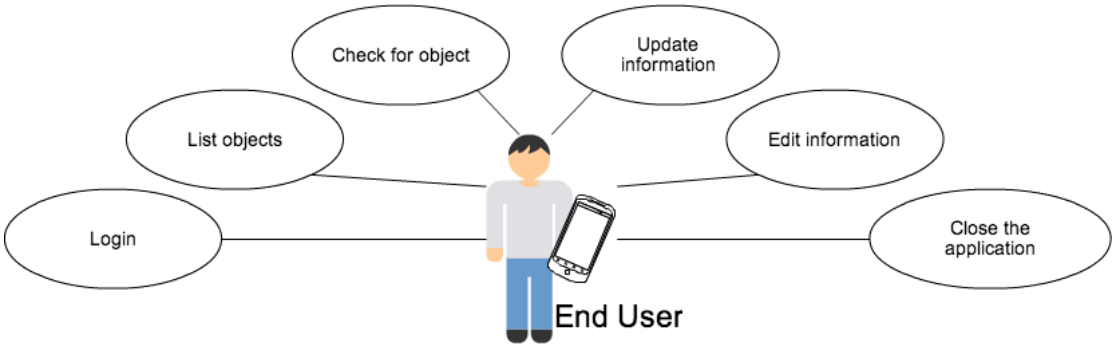
The usage model is easy. Users will first install the certification file into the android device, like you install it on windows. The mobile application would try first to authenticate to the server using the public key and then after the connection is established will try to authenticate using the private key stored in the certificate. So far, the operations about network, storage and computes are available.

2.2.1 Use cases

User	Operation	Description
End User	Login	User adds the credential information manually into the mobile device. Afterwards, if the certifications are installed, it will allow access to the Cloud service.
	List objects	User can choose the category (ex. Network) s/he wants to display. The list of chosen category will be displayed on the monitor.
	Check for object	User can choose the object (ex. localhost) s/he wants to see in detail. The detail information of the chosen object will be shown on the monitor.
	Update information	When the device is connected to the Internet, user can update the information of objects to

		the database.
	Edit information	When an object is displayed, the user has the possibility to change some of the properties, the ones which are allowed to be changed. If the servers supports the objects can be updated to the server.
	Close the application	User presses the back button on the phone to close the application.

2.2.2 Use cases diagram:



3 Application Description

3.1 Problem and Specification

3.1.1 Problem

According to DeLone and McLean's Information System Success Model[i], system quality is one of the most important factors for success information systems. Platform as a Service (PaaS) providers play the role on hosting and managing their customers' machines that usually couldn't stand for system failures. If a failure happened, PaaS providers should fix it in a short period of time.

We are trying here to offer a solution to improve the system quality of PaaS by letting system administrators monitor, adjust and manage the status and settings of their cloud platform anytime, anywhere – Using a mobile device.

3.1.2 Specification

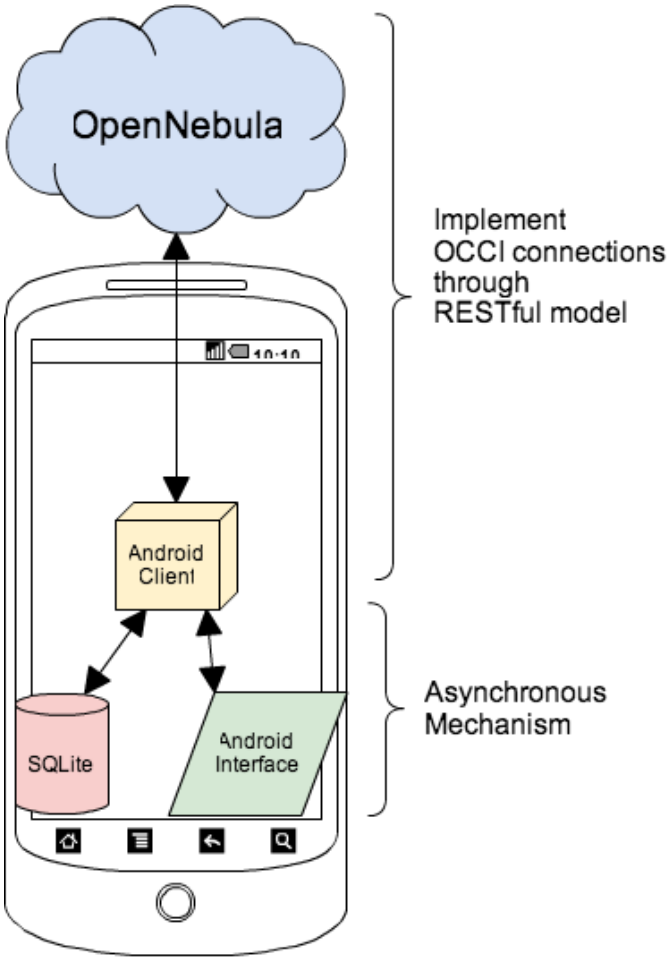
According to the statement of the problem, we come up with an application with a good architecture within the following specification:

- The application must be able to fetch the latest platform status through mobile devices when it's linked to the internet.
- When there is no internet connection, the application must be able to display the status of the platform.
- The application must ensure the safe connection between the mobile devices and the cloud platform.
- The application must provide functions for editing and updating platform contents.
- The application is based on the OCCI standards.
- The architecture is reusable, adjustable, maintainable, extendable and stable.

3.2 Application architecture

3.2.1 Components Interaction Flow

Fig. 1 Components Interaction Flow



3.2.2 Application Architecture

The application has, as it can be seen from figure 2, 4 major components. The components are Object Model, Database Model, the Data Access Manager, Synchronization Manager and the Rest Client.

3.2.2.1 XML

Because the xml received from the server is composed from xml elements, in order to construct a good and easy to use concept, the Object model is composed around two components, BaseElement and Properties. We define to object to be the entity from the root XML.

All the other elements are defined as to be properties of the object created from the root xml. Every property has a key and a value. For example in the case of the xml <NETWORK>….<GROUP>students</GROUP>…</NETWORK> the key is GROUP and the value is students. After the xml is parsed, every object is composed to contain a list of properties.

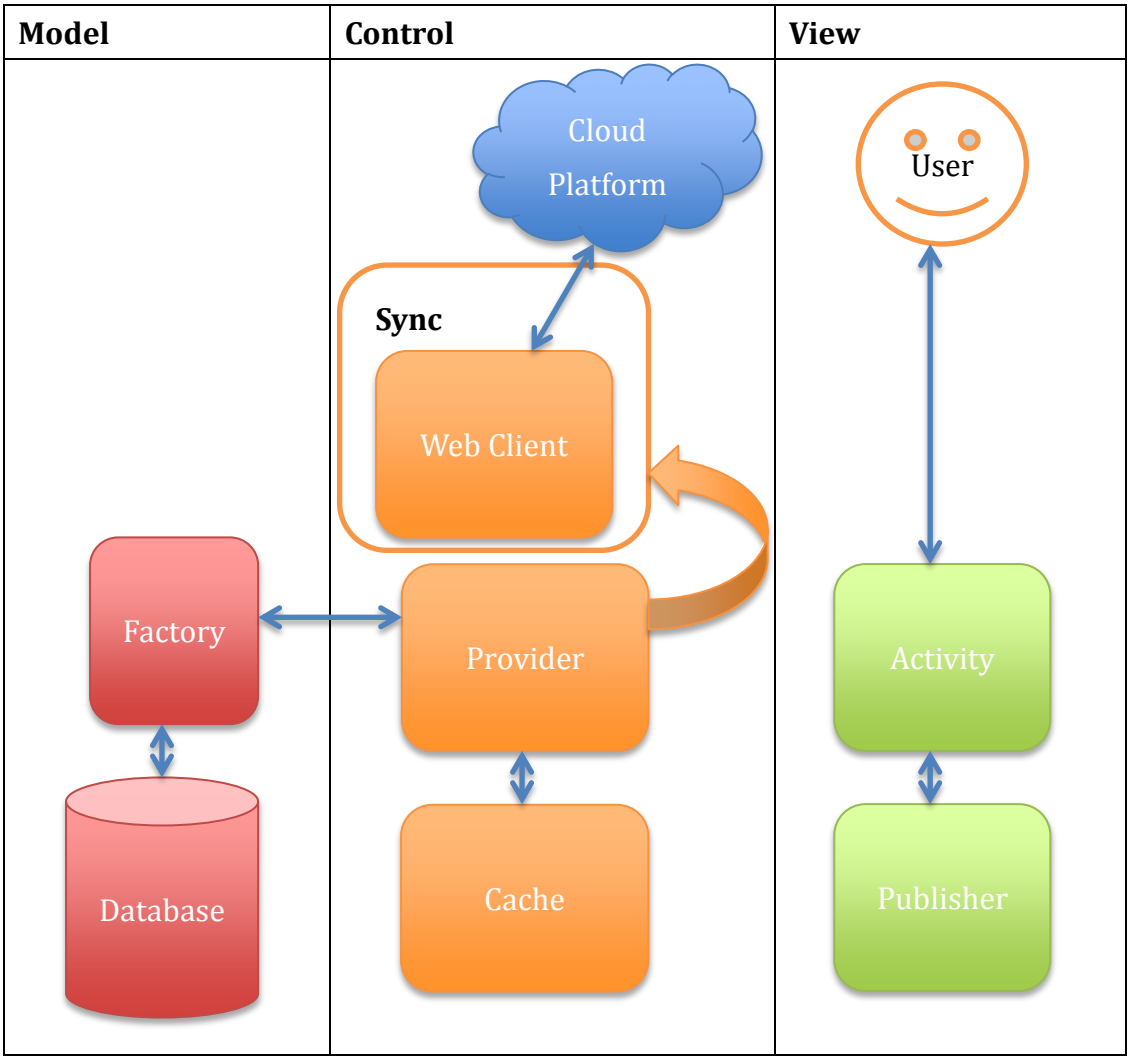
Every time a new object is created or updated, using the observer pattern, the object is shown or updated onto the screen.

3.2.2.2 The Synchronization Manager

The Synchronization Manager is the most important component of the application since it handles asynchronously the requests from the client to the server and the database inserts. When one wants to make a request for some type objects, a new task is created and added to the Synchronization queue that will later process it. The queue of tasks is processed in the ordered they arrived. The Synchronization manager has in its composition a Synchronization Worker, the component that is the worker thread and acts like a background thread, and between the Synchronization manager and Synchronization Worker there is another component called Synchronization publisher. The purpose of this component is to notify the listeners when a

task is finished. The task can be a failure or a success.

Fig. 2 Application Architecture



3.2.2.3 Main Components

Item	Description
Database	Insert, Update, Delete, Retrieve
Factory	Database Access
Web Client	Rest Client to connect to the Server
Provider	Connects all the parts of the Application
Cache	Stores temporary data.
Adapters	Display data onto the screen
Sync Manager	Ensures asynchronous processing of the data.
Publisher	Acts like the Observer pattern.

3.2.2.4 Data Control manager

The data control manager is composed by three components, Data Provider, Data Factory and Data Cache. The data provider is the component of the application which connects the Data Cache component and the Data Factory component.

At the beginning when the application starts, the data provider first makes a request to the server for the all the possible collections. After the collections are received they are stored into a temporary cache called the Data Cache. The Data Cache is the component which stores all the temporary collections and all the objects part of these collections received from the server.

In order to view an object from a collection a request is made from the data provider. If the object exists in the database or in the Data Cache then it is taken from there and shown, otherwise a request to the server is made. After the object is received, the data provider is accessing the Data Cache for saving it into the local cache and, afterwards, through the Data Factory is inserting it into the database. Data Factory is the component which accesses the database. It is capable for insert, delete and update of objects.

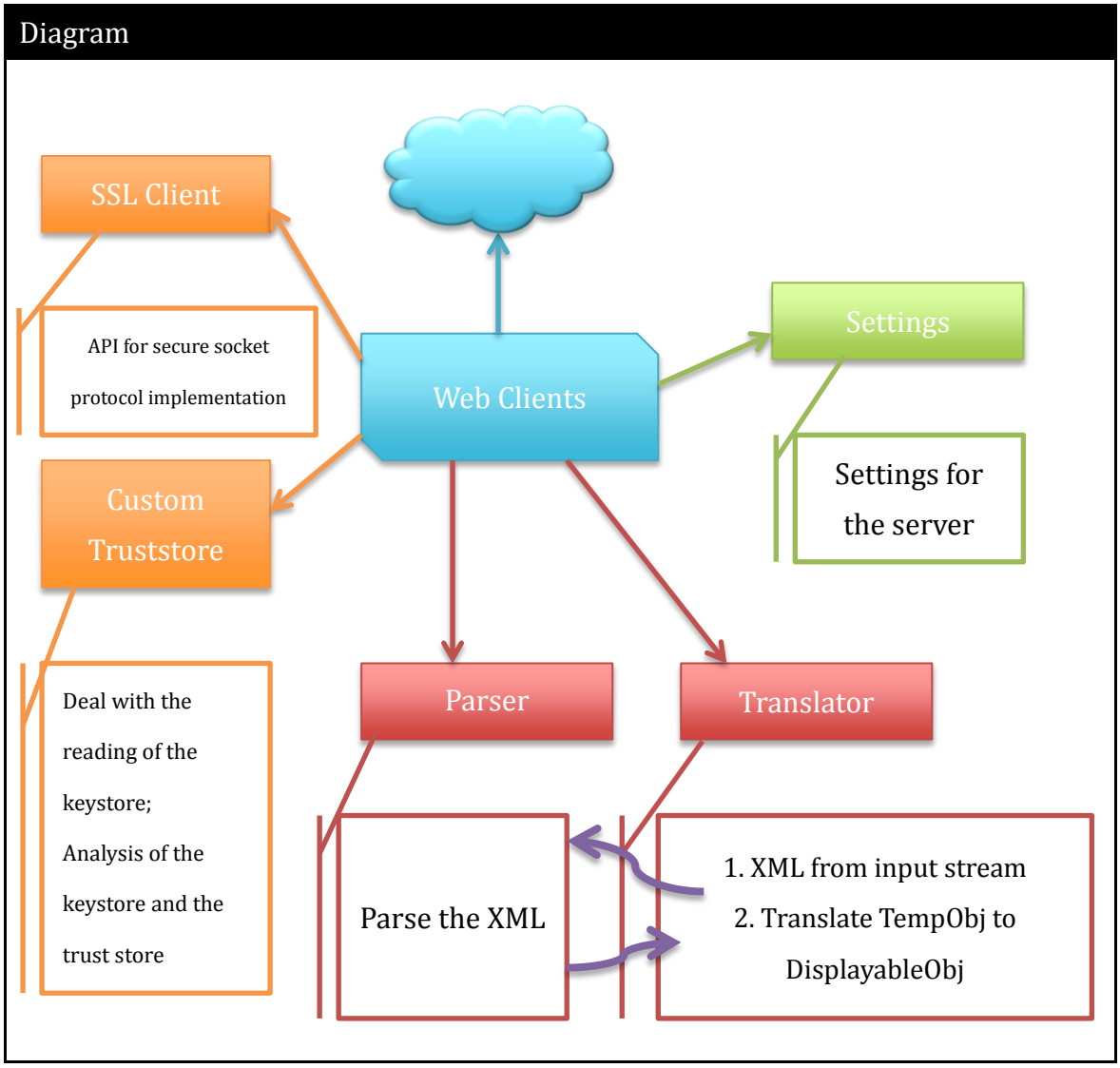
Currently update is not used.

3.2.2.5 Rest Client Structure

The Rest Client is the interface through which the application connects to the Cloud Service. The first step in order to connect to the service is to make a secure connection. The first thing is to initialize the settings required for the connection, like the URL of the cloud service the protocol and other settings needed. This is done by using a Settings manager which, at the starting of the application creates a list of settings. The second step done is to create a secure connection and this is done by using a TrustStore manager which handles the searching and processing of all the truststores from the android device. After the connection is

established, depending on the type of operation, a request to the server is made. The inputstream received from the server is afterwards parsed to some temporary objects.

Fig. 3 Web Client Structure



3.2.2.6 Database structure

The database structure is split in several parts.

The first part is the part of the creation. When the application starts for the first time the database tables are created by using some predefined queries. After the database is created for every type of object a similar row is created and it is based on type of object that needs to be stored. In order

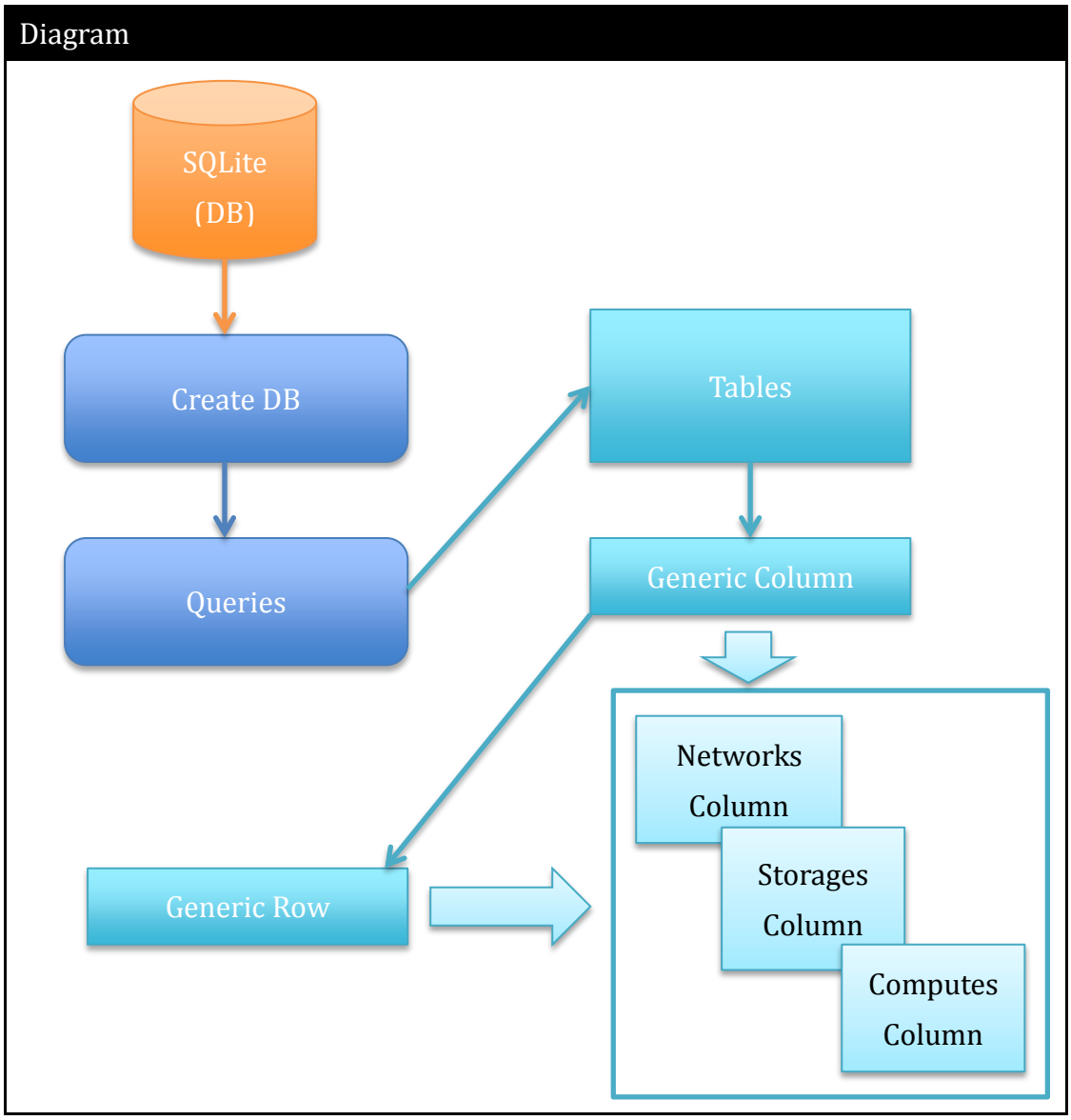
to simplify the structure they all have in common the same component, a generic row that contains all the SQL methods.

The next layer in the database model is the layer where the queries are constructed.

The next layer in the database model is the model where all the dependencies are constructed, for example when one object wants to find all its dependencies from other tables.

On top of the database model is the Data Factory, which is the component that accesses the database model.

Fig 4. Database Structure



3.3 Problems

3.3.1 Credential problem

The communication between our application and the cloud must be done using a certificate. We encountered problems in accessing the cloud because of the certificate. We solved it by using all the TrustStores installed on the android device.

3.3.2 Offline Mode

Since the entire project is based on the mobile device, the smart phone won't be able to connect to the Internet all the time.

Thus a local database was created for the purpose of saving the data received from the cloud service.

When the device is not connected to any internet source, the data will be retrieved from the mobile device.

3.4 Advantages of designed architecture

3.4.1 Independent Modules

"Manage Cloud through Mobile Devices" is designed to have different independent modules handling independent tasks to fulfill the goal.

3.4.2 Easy to debug; Easy to extend

"Manage Cloud through Mobile Devices" is designed and implemented under a proper structure for extensions. Since the base is already built properly through the designed architecture, further applications can be extended, modified, improved and debugged easier.

3.5 Why is it to the cloud

"Manage Cloud through Mobile Devices" is designed to work after Open Cloud Computing Interface (OCCI), which uses RESTful as the communication model. Since OCCI is the communication standard for most of the main platform providers such as Oracle, Cisco and OpenNebula[ii]. We can say that this client-server designed RESTful communicating architecture is totally designed for the cloud environment.

3.6 Limitation

The limitation of the current application is that, "Manage Cloud through

Mobile Devices" is now designed only for accessing OCCI service through RESTful model. This is the specification and scope of the project. Thus, any other possible communication models and interfaces are not support.

3.7 Future works

[Please help filling up]

4 Application Category

Our application works as a twist of Software as a Service and a service to the Platform as a Service. We can categorize it as Mobility as a Service toward a Cloud Platform implementation.

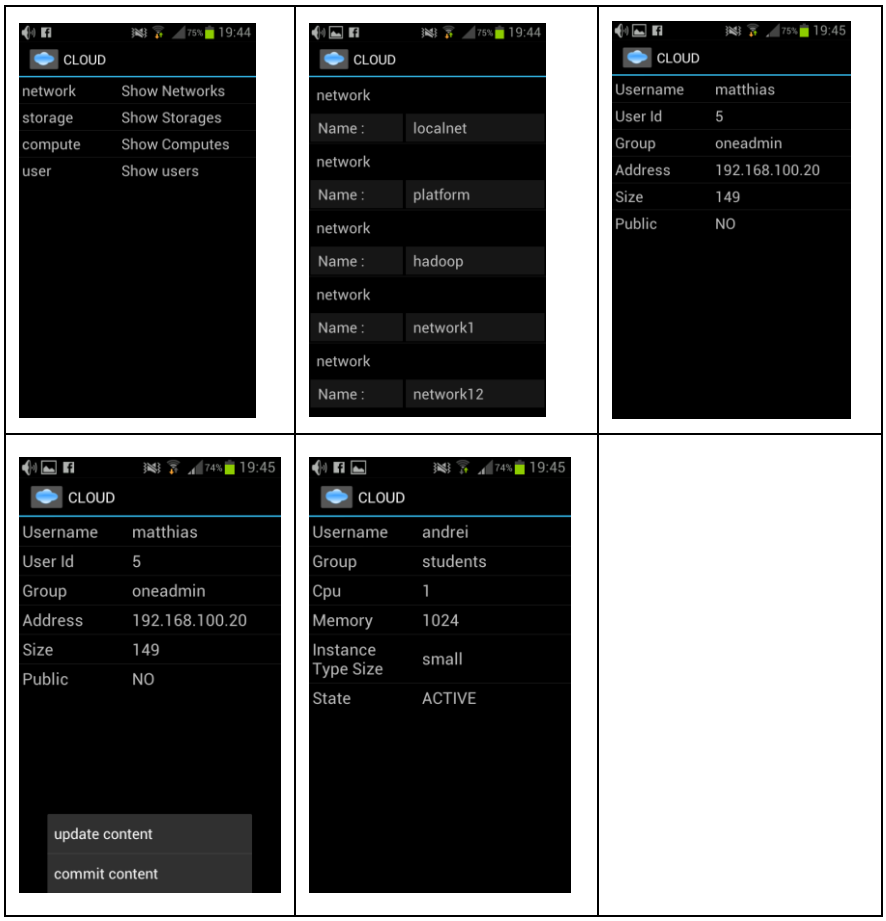
Mobility as a Service, or MaaS, has several good characteristics, described below [iii]:

- Available Everywhere: With access to the 3G Internet, people can manage their server everywhere, using their smartphone.
- Provides Universal Connectivity and Access Controls: This is our target, our aim, and our purpose. The first thing to achieve is connectivity and access of the cloud interface.
- Support for all Android devices: People can get access to these functions on the Android device.
- Ensures Endpoint and Data Security: This is something we aim for. Open Nebula itself offers solutions for security, and we have to work on application security on Android device.
- Simplifies User Experience: This is what we aim for.
- Improves Risk Management and Cost: This is what this project is all about, improving risk management and cost by foreseeing failures on the cloud machines.

To sum up, our project is totally suitable for Mobility as a Service for fitting most of the characteristics of MaaS.

5 Icon for the application

6 Main pictures of the application



[i] DeLone, W.H. and McLean, E.R. 1992. "Information Systems Success: The Quest for the Dependent Variable," *Information Systems Research (3:1)*, pp. 60-95.

[ii] http://en.wikipedia.org/wiki/Open_Cloud_Computing_Interface

[iii] http://wikibin.org/articles/mobility-as-a-service.html