ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

University of Piraeus
MSc Digital Systems
Network-Centric Systems

Mobile Web Services: Design and development of Web Services platform to provide information to mobile devices based on geographical location

Supervisor: Vera Stavroulaki

November 2009, Athens

Author: Kosmas Beros

Contact info
E-Mail: k.beros@gmail.com

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Πανεπιστήμιο Πειραιά
Μεταπτυχιακό Δίπλωμα στα
Δικτυοκεντρικά Συστήματα

Κινητές Υπηρεσίες Παγκοσμίου Ιστού (Mobile Web Services): Σχεδιασμός και ανάπτυξη πλατφόρμας με χρήση Web Services για την παροχή πληροφοριών και υπηρεσιών σε φορητές συσκευές με βάση τη γεωγραφική θέση

Επιβλέπουσα Καθηγήτρια: Βέρα Σταυρουλάκη

Νοέμβριος 2009, Αθήνα

Συγγραφέας: Κοσμάς Μπέρος

Στοιχεία Επικοινωνίας
E-Mail: k.beros@gmail.com

Abstract

This dissertation describes and explains the Location Based Services and all the technologies that derive them and enable developers to provide to users, location aware services that they can use on the go. It documents, along others, the design, the implementation, the testing strategy, and the evaluation of the final product in the perspective of usability, performance, finance and power consumption of the MEWS platform.

MEWS is a distributed platform that provides a variety of services, implemented to be available to users on the move, using any mobile device which is JAVA enabled. The services provided by the platform can help someone get information like the weather, and news. The most advanced services allow the users to create, edit, and search Points of Interests (POI) around. The users can also be informed if their friends are around them and send those messages, find events in progress around their location, and get informed about potential traffic and road hazards on their route. They can also be informed for advertisements concerning products and stores around them. MEWS is implemented with expandability and consistency in mind. The key powers of the platform that distinct it from the rest are *that it is the users* that using a *simple unified environment* can use services that *can be enriched with content and functionality* to meet their expectations in a *secure and convenient way*.

Περίληψη

Η διπλωματική εργασία αυτή περιγράφει κ εξηγεί τις Υπηρεσίες **LBS** και όλες τις τεχνολογίες από πού προέρχονται, και καθιστούν ικανούς τους μηχανικούς λογισμικού να προσφέρουν στους χρήστες, υπηρεσίες που γνωρίζουν την τοποθεσίες αυτών κ μπορούν να χρησιμοποιηθούν εν κινήσει.

Τεκμηριώνει, εκτός των άλλων, την σχεδίαση την υλοποίηση, την στρατηγική δοκιμών, κ την αξιολόγηση του τελικού προϊόντος στο πλαίσιο της χρηστικότητας, απόδοσης, χρεώσεων και κατανάλωσης πόρων.

Το **MEWS** είναι μια κατανεμημένη πλατφόρμα η οποία προσφέρει μια πληθώρα υπηρεσιών, υλοποιημένη να είναι διαθέσιμη σε χρήστες που βρίσκονται εν κινήσει, κάνοντας χρήση οποιοδήποτε κινητού η φορητής συσκευής που διαθέτει την τεχνολογία της **JAVA**.

Οι υπηρεσίες που προσφέρονται από την πλατφόρμα μπορούν να προσφέρουν σε κάποιον πληροφορίες όπως ο καιρός και τα νέα. Οι πιο εξελιγμένες υπηρεσίες επιτρέπουν στον χρήστη να δημιουργήσουν, επεξεργαστούν και αναζητήσουν σημεία ενδιαφέροντος **(POIs)** γύρω του. Επίσης, ο χρήστης μπορεί να ξέρει αν φίλοι του είναι κοντά του και να τους αφήσει μήνυμα, να βρει εκδηλώσεις που λαμβάνουν χώρα στην περιοχή του, και να ενημερωθεί για τυχών προβλήματα και κινδύνους που μπορεί να βρίσκονται στην διαδρομή του καθώς μετακινείται. Τέλος μπορεί να έχει ενημέρωση για διαφημίσεις σχετικά με προϊόντα από καταστήματα που βρίσκονται κοντά του. Το **MEWS** έχει κατασκευαστεί με γνώμονα την επεκτασιμότητα κ την συνεκτικότητα. Τα «δυνατά χαρτιά» του που το κάνει να ξεχωρίζει από τα υπόλοιπα ανάλογα προϊόντα είναι, ότι *είναι ο χρήστης αυτός* που *χρησιμοποιώντας ένα απλό ενιαίο περιβάλλον*, έχει στα χέρια του υπηρεσίες που μπορούν να *εμπλουτιστούν με περιεχόμενο κ λειτουργικότητα* με σκοπό να καλύψει τις ανάγκες του με ένα *ασφαλές κ βολικό τρόπο*.

Table of Contents

List of Figures

List of Tables

Acronyms

POI	Point Of Interest
GPS	Global Positioning System
Wi-FI	A trademark for the certification of products that meet certain standards for transmitting data over wireless networks.
3G	Third generation of mobile phone standards and technology
LBS	Location Based Services
MPCs	Mobile Positioning Centers
GMLCs	Gateway Mobile Location Centers
TCS	Telecommunications Systems

SMS	Short Message Service
PDA	Personal Digital Assistant
CEPT	Conference of European Posts and Telegraphs
GSM	Global System for Mobile communications
ISDN	Integrated Services Digital Network
ETSI	European Telecommunication Standards Institute
MSC	Mobile services Switching Center
WSDL	Web Services Description Language
SOAP	Simple Object Access Protocol
HTTP	HyperText Transfer Protocol
CORBA	Common Object Request Broker Architecture
RMI	Remote Method Invocation
XML	Extensible Markup Language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JVM	Java Virtual Machine
GUI	Graphical User Interface
IDE	Integrated Development Environment
UML	Universal Markup Language
RSS	Really Simple Syndication
JSP	Java Server Pages
URL	Uniform Resource Locator
PNG	Portable Network Graphics
JDBC	Java Database Connectivity
MIDP	Mobile Information Device Profile
J2ME	Java 2 Micro Edition
JAX-WS	Web Services Interoperability Technology
WSIT	Web Services Interoperability Technology
JAXB	Java API (Application Programming Interface) for XML (Extensible Markup Language) Binding

Scope

The scope of this document is to provide information to the reader in order to understand, firstly, the background technologies that the software relays, and secondly, the architecture of the system itself.
The technical documentation follows a template inspired by the waterfall model the software was designed on. That way the examiners can understand the way every step of the process was performed, from the requirements gathering process to the testing and evaluation of the final system. Finally the design and especially the implementation sections will give enough technical knowledge so that every complex part of the code can be clearly understood even for debugging and maintenance of the software from other developers.

1. Introduction

People move around a lot more than they used to. The way of living is fast and the need for information while being not around a computer is common. Sharing information between people can be used to provide a richer, more up to date, and more honest context. Since anyone nowadays carries almost all the time a mobile phone, which is actually a micro computer, it is possible to utilize this device for this purpose. The context that people share can be anything that someone would like to see himself. This info along with their location, stored in a usable way and presented in a mobile environment, enables a variety of services that can be provided by a platform called MEWS.

By combining the technologies like Web Services, Java, GPS, Wi-Fi, 3G, etc it is possible to have an incredible amount of usable info in our palm. The Technologies are good enough to be used and reach the level of usability and flexibility that can follow the modern standards, the ignorance of technology of the user, and the necessity of expandability and fast growth of the relative technologies.

The MEWS platform is a common environment that hosts various Location Based Services (LBS) that users would find useful while being on the move and they would like to find places of interest (POI), find events around them, be alerted for dangers, read the news, the weather, find friends and send them messages, etc.

Location Based Services provided in a consistent manner would save time and money in comparison to other similar services from various vendors and different environments.

In the following sections a brief description of all the related technologies that have their part on the implementation of the software can be found. Each technology plays its role, and provides a layer from the bottom which is the hardware, to the top which is the human, the user of the platform. Later, both functional and non-functional requirements for the platform are described to let the reader know what it is about and how it is expected to perform. After that, the important contribution of the design of the system is detailed separated by web services. In addition, the implementation section gives details about how various modules and sub systems were implemented along with the problems encountered and their solutions during the process of coding. The testing and evaluation section follows to allow the reader know what testing strategy was followed and what tests were performed on the system, to make sure the it behaves as normally as possible and meets all the requirements given at the

begging of the lifecycle of the software. In the evaluation section, an investigation in terms of finance and consumption is performed to evaluate how feasible the final system is under real world usage. Finally, a conclusion and suggestions for future work can be found.

2. Background of Location Based Services (LBS)

Location Based Services is not new. It is not mature though also. Vendors and carriers have shown interest in LBS since 1997. It is only then that they realized that "People have the need to know the location of something or someone relative to their location". The idea was born. Profit could be gained by utilizing this technology and create applications and services.

They introduced network nodes, now-standardized elements in a wireless network, known as mobile positioning centers (MPCs) in IS-41 networks and gateway mobile location centers (GMLCs) in GSM networks. Early providers of these network location components were Ericsson, Nokia, SignalSoft (now Openwave), and Telecommunications Systems (TCS). Others to follow in their footsteps were typical wireless infrastructure vendors such as Alcatel, Hughes, Intrado, Lucent, Motorola, Nortel, Siemens, and SchlumbergerSema.

The classic problem around communication technologies was the "communication".[1][3]

No standards, lack of specifications and concrete APIs made the architecture almost unusable for the carriers. There was a plethora of problems encountered with the Phase I approach; integration and interoperability with existing systems dominated the many outstanding issues. Each time a carrier/operator elected to offer a new application to subscribers on behalf of the application developer, the carrier/operator was forced to enact rigorous, expensive, and custom implementation and integration efforts for that LBS application.[3]

In 2001, the LBS industry experienced a reactionary shift in how carriers/operators chose to implement their systems. Several carriers/operators around the world, who had experienced the integration pains of introducing one-off applications into their networks, recognized the need for a centralized system. The solution to the problem was an architecture founded on the principles of centralized Web Services and one-stop shopping environments for developers. In a Web services model, application developers build Location Based Applications by using a suite of enabling technologies and APIs. That drove them to reconsider and redesign the model. The implementation was the phase II model.[3]

The Phase II model ensures that the carrier/operator has complete control over how any application developer interfaces to the core network, and it dramatically reduces implementation costs.[3]

LBS are not only about the carriers. The hardware, along with the software installed on the servers and the clients that are part of this LBS system should go along with the wave. A brief history and milestones of the LBS and the relative technologies can be found bellow.

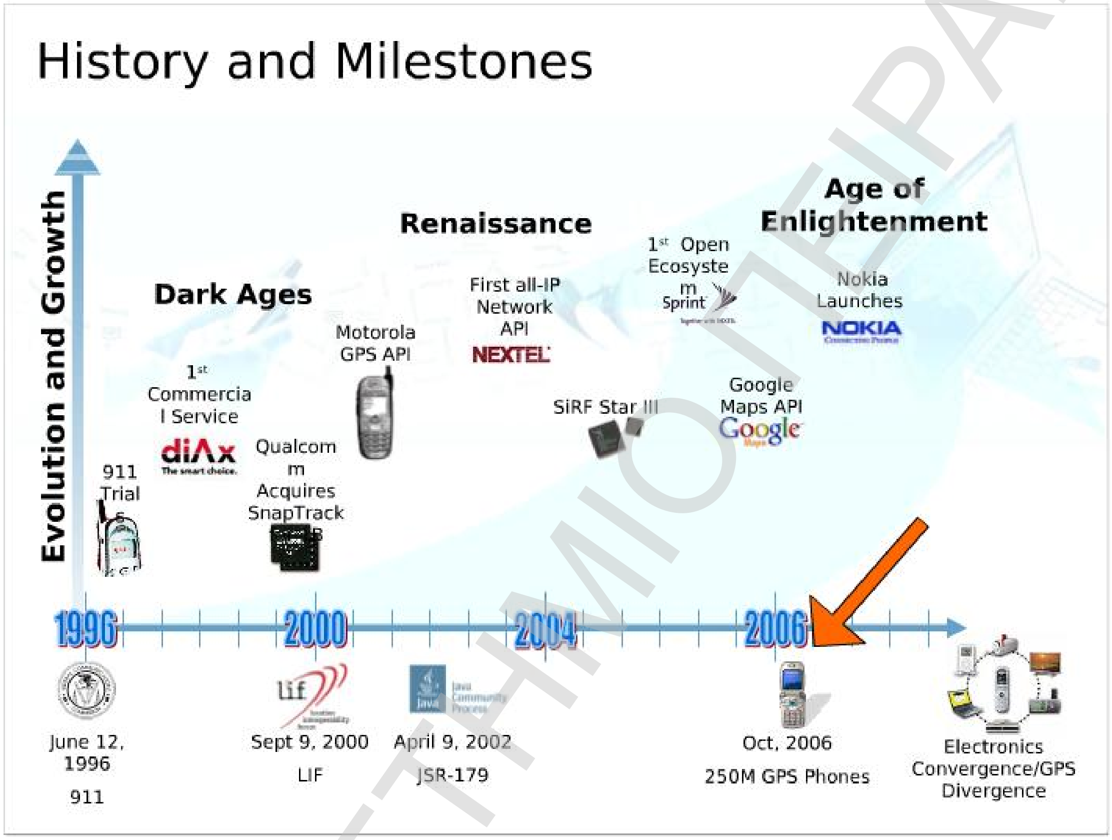


Figure 1: LBS History and Milestones

Since 1996 the companies like Motorola, Nokia and SUN, along with the chipsets manufacturers managed to create a SW/HW combination that would provide and utilize the LBS. Today almost 1 out of 3 mobile devices have a GPS chipset and the software platform to provide any kind of LBS to the user. Nokia predicts that by 2012 all mobile devices will be equipped with a GPS chipset. The luxury will be need, and users will use the LBS the same way they use SMS texting.[10]

3 Mobiles & Location Bases Services (LBS)

Mobiles are tightly connected with Location Based Services. The whole concept of location is the fact the people are on the go. So, since having a PC all the time in your pocket is not convenient, and nowadays a mobile or any handheld device is basically a powerful computer of the 90s, mobile phones and PDAs are the best tool to enjoy the LBS. On this section we will examine the technologies and the history of mobile phones, as well as the techniques and technologies used for location a device.

3.1 Mobiles

During the early 1980s, analog cellular telephone systems were experiencing rapid growth in Europe, particularly in Scandinavia and the UK, but also in France and Germany. Each country developed its own system for their specific needs, which was incompatible with everyone else's in equipment and operation. This was an undesirable situation, because not only was the mobile equipment limited to operation within national boundaries, but there was also a very limited market for each type of equipment, so economies of scale and the subsequent savings could not be realized. [5]

The Europeans realized this early on, and in 1982 the Conference of European Posts and Telegraphs (CEPT) formed a study group called the "Group Special Mobile" (GSM) to study and develop a pan-European public land mobile system. The proposed system had to meet certain criteria: [5]

- **Good subjective speech quality**
- **Low terminal and service cost**
- **Support for international roaming**
- **Ability to support handheld terminals**
- **Support for range of new services and facilities**
- **Spectral efficiency**
- **ISDN compatibility**

In 1989, GSM responsibility was transferred to the European Telecommunication Standards Institute (ETSI), and phase I of the GSM specifications were published

in 1990. Commercial service was started in mid-1991, and by 1993 there were 36 GSM networks in 22 countries. Although standardized in Europe, GSM is not only a European standard. Over 200 GSM are operational in 110 countries around the world. In the beginning of 1994, there were 1.3 million subscribers worldwide, which had grown to more than 55 million by October 1997. [5]

A GSM network is composed of several functional entities, whose functions and interfaces are specified. Figure 2 shows the layout of a generic GSM network. The GSM network can be divided into three broad parts. The Mobile Station is carried by the subscriber. The Base Station Subsystem controls the radio link with the Mobile Station. The Network Subsystem, the main part of which is the Mobile services Switching Center (MSC), performs the switching of calls between the mobile users, and between mobile and fixed network users. The MSC is also responsible for the mobility management operations. Not shown in the figure is the Operations and Maintenance Center, which oversees the proper operation and setup of the network. The Mobile Station and the Base Station Subsystem communicate across the Um interface, also known as the "air" interface or "radio link". The Base Station Subsystem communicates with the Mobile services Switching Center across the A interface. [5]

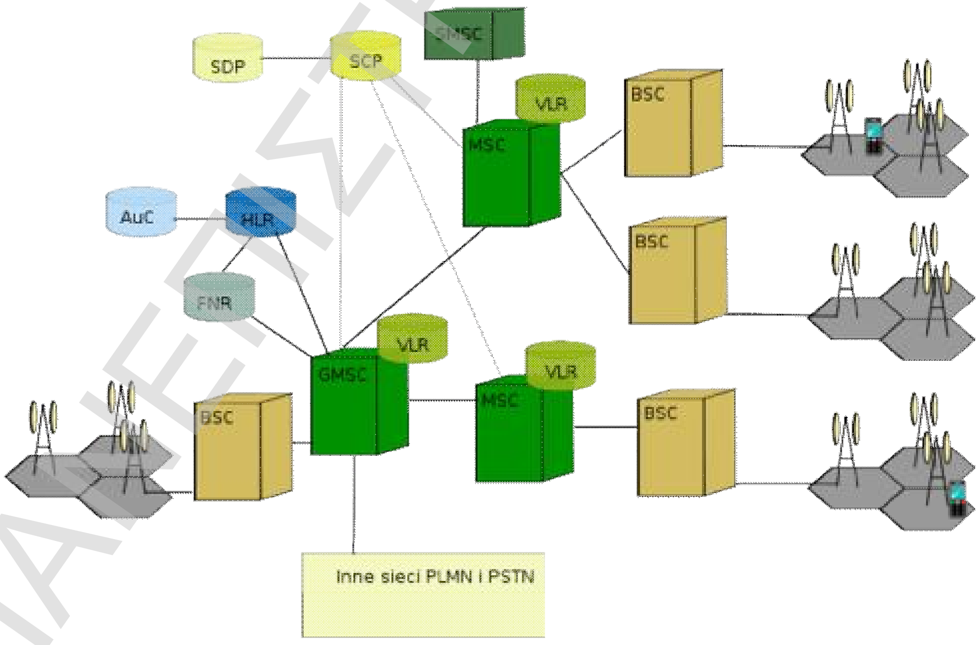


Figure 2: Mobile Network Architecture

GSM is a complex standard but this is necessary in order to overcome the restrictions and weaknesses of the radio environment, while providing quality and integration possibilities. On top of this mature and well-tested platform, the new technologies can rely and provide new ways of communication and context delivery to the subscribers.

3.2 Locating Technologies & Techniques

In order to determine the position of the device there are various techniques that manage it, and they are separated in 3 categories: *Network-based*, *handset-based*, and *hybrid*. All of them can determine the position, but with different costs.

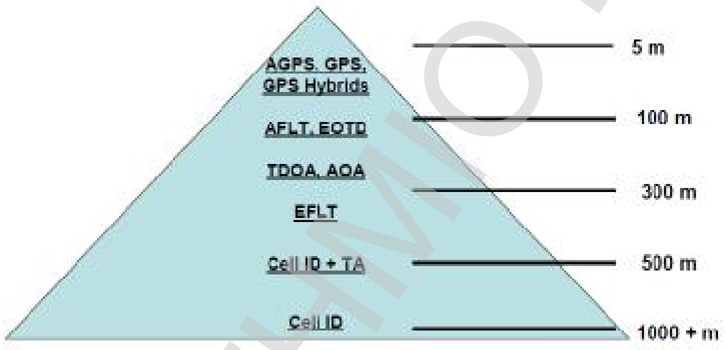


Figure 3: Locating Technologies and Their Accuracy

Figure 3 show that each technology attains a different accuracy. The GPS related technologies are obviously the most accurate since it can even achieve 5 m accuracy. But that comes with a cost, energy efficiency. Next, the most important technologies as well as their pros and cons are explained. Finally a comparison and evaluation for these technologies will show and summarize their characteristics. [11][12][13][14]

Cell-ID: Is the most basic and the first technology for locating the device. The determination is only done by using the service cell which can vary since the cell covers a big area of about 1000 or even 3000 meters radius.

Cell-ID + Time Advance: Timing advance only gives a distance. Cell-ID only gives a cell. Combing the two makes it possible to determine what distance from the base station in the cell the user is located. If you know the position of the

antennas, you can calculate the position of the user and use Timing Advance value to calculate the distance to get an arc for the user's position.

Time Difference of Arrival (TDOA), Angle of Arrival (AOA), - (Triangulation): **In all techniques the locations is triangulated by measuring the time at which signals from three sources arrive. Similar to GPS but instead of satellites, antennas are used. TDOA uses a lot of signaling in order to determine the position, so although it was extensively used at the beginning for commercial LBS services, it was abandoned. The accuracy of the first two techniques is about 300 m.**

EOTD is a method in which the phone measures the difference in the time of arrival signals from different towers. Essentially, the phone triangulates its position using signals from the towers, the reports the position back to the network.

EOTD is one of several technologies that carriers can deploy to meet E-911 requirements. Unlike TDOA technology, the phone actively participates in the location process. Therefore, EOTD only works with phones that specifically include EOTD technology. The accuracy of these techniques is about 100 m.

EFLT (Enhanced Forward Link Trilateration , ALFT (Advanced Forward Link Trilateration):

To determine location, the phone takes measurements of signals from nearby cellular base stations (towers) and reports the time/distance readings back to the network, which are then used to triangulate an approximate location of the handset. In general, at least three surrounding base stations are required to get an optimal position fix.

Both AFLT and EFLT are CDMA specific technologies that uses the signaling characteristics of a CDMA to make a positioning determination. They act as a fall-back method for CDMA devices if A-GPS is not successful and the network was requested to make a location determination.

A-GPS: **Assisted GPS is an improvement on GPS, primarily for mobile devices with network connectivity. Since mobile devices are connected to a wireless network on a known wireless base station, the location of the base station can be used to aid the GPS calculation. Assisted GPS takes the known location of the base station and provides synthetic or seed data to the mobile device's GPS chip to improve the speed of GPS signal acquisition. This allows for a faster location fix**

time and makes possible GPS use in some indoor environments. Assisted GPS does require device chipset support and network communication.
In the following table all the technologies are gathered along with their requirements, pros and cons.

Technique	Req. HW	Req. SW	Precision (m)	Energy Cost	Data Traffic
Cell-ID	No	No	~1000-3000	Low	No
Cell-ID+TA	No	No	~500	Low	No
TDOA,AOA	No	No	50-200	Low	Yes
EOTD	No	Yes	50-200	Low	Yes High
AFLT	No	Yes	~50-200	Low	Yes
EFLT	No	No	250-350	Low	Yes
GPS	Yes	Yes	5-30	High	None
A-GPS	Yes	Yes	5-30	High	Yes Low

Table 1: Locating Technologies Comparison

As it was demonstrated all methods have their ups and downs. In order to overcome the disadvantages and problems hybrid methods are used. The most common combinations are:
AFLT/AGPS
EOTD/AGPS
Cell-ID/AGPS
Depending on the availability and the precision requested the methods are used accordingly. The choice generally depends on:
- Wireless network technology
- Operating Environment
- Applications (requirements).
- Cost

4 Web Services

Consider a scenario where someone wants to find a specific store. He would like to buy a camera he saw a friend of his has. What he would do is to go to the web site and find the address in the contact info page. Since he does not know it, he would google it. He uses a specific way (language) to search and get the desirable results. After he finds it he visit the website, and he finds the contact info. He writes down the address, gets his car and drives there to buy the camera.
Web services work in a similar manner.

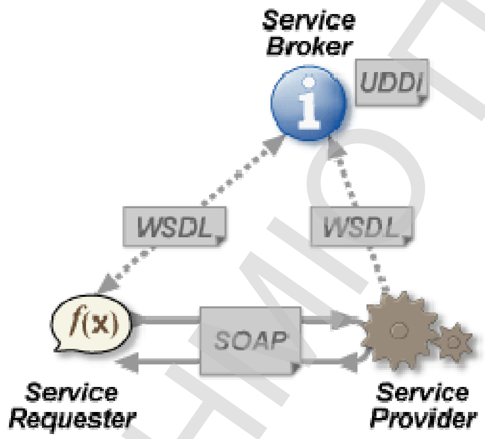


Figure 4: Web Service's Lifecycle

The service requester needs a specific result and can provide specific arguments. Asks the Service Broker using a WSDL file where it can find the Service Provider. The service Broker also responds with a WSDL file, giving enough information of where and how to invoke the web service. The Service Requester then directly sends a request to the Service Provider using a SOAP messages over HTTP and the Service provider, after it processes the request responds also using SOAP with the results. The Broker is not necessary if the service requester knows where to find the Service provider. The provider itself can give the rest of the info by itself (how to invoke it).[20]

So what is a web service? "A software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages,

typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards". [16]
Web Services take web applications to the next level. By using web services, an application can publish its function or message to the rest of the world.
The success of web services is two-fold. [18][17]

- Reusable application-components: There is things applications need very often. So why make these over and over again? Web services can offer application components like: currency conversion, weather reports, or even language translation as services.

- Connect existing software: Web services can help to solve the interoperability problem by giving different applications a way to link their data. With web services you can exchange data between different applications and different platforms. [18][17]

There were similar technologies for the same reasons the web services were introduced, like CORBA, RMI, etc. All these technologies though were not platform independent and language independent, neither.
On the other hand web services have overhead and lack of versatility.
The overhead is caused because the transportation of data is not in binary code but in high level HTTP messages. So in order to win in portability, they lose in efficiency. [21]
Currently, Web Services are not very versatile, since they only allow for some very basic forms of service invocation. CORBA, for example, offers programmers a lot of supporting services (such as persistency, notifications, lifecycle management, transactions, etc.). Fortunately, there is a lot of emerging web services specifications (including WSRF) that are helping to make web services more and more versatile. [21]
Web Services in the business world, in the most simplistic fashion, provides a mechanism of communication between two remote systems, connected through the network of the Web Services. For example, in case of a merger or an acquisition, companies don't have to invest large sums of money developing software to bring the systems of the different companies together. By extending the business applications as Web Services, the information systems of different companies can be linked. These business systems then can be accessed by using simple SOAP messages over the normal HTTP Web protocol. For example, a manufacturing company requires some raw materials to be supplied whenever

the material in stock reaches the threshold levels. These levels can be constantly monitored by the business system of a trusted supplier, and promptly replenished, without having to wait for a supervisor to notice it and generate a work order. [19] [21]

SOAP and WSDL are essential parts of the Web Services Architecture: [21]

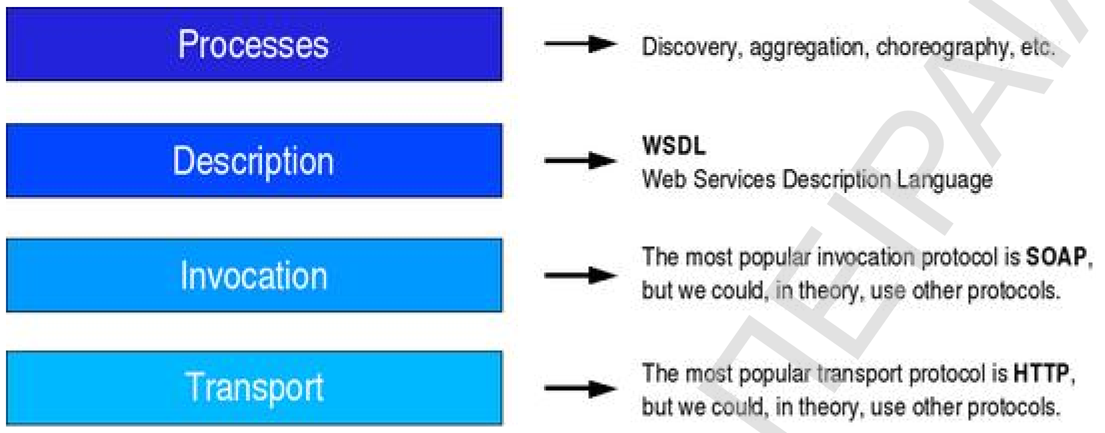

Processes
Discovery, aggregation, choreography, etc.
Description
WSDL
Web Services Description Language
Invocation
The most popular invocation protocol is SOAP, but we could, in theory, use other protocols.
Transport
The most popular transport protocol is HTTP, but we could, in theory, use other protocols.

Figure 5: The Web Service Architecture

- Service Processes: This part of the architecture generally involves more than one Web service. For example, discovery belongs in this part of the architecture, since it allows us to locate one particular service from among a collection of Web services.

- Service description: One of the most interesting features of Web Services is that they are self-describing. This means that, once you've located a Web Service, you can ask it to 'describe itself' and tell you what operations it supports and how to invoke it. This is handled by the Web Services Description Language (WSDL).

- Service invocation: Invoking a Web Service involves passing messages between the client and the server. SOAP (Simple Object Access Protocol) specifies how we should format requests to the server, and how the server should format its responses.

- Transport: Finally, all these messages must be transmitted somehow between the server and the client. The protocol of choice for this part of the architecture is HTTP (HyperText Transfer Protocol), the same protocol used to access conventional web pages on the Internet. Again, in theory we could be able to use other protocols, but HTTP is currently the most used one.[21]

Most of the Web Services Architecture is specified and standardized by the World Wide Web Consortium, the same organization responsible for XML, HTML, CSS, etc.

4.1 Combining the State of The Art Technologies

All the technologies described above are the latest efforts of researchers, universities, and companies. By combining, the location awareness of devices, the connectivity of the mobile networks, and the easily and interoperable applications using web services we can have rich context, efficiently and on time on the go. Locations Based Services can be even more accessible using the web services technologies and their latest specifications. The only thing missing is only ideas, for any kind of use and application. Sky is the limit.

4.2 Examples From Around the World

There are already many examples of location based services that utilize all the above technologies. Some of them are successful, some of them are not. Some worth mentioning are: [22][23][24] [25]

MyMap - A complete suite that provides Routing instructions, POI finder, and Friend Finder.

Fleet and Workforce Management - A service that allows companies of various types and sizes to effectively track and manage mobile personnel and vehicles, using either standard or specialized mobile devices. Office managers may assign tasks, send open orders and dispatch text messages, as well as assist workers in the field by providing them with POI information, maps and navigation instructions.

Buddy Cloud - Is an application for your mobile phone that lets your friends know what you are doing and where you are.

Carticipate - Is an experiment in social transportation, the first and only rideshare application on a location aware mobile platform.
Users can save money on gas while helping the environment by 'carticipating' with friends, family, groups, or co-workers.
They can coordinate driving plans by indicating where you are going, when, and post your ride.

Geocaching - Is a high-tech treasure hunting game played throughout the world by adventure seekers equipped with GPS devices. The basic idea is to locate hidden containers, called geocaches, outdoors and then share your experiences online. Geocaching is enjoyed by people from all age groups, with a strong sense of community and support for the environment.

GyPSii - Is a geo-location and social networking platform and service for mobile, web and set-top box devices. GyPSii is a UGC (User generated-content), social networking, search and locationbased suite of integrated mobile and web solutions for users to share, view and upload pictures, video, text and POI (points of interest) with a geo-tag (location data) place and track each other in their select communities. The main features of this application are its User Generated Co tent (UGC), Real Time Friend Finder, Mobile Proximity Search of UGC and Maps & Navigation that is the heart of the application.

Senda - Is a provider of travel-time management information services dedicated to companies in a variety of industries including media, transport, warless and the web. Chronomove is an always-on travel-time management solution that lets commuters: choose their Estimated Time of Arrival (ETA) and traffic events, and communicate their ETA to persons awaiting their arrival.

UbiSafe - Is a location based service for personal security, contact locating and emergency needs. UbiSafe is targeted to the consumers' market. It can be used for monitoring elder or invalid persons in need of attendance, for personal safety (at night, in walking risk zones, etc.) or for children security.

5 Motivation

The idea of Location Based Services is not something new. As previously shown, there are many application and services that provide information and allow users to connect while keeping them informed about the location of other people and places. So what makes MEWS better? What is that special features that could make something like that work in the real world?
The MEWS platform unites all the services that someone needs on the go. It provides a UI that through that people can ask for information, or find friends like other commercial services but in a more concrete and consistent way, and with the addition of the power of the user contribution. The services the MEWS platform provides have nothing to be jealous of the competition. In contrary, most of the services that MEWS provides offer even more features and make the user experience even more enjoyable.

The MEWS Platform

6 Requirements

6.1 Introduction

The project was suggested and chosen by me. We as students were free to choose exactly what to implement. The mobile Location based services subject was always interesting for me.
The requirements were given by both me and my supervisor, while keeping in mind, what the user would like to have in his palm while on the go.
The basic requirements only came a week after the project approval. The rest of them were made during the design. Some of them were my suggestions that again asked for the approval of the supervisor. The project started with a list of "initial" requirements that are described in section 6.2.1. The most advanced requirements are included in the "later" requirements section. This actually happened because we were not sure what we could ask from a system like that and what limitations we have in the domain of security, locating and mapping.
The management of the project was to try to achieve the initial requirements as soon as possible, and then anything possible from the later ones that would be done during the process of implementation. They are divided in functional and not functional requirements. The first category will be sub-divided to initial and later requirements.

6.2 Functional Requirements

In the following section the tasks that the final system should be able to perform for the end-user are described.

6.2.1 Initial Requirements

1. Central Login System - Users will login once giving them access to all the services.

2. News Service - Users can ask for news using a third party service.

3. Weather Service - Users can get the weather for their location, or any other City, for the current date or any date available to the MEWS system.

4. POI Service - Users should be able to add, edit, search and review Points of Interest divided in categories and sub-categories, and specific radiuses.

5. Events Service - Users should be able to add, edit and search for events by category and specific radiuses.

6. Traffic Alerts Service - Users can manually search, add or edit alerts for dangerous incidents on the road network. In addition, users can setup the system so they will be notified for dangers that are on their route.

6.2.2 Later Requirements

1. More secure communication - The security of the communication should be achieved using tokens for each message transferred and WSIT specifications for web services communications between the Mobile device and the MEWS central server.

2. Charging system - The users should be charged per usage. For example if someone added a new POI there should be no charge. If someone asks for the weather there should be a charge according to the MEWS platform policy. Each service and each request can have different prices.

3. Top Up system - Users should be able to top up their accounts using top up cards. These cards contain 10-digits phrases that accordingly deposit money to their accounts.

4. BuddyLoc service - A service that users can use in order to find friends around him/her (that have been added to their friends list), change statuses, send messages and get the exact distance form their friends.

5. User status updates - Users can update their statuses like facebook so other people can see what they think or want to share.

6. Leave messages - Users can leave messages to their friends that they can either instantly read or when they are back online.

7. Friend invitation via E-mail - Users should be able to send e-mail invitations to friends by simply submitting an e-mail account through their mobile phones.

8. Points on maps for all services - All services should provide to the users map with their current location and the point the service refers to. So someone can see on the map where he/she is while at the same time can see the location of the danger, the POI, the event or the friend they are interested in.

9. Advertisement service - A service that stores can advertise their products and firms. The service should be targeted to people that are around the stores and the advertisements should be equally delivered.

6.3 Non-functional Requirements

Here are described the requirements that answer questions like "how safe? How fast? How much? They describe measurements in particular attributes of the system.

Interface and end users – The system focuses to any kind of people. They can be IT experts, housekeepers, or children. So the GUI should be as simple and

functional as possible. The usability is a very important factor for the way the system was implemented.

Hardware and performance considerations – The system should work with as many devices as possible. Because it is implemented in java, it shouldn't need a different version for each operating system. Performance issues can be raised because of heavy data traffic and 2D maps. Also location techniques should be considered for different operating systems and different kinds of devices.

Security considerations – The nature of the system demands quit strong security. The communication between the handheld devices and the MEWS server should use secure technologies to protect the transaction of sensitive data.

Error handling – The interface will be used on the go, making it easier for users to choose or enter mistaken data. The system should protect the users from as much errors as possible. Also, useful and explanatory feedback to the user should exist.

Deployment – The user should be able to install and use the handheld client software without the present of an IT expert. A jar file would be desirable.

6.4 Feasibility Study

Software Engineering hides many risks. "Not everything imaginable is feasible". It requires a lot of analysis and planning to understand the limitations and probable difficulties that may occur. Bad choices and wrong decisions might not look so significant for the process of implementation but in the feature they can be so significant that can cause halt of the project and require backtracking of months coding.
A Gantt chart was created to estimate the time for individual parts of the project. The process of design took quite a lot because of research and experimentation on the various issues like locating techniques and map rendering. The creation of the project would follow the Gantt chart only if everything would be feasible and not extreme time periods would be wasted for problems that may occur during

the implementation. Some potential problems were tracked during the planning at the early stages and are listed in 6.4.2 section.

6.4.1 Financial Feasibility

Money is not an issue for this project, because it is developed for academic purposes.

6.4.2 Potential Problems

Device & Software compatibility for locating techniques - The Software might not be able to use all or some of the underling GPS and network related locating technologies.

Map Rendering - There might not be an easy way to provide maps in J2ME Midlets.

Limited memory/CPU power - The client will be running in limited resources Java Virtual Machines hosted sometimes in cheap mobile phones with limited memory and CPU. That can be a serious problem for the application, making it even impossible to run in full potential or causing it to crash.

6.4.3 Risk Analysis

Risk analysis is part of the project planning to examine and evaluate the possible risks that exist in a project of that size. The risks can hide behind the management of the project, or can be technical problems that during the implementation can occur. A plan of risks and the ways to avoid them or to solve them in various different stages of the progress, can be prepared and followed by the manager and the team to save time and money. In that project the manager and the programmer is only one person and that makes the planning easier.

Usually management risks are become real problems when technical problems are found. Serious technical problems may cause delay of a day or even a week to find a solution and continue to the next step. One of the duties of the manager is to arrange the schedule of the project so that it can handle a reasonable time of delay.

That strategy was followed by the manager of this project too. The design and planning was scheduled to finish as early as possible, as also happened with the main implementation tasks.

6.4.4 High Level Risks and Solutions

Minor risks can only cause small delays of some hours that are not so important for the progress of the project. But what happens if major problems cause huge delay during the implementation process? High risks should be tracked and assessed as soon as possible.

Resources & Performance

Although mobile and handheld devices are now days quit powerful, there are still devices that cannot perform acceptable when running heavy and demanding applications. The MEWS platform by design would run various thread, possibly 2 or 3, while maintaining communication with the server for all of them and at the same time the location of the device would need to be obtained using the GPS chipset (embedded or via Bluetooth). In addition large amounts of data arrive from the server storing them in data structures for later use. Since the developer has not experience with mobile systems, a prototype to test the various components of the resources was developed. The prototype tests in parallel the location mechanism, the data transfer, and also runs 4 threads that also request data from the server.

In case the client can not perform as desired, various alternatives are used. For example the J2ME specifications allow choosing the criteria of the energy and resource used. So instead of GPS, the locating could be performed using network related locating techniques like TDOA or AOA. The software would be less demanding in case the J2ME could not handle many threads at once. For example the feature of automatic alerts or automatic message notifications could be taken out. In case of inability to store the desired amount of data in the appropriate data structures, or being unable to perform calculations with data from these structures, various others, not so convenient data structures can be used. Also

some software engineering techniques were used to gain performance while doing the calculations. The last resort would be to crop data retrieved from the server. So the platform would provide less information than the amount that was planned.

Locating Techniques & Hardware / Software Compatibility

Although the platform is all software, this software must somehow maintain a communication link with the hardware underneath. The part that the platform must use is the GPS chipset. Since MEWS is built completely on J2ME, the JVM is responsible for this communication. The link is abstract to the developer. But unfortunately each device can have a slightly different distribution of the JVM or even completely altered. So by having, a range of different devices, with different Operation Systems and different JVM distributions, raises the problem of incompatibility. The problem could be fatal for the application since without knowing the location of the device, would make the application unusable. In case the problem occurs the developer should force the application to use network related techniques if possible. Otherwise the project would be unfeasible. After online research, on the design stage, it was found that most of the devices would be compatible. The problem would be the distribution of the windows mobile devices. The JVM used is a cropped version, that doesn't contain the required packages to perform a communication with the underling GPS unit. So the decision would be to make use of the GPS unit where possible, and use the network related technologies where the JVM distribution is not capable to use it.

6.5 Planning

After research and brainstorming concerning the technologies and the platform to be created a planning process took place. This is the process where the engineer has to make some kind of timetable with the important tasks of the lifecycle of the creation of the software to be made. A standard and universally accepted way is a Gantt chart. The following figure demonstrates the tasks and the periods that were planned and were followed in order to complete the platform along with its documentation.

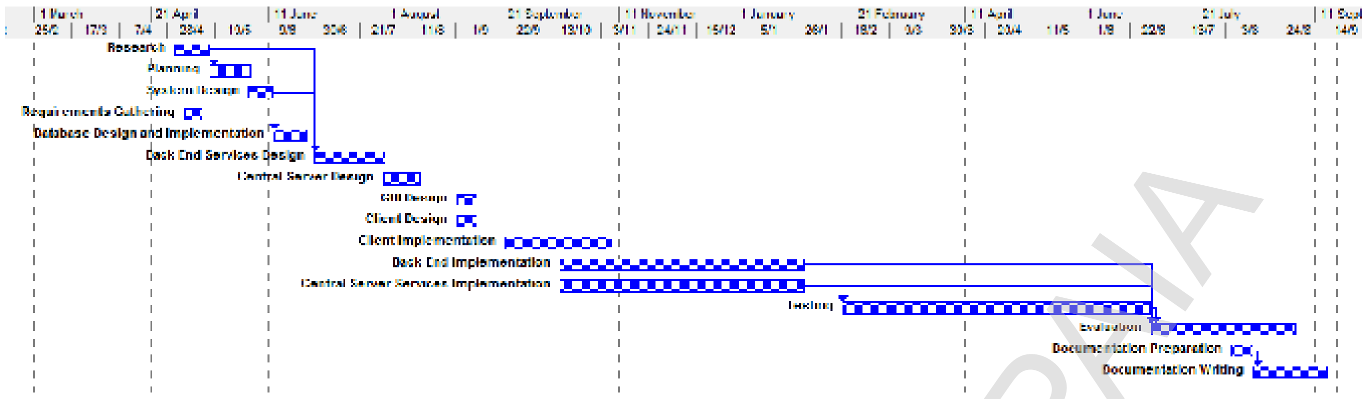


Figure 6: The Gantt Chart of the MEWS Platform

7 Design

7.1 Introduction

The purpose of this part is to combine the requirements and the so far planning of the project, and create a new plan of coding and integration that will produce the requested application. Although it looks useless sometimes, it is very important and going directly to the phase of implementation is a big mistake. It is the foundation of a project especially if its size can be considered as big.
The design tries to outline in detail the shape of the coding and connect the various modules of the system. It is the stage where the project starts to take form, a form that will be very similar to the final. It is really important to evaluate the output of the design in every stage and backtrack to create the system that should look like the final. Since the system is modular by nature, the design as well as the implementation sections is covered separated as web services.

7.2 System Model

After the gathering of the initial requirements and the planning of the system, a choice of the software engineering model should be done. The most widely used model is the water fall or sequential. The first one was the choice for the particular system. Since the requirements were already known, both initial and later, and it is the most commonly used model, the choice was clear.

7.3 System Architecture

The system architecture describes the collaboration and modular links of the system. It is very similar as a building construction works. The base of the building and how strong the floors are connected is really important to withstand any danger. Predictions for earthquakes for example must be made for a building before the construction begins. Similarly, software should withstand any

unpredicted danger or abnormality of the surrounding systems that it is linked to, including the user. Reusability and maintenance should be considered from that stage too.

The construction of the system followed a top – down architecture. The required services are designed and implemented one by one, while keeping the big picture in mind. Each component separately fulfills its task and is attached to the platform, combining it with the rest of the services according to the design. The skeleton of the system was implemented first, containing some core modules from all the sub systems and almost all layers.

Since the system is a group of web services and their consumers, each one with specific inputs and outputs, it is easy to view each one as a module and work from small to big or the opposite where required. Each module was implemented and tested autonomously for producing the required results and performing acceptable.

The following figure describes the architecture of the system, containing all the components and services, as well as the way they interact.

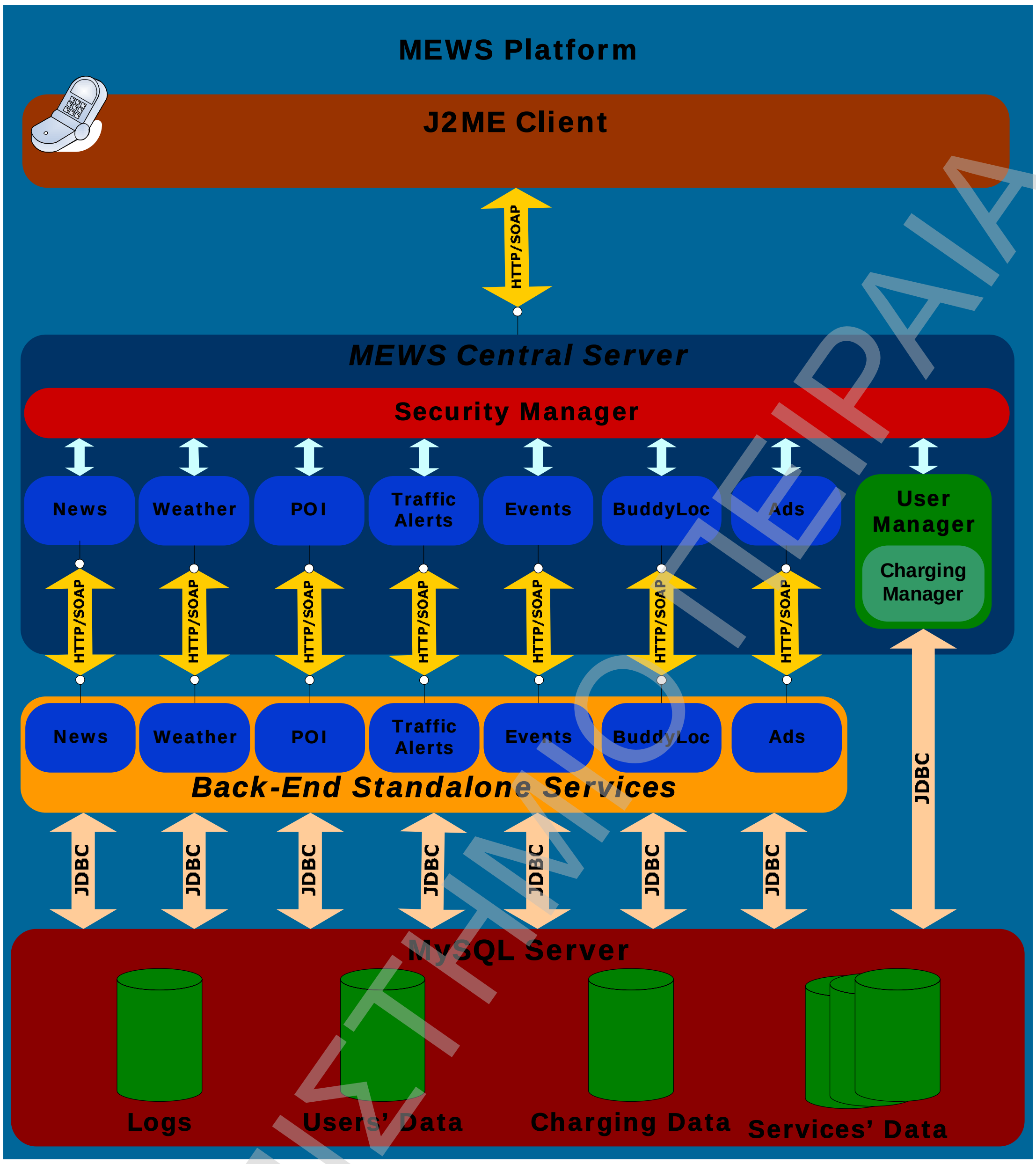


Figure 7: MEWS System Architecture

By viewing the architecture from bottom to top, the first layer is the storage layer. MySQL server 5.1 was used to store the permanent data of the platform. It consists of one database containing more or less 15 tables that contain data like logs, user's credentials and info, actions and their charges, and finally the content generated from the users while using the various services.

By moving up in the architecture the standalone services can be found. These services contain the actual business logic to gather, generate and forward the content to and from the upper layers, as well as the opposite direction, the storage layer. The connection to the database layer is implemented using the JDBC MySQL connector 5.0.7. Each service is independent from each other so each one of them must start and maintain a new connection with the MySQL

server. The MySQL server user has all privileges and the restriction of the actions of the user is implemented using the User Management and Security Manager. Both the standalone services and the central server services are directly communicating with the database layer.
The central server's services are almost 1 to 1 with the standalone services. Their role is to receive the requests from the top level, the client, use the security, and user management modules, and if applicable forward the requests to the lower layers. The central server services don't contain any login relative to the content of the services but only relative to the authentication and charging subsystems. Each one of them is paired with a consumer that is used to forward the request and after the corresponding standalone service has gathered and manipulated the data, the consumer passes the results to the attached service that will be consumed by the mobile client. This of course works the other ways around too, when the user performs and action that requires storing content without the need to return back any data, on the same manner. The communication between the central server consumers and standalone services is achieved using the Metro stack (part of GlassFish Application Server) consisting of JAX-WS, JAXB, and WSIT which enable the creation and deployment of secure, reliable, transactional, interoperable Web services and clients. As previous mentioned part of the central server are the User Management and the Security Manager. Both components interact with each other while being integrated and with constant interaction with all the services. They are responsible to make sure that the request is not coming from a random unauthorized source, by keeping sessions with the help of unique tokens assigned to each session with a user. While at the same time using the WSIT implementation the data traveling through cables or air is encrypted.
The top layer has a different nature from the rest. That's because it is a GUI client build to work on most of the mobile devices that exist and are JAVA enabled. It is a J2ME MIDP 2.0 application that provides forms with rich content to the user, who by interacting with it, triggers consumers that communicate with the services located on the central server.

7.3.1 Considerations

While designing a system, it is the right time to consider the aspects of it. Aspects like extensibility, reusability, robustness, and fault tolerance are some of them

that have been taken into account while drawing the components and designing the way they interact.
The nature of web services gives a big advantage by providing a result of reusable and modular system. By designing the components in a way to be reused where possible can save a lot of time and money. In addition modularity, allows each component to be maintained and tested separately, making it easier for the project development to work. The client modules are not web services, but a different kind of application since it is a J2ME application. J2ME follows different style of programming and need more effort to separate the presentation from the business logic.
The system, especially of this kind, must be able to allow easily other components and features to be added later. That is clearly a design issue. The components must provide a clear and unambiguous input/output as well as APIs where possible to allow other components to connect with them. By predicting what could be added and how would dramatically help to make it possible.
Fault tolerance and robustness are also aspect that can be considered from the design stages. By providing backup elements and predicting the faults that can be caused, while keeping core components intact from other components' failures the system can withstand serious faults while using it. Again the modularity helps a lot.

7.4 Graphical User Interface

One major component of the system is the J2ME Graphical User Interface (GUI). The J2ME platform doesn't give you complete freedom on the design of the graphics so a lot of tries should take place in order to design a final prototype GUI.
The idea was to separate the services in categories. Each category would be a button on the main screen, where the user would choose one and see more buttons in the same manner as the categories. But this time the buttons would be the services that can be used. There will always be a back option on the bottom left side of the screen and the options on the bottom right.

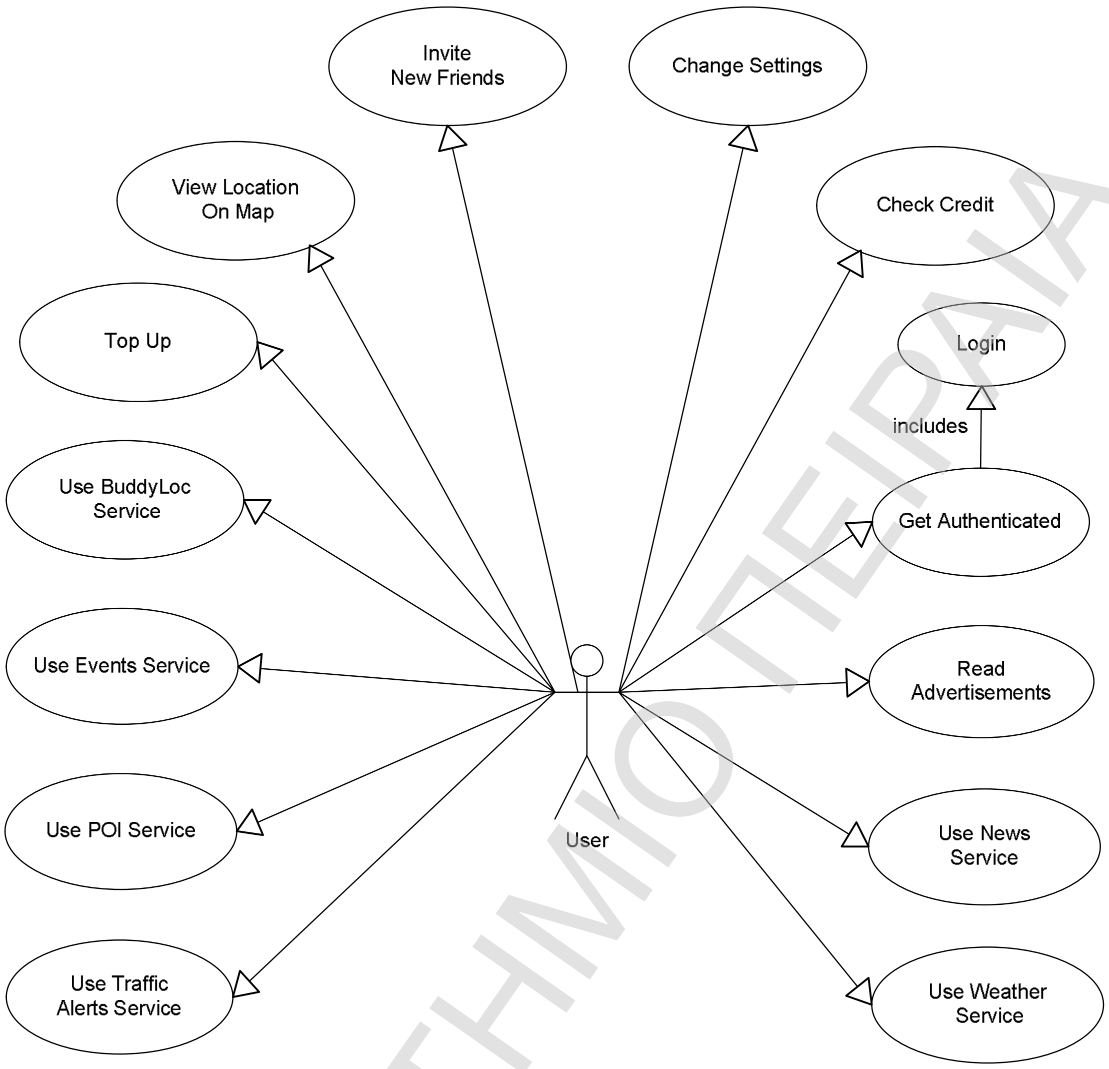


Figure 8: User High Level Use Case Diagram

The Use Case diagram above describes the actions the user is allowed to perform from a high level perspective. In the services design section each one in a lower level will be examined.

Other than using the services, after logging in, the user can check his credit, top up using a top-up card, send invitation e-mails to friends to use the application, change settings, and view his current location on the map.

7.4.1 GUI Design

The following figure demonstrates the navigation map of the GUI available to the user's handheld devices.

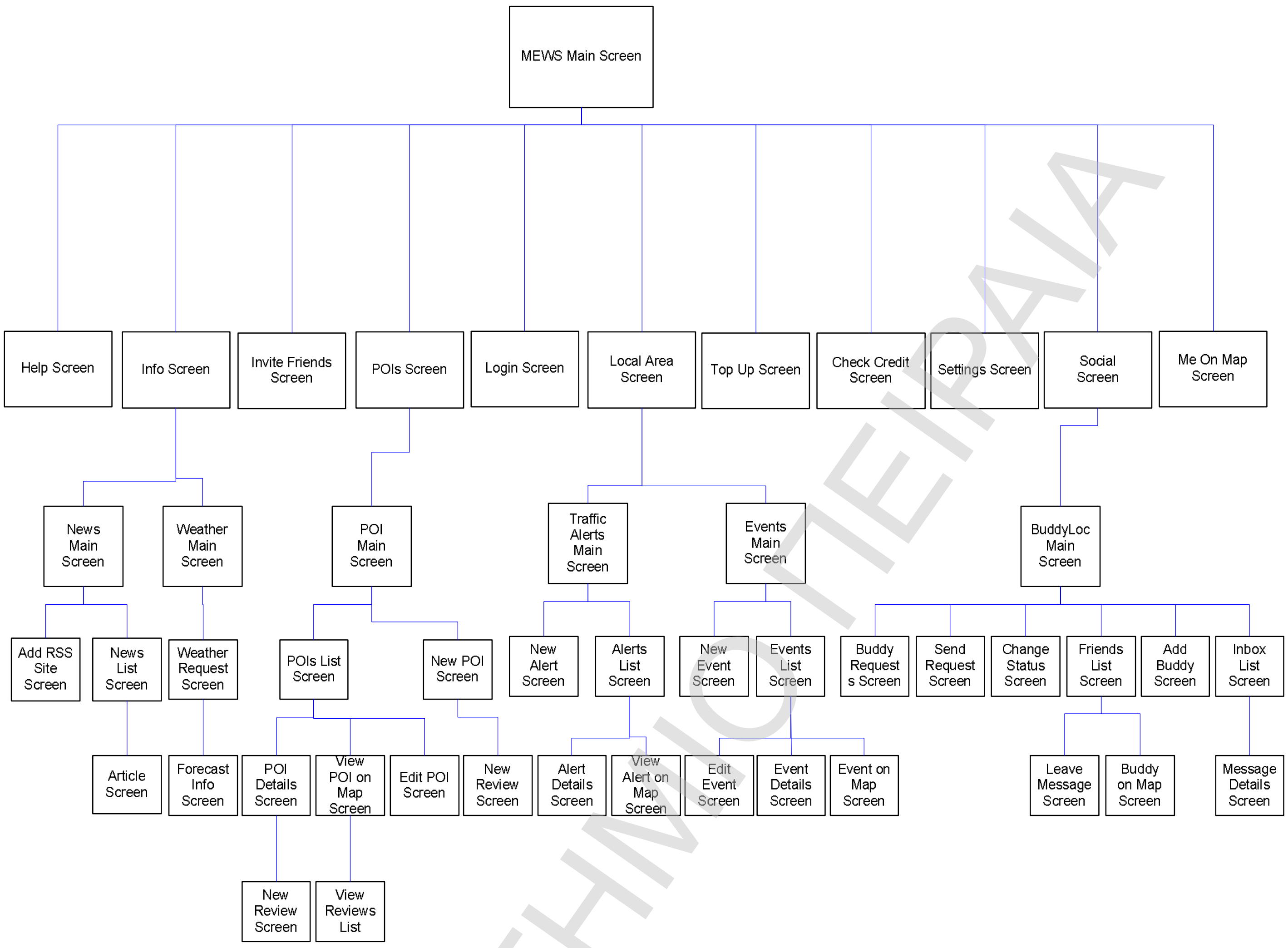


Figure 9: MEWS Client Navigation Map Screens

The mobile client application is the only part of the platform that provides a GUI since it is the mean for the user to use the MEWS services.

The idea of buttons and categories as groups of services was designed on a prototype and tested for speed and convenience. The result was satisfactory so almost all the screen were design using the GUI editor of the IDE to produce the result faster while keeping control of the appearance while doing it.

The following figure is from the prototype GUI, and demonstrates the first screen the user sees, which is the main navigation screen.

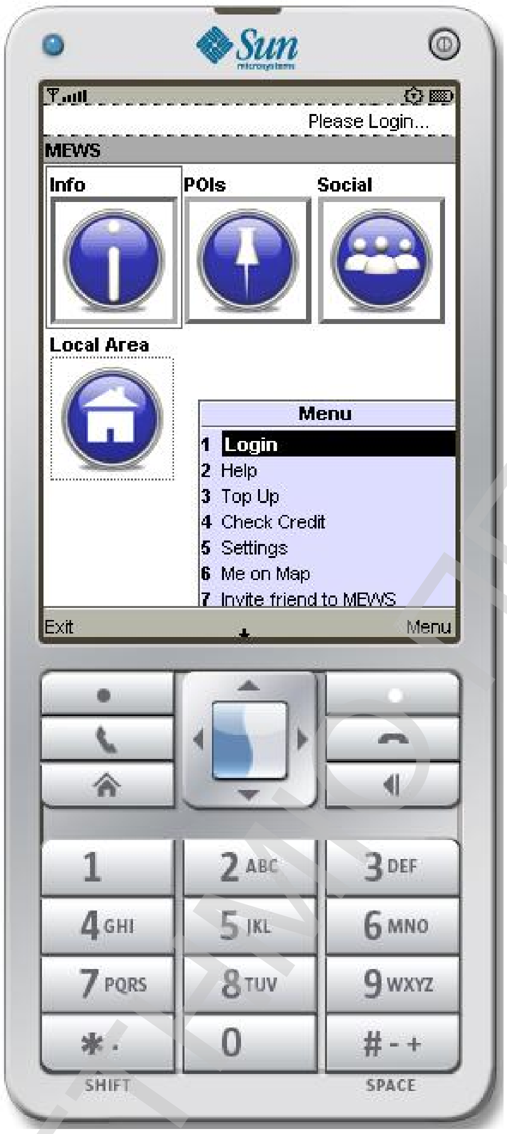


Figure 10: MEWS Client Main Screen

The GUI template follows a similar design for all screens. The buttons have icons and text, the left button navigates back or exits the application if the user is in the main screen. The right button will display a menu with the available commands. In the main screen case the commands are the ones described above along with the action that opens the selected category.

7.5 Database Design

As it is demonstrated in Figure 7: MEWS System Architecture), the system is using a central database for storing the permanent data. The running client uses

temporary data structures that are transformed and used by the web services' consumers to send the data to the server and store the information to a relational database. The following figure demonstrates the schema of the database used by the MEWS platform.

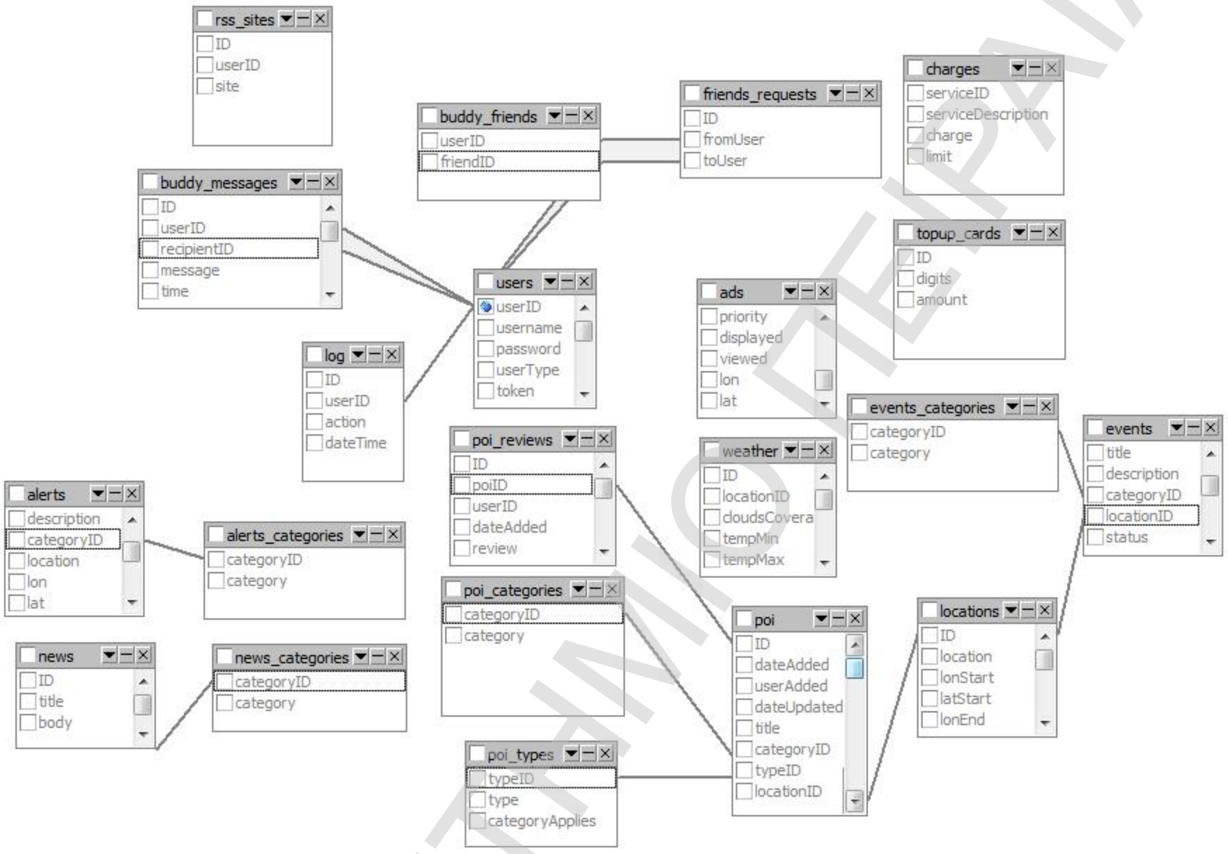


Figure 11: MEWS Database Schema

The dataset is not so complex since the data tables are not connected with many neighbour tables. This happens because each set of tables is used by a specific service, and most of services have no relation with the rest. In addition, the database contains tables that are used to store charges rates, logs and top up card numbers. The database is in 3rd normal form and is easily maintainable with self descriptive column names.[2]

7.6 Services as Modules Design

7.6.1 ManageUsers

The first service that was designed and was also used as the implementation test for the prototype system was the service to manage the users. It is a special and important service since it plays many roles and is used constantly.

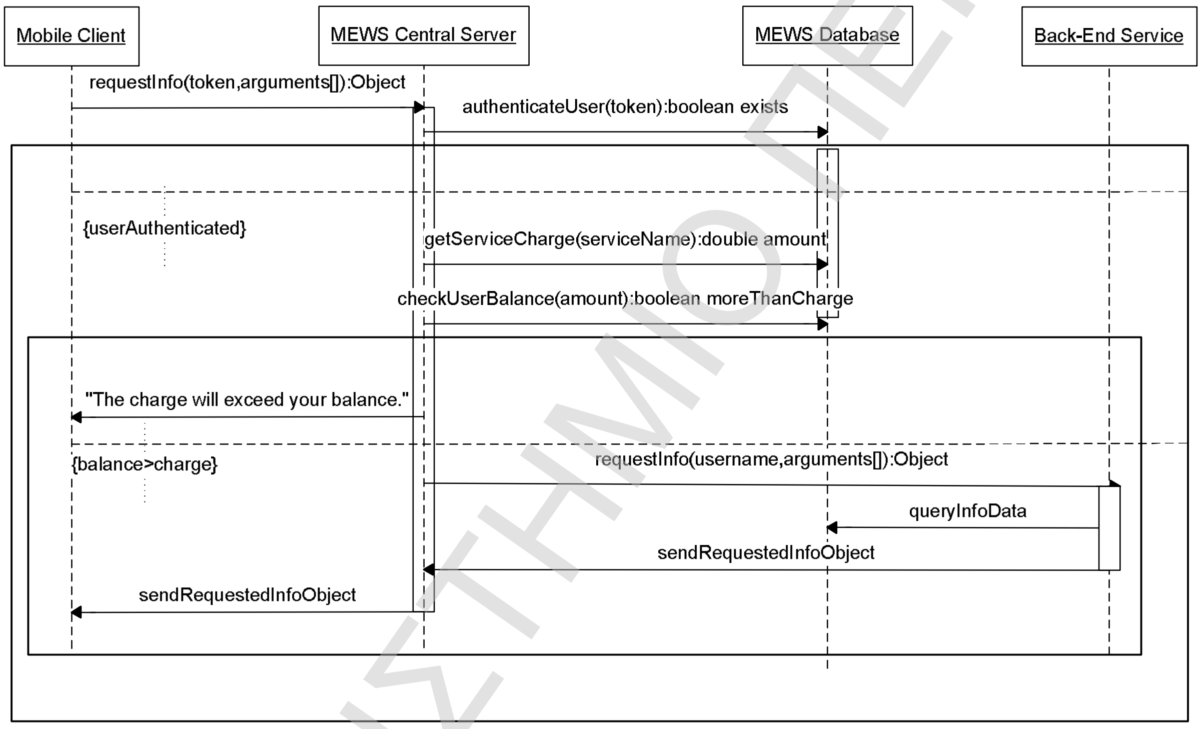
The service provides the following methods:

login:String	***: The login method.***
topup:Boolean	***: Tops up the user account.***
checkCredit:String	***: Checks the credit on the user account and returns the amount.***
sendFriendRequest:String	***:Sends an invitation to another registered user to add him to the buddy list of the user asked for it.***
answerFriendRequest:String	***: Replies to the request sent by another registered user.***
inviteNewFriend:String	***: Sends an invitation via e-mail to an unregistered user to use the application.***
getFriendsRequest:String[]	***: Returns the list of requests from other registered users.***

The login method is the first security measure. It uses the username and password arguments to authenticate the user and gives him/her a random token. This token is used for any request from that point and on from the mobile device to the server. The server recognises which user is using this token. The token is also serialised using the native web services security specifications, so there is no danger of sniffing the token. No action at all is possible without the user being logged in. All the rest are normal request and response methods, each one serving its purpose as whole functionality.

Finally, the *manageUsers* service is internally providing "cash-desk" functionality. Each request by the users, after it is validated that the user is authorised and connected, is matched by an entity in the "charges" table and the corresponding charge is passed as response. This amount is charged to the user account if necessary (some actions like giving a review for a restaurant are not

charged). The reason that this method is not provided from the web service to the outside directly is for security purposes. Since the design of the architecture made possible the communication of all the requests with this internal module of the central MEWS server, allowed to the developers to hide it and make sure it not exposed to the outside world.



Mobile Client
MEWS Central Server
MEWS Database
Back-End Service
requestInfo(token,arguments[]):Object
authenticateUser(token):boolean exists
{userAuthenticated}
getServiceCharge(serviceName):double amount
checkUserBalance(amount):boolean moreThanCharge
"The charge will exceed your balance."
{balance>charge}
requestInfo(username,arguments[]):Object
queryInfoData
sendRequestedInfoObject
sendRequestedInfoObject

Figure 12: MEWS General Request/Response Sequence Diagram

The above figure describes using the UML Sequence notation the procedure of a service request from the mobile client until the response from the server according to the MEWS security and charging policy.

The client sends a request to the central server, lets say getWeather*(token,arguments[]).* The message contains along the arguments necessary for the response from the server, the unique token in order to maintain the session that is authorised for this specific user. The central server will check the authentication of the user in the database (*authenticateUser*), if the user is authenticated the server checks the charge that applies to the specific request

(***getServiceCharge***). Next, the server compares the charge with the user balance retrieved from the DB (***checkUserBalance***). If the balance is less than the charge then the server responds to the client with the appropriate message. If it is more than the charge, then the request is forwarded to the appropriate back-end service along with the username where necessary. Finally, the back-end responds with the requested object to the central server, and it is forwarded by it to the client.

The policy is followed for all the requests than need access to the back-end services. Requests like "***checkCredit***", "***topUp***" "***SendFriendRequest***", etc. that only involve the central server have simpler policy. It actually skips the part of the charging, so only the authorisation of the user is checked.

7.6.2 News

The News service is the only service that uses a third party one. It uses an RSS library that retrieves the data from any site that supports RSS and forwards the response to the client. Other than that it behaves exactly like the rest of the services and obeys to the same policies.

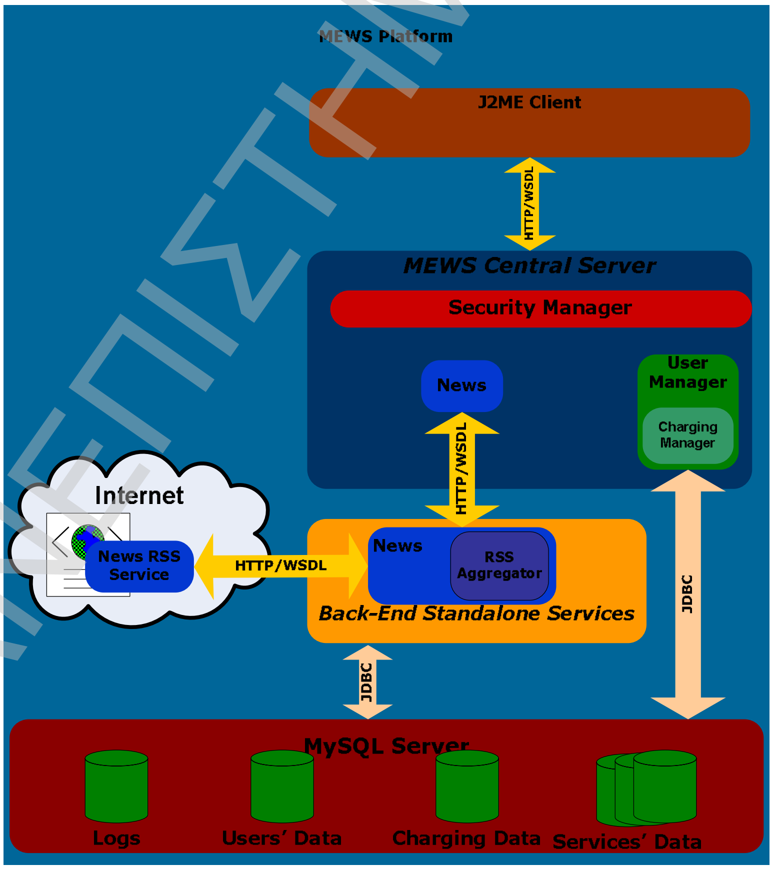


Figure 13: News Modules Diagram

The request is sent by the client, the same way that figure 12 explained. After the user is authorized and charged in the central server the request is forwarded to the back-end news service, which uses the RSS feed reader to consume the RSS web site service. The response is sent back as XML document. The news service then responds to the central server with the results, and finally is forwarded to the client.

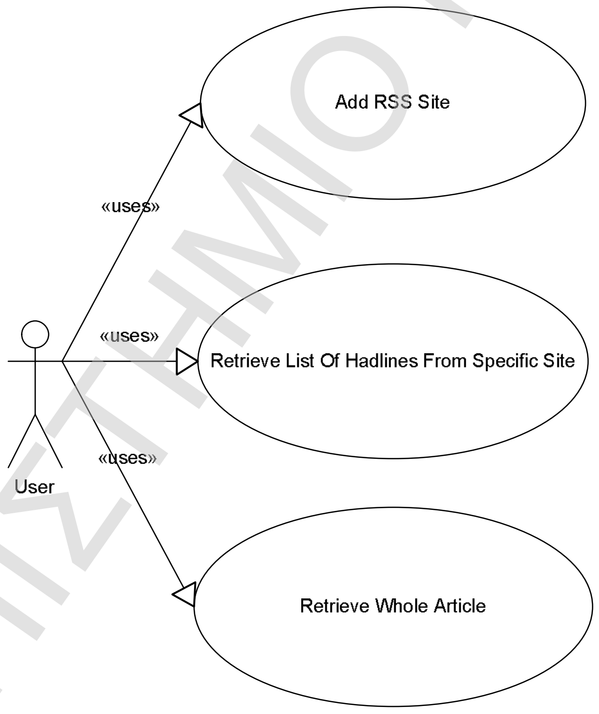
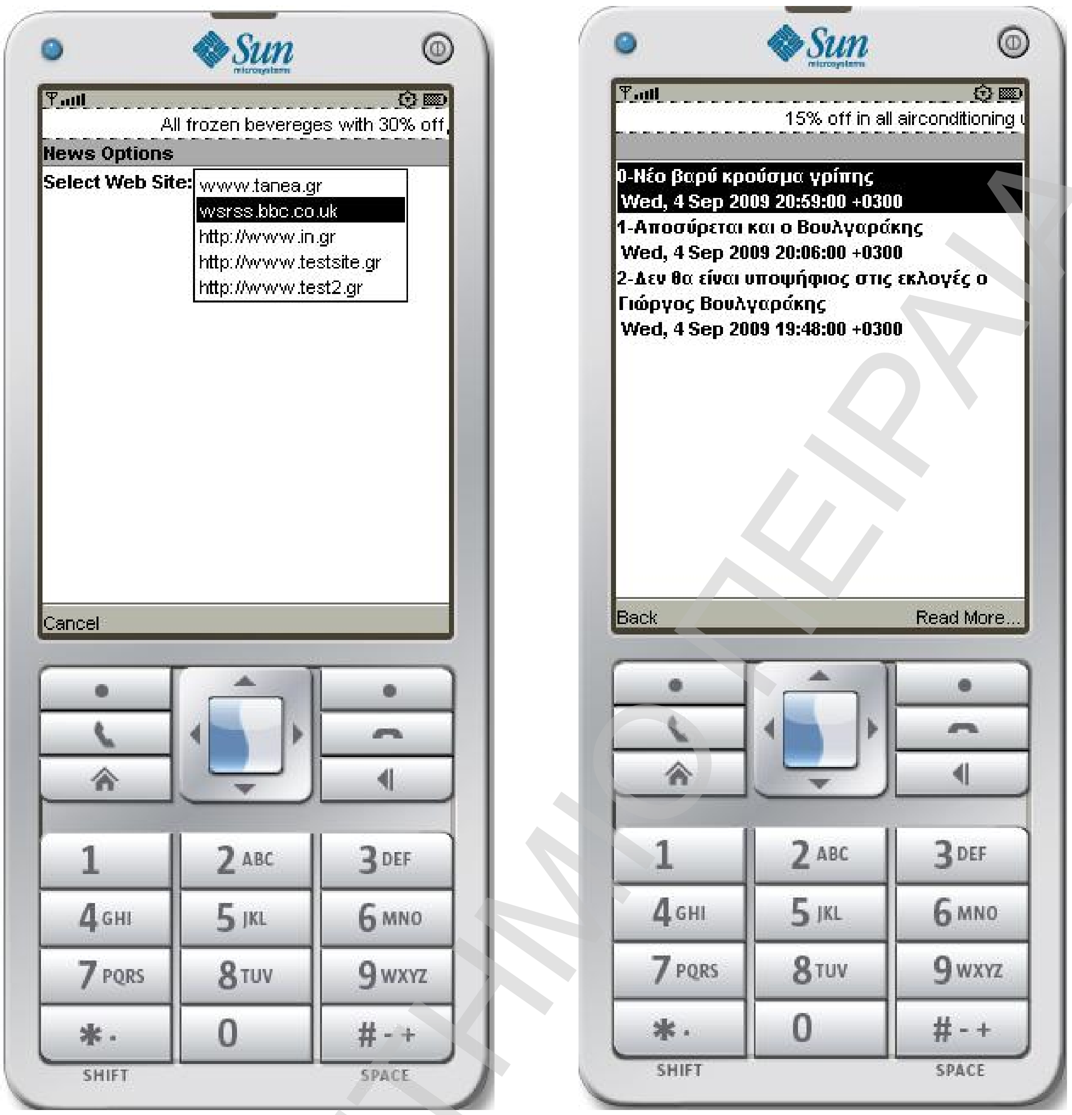


Figure 14: News Use Case Diagram

The service provides the following methods:

addWebSite:String:	***Adds a new RSS-enabled web site to the user database.***
getWebSites:String[]:	***Returns a list of the user's stored web sites.***
getRSSFeeds:RSSObject[] :	***Returns a list of structured RSSObjects containing all the RSS news from the specific site.***


All frozen bevereges with 30% off

News Options

Select Web Site: www.tanea.gr
wsrss.bbc.co.uk
http://www.in.gr
http://www.testsite.gr
http://www.test2.gr

Cancel

15% off in all airconditioning

0-Νέο βαρύ κρούσμα γρίπης
Wed, 4 Sep 2009 20:59:00 +0300
1-Αποσύρεται και ο Βουλγαράκης
Wed, 4 Sep 2009 20:06:00 +0300
2-Δεν θα είναι υποψήφιος στις εκλογές ο Γιώργος Βουλγαράκης
Wed, 4 Sep 2009 19:48:00 +0300

Back Read More...

Figure 15: News Use Screenshots

The user is able to add an RSS site and store it to his account's database for his personal use. Then he can use that entry to retrieve the headlines. While browsing and reading the headlines, the user can ask for the whole article of this specific feed. The headline will be marked as read (not Bold font), so the user can know what headlines hasn't read yet.

7.6.3 Weather

The weather service uses some forecasting knowledge to express numbers with images.

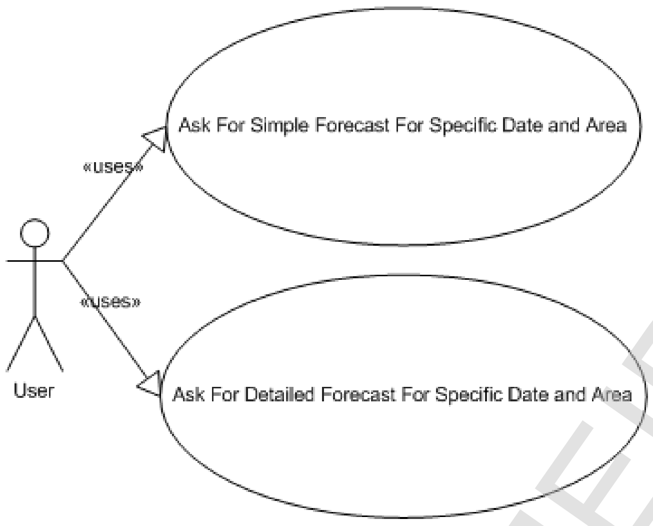
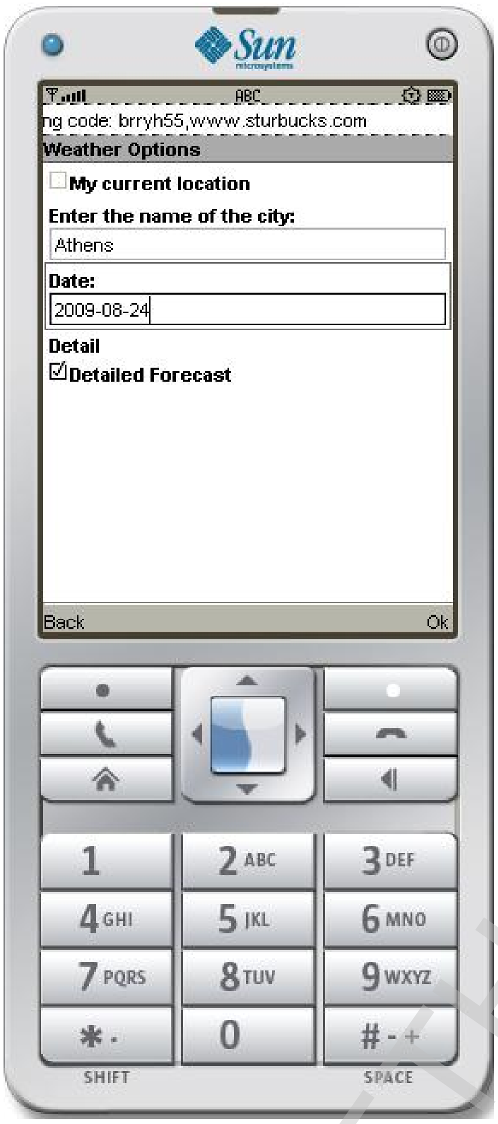

Ask For Simple Forecast For Specific Date and Area
«uses»
«uses»
User
Ask For Detailed Forecast For Specific Date and Area

Figure 16: Weather Service Use Case Diagram

The service provides the following methods:

getSimpleWeather:WeatherData	**: Returns an object containing a simple forecast for a specified lon, lat pair and date.**
getDetailedWeather:WeatherData	**: Returns an object containing a detailed forecast for a specified lon, lat pair and date.**
getSimpleWeatherByCity:WeatherData	**: Returns an object containing a simple forecast for a specified city and date.**
getDetailedWeatherByCity:WeatherData	**: Returns an object containing a detailed forecast for a specified city and date.**


ng code: brryh55,www.sturbucks.com

Weather Options

☐ My current location

Enter the name of the city:

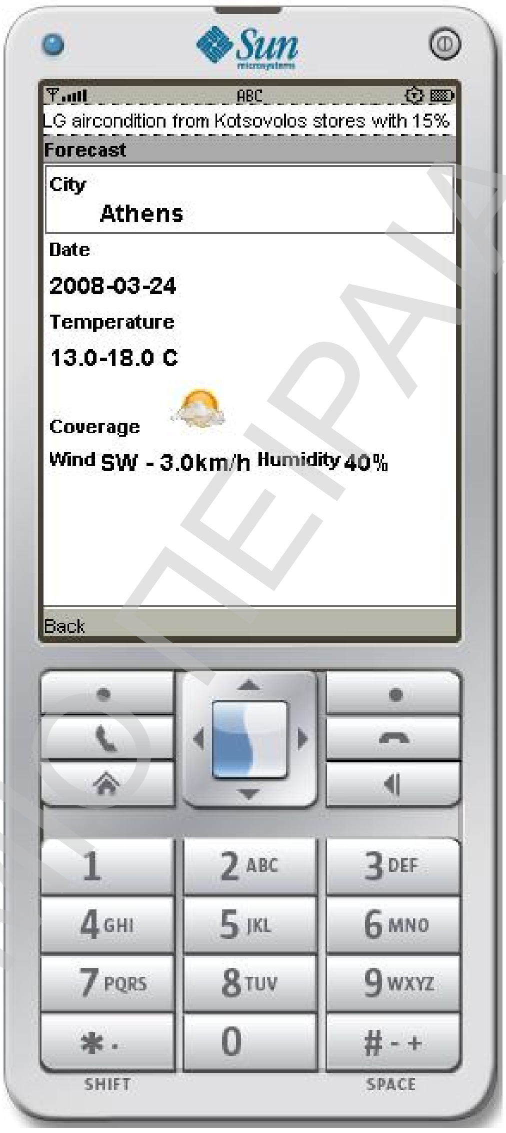
Athens

Date:

2009-08-24

Detail

☑ Detailed Forecast


LG aircondition from Kotsovolos stores with 15%

Forecast

City

Athens

Date

2008-03-24

Temperature

13.0-18.0 C

Coverage

Wind SW - 3.0km/h Humidity 40%

Figure 17: Weather Service Screenshots

The forecast screen is a form with fields, populated with text and images, using different layout from the rest of the forms, in order to have items next to other items.

The user is able to enter the Location/City he is interested in, or check "My Current Location", where the city entered will be ignored. The date can be entered by the user, and finally an option if a detailed forecast or a simple one is wanted. In the simple forecast the user will see the location, the date, the temperature, and the coverage. In the detailed view the user will see the same set and in addition, the wind speed, wind direction and humidity are visible. More details about the forecasting algorithm can be found in the implementation section.

7.6.4 POI

The POI service allows the user to search and manage Points of Interest around his location.

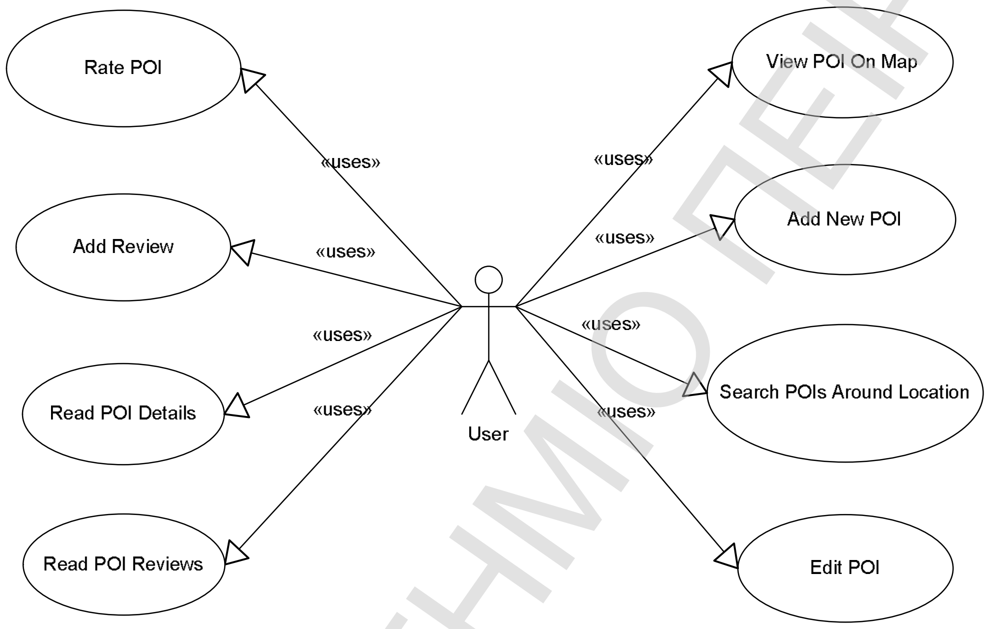


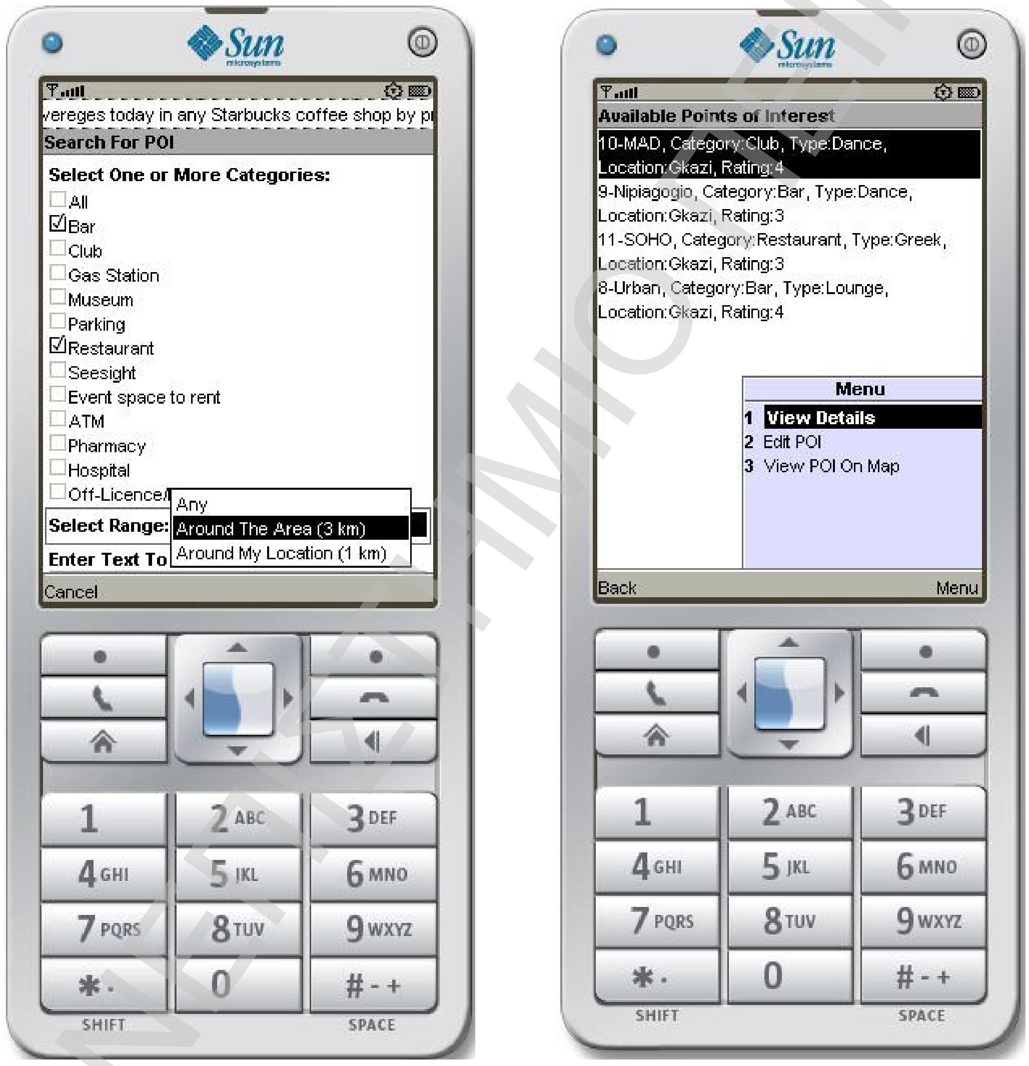
Figure 18: POI Service Use Case Diagram

The service provides the following methods:

addNewPOI: String	**:Adds a new POI to the database**
editPOI: String	**:Edits a specific POI.**
getCategories: String[]	**: Returns the available POI categories.**
getTypes: String[]	**: Returns the available types for a specific category.**
addPOIReview: String	**:Add a review for a POI.**
getPOIDetails: POIData	**: Returns the details of a POI.**
getReviewsList: POIReviewsData[]	**: Returns a list of POIReviewsData containing reviews for a specific POI.**
getPOIList: POIData[]	**: Returns the list of the POIs.**

The user is able to search and browse for POIs around his location with criteria like, *category*, *range* and specific words. He can then browse the results. Each result uses 2 lines and contains some useful info like, the *title*, the *category*, the *type*, the *location* and the *rating*. By choosing any result the user is able to view more details like the *contact info*, a *photo*, etc.

The user is also able to add a Point of Interest and at the same time add a review along with a rating from 1 to 5. Finally he/she can edit a POI or add a review for a pre-existing one and rate it.



vereges today in any Starbucks coffee shop by p

Search For POI

Select One or More Categories:

- [ ] All
- [x] Bar
- [ ] Club
- [ ] Gas Station
- [ ] Museum
- [ ] Parking
- [x] Restaurant
- [ ] Seesight
- [ ] Event space to rent
- [ ] ATM
- [ ] Pharmacy
- [ ] Hospital
- [ ] Off-Licence/

Select Range: Any / [✓ Around The Area (3 km)] / Around My Location (1 km)

Enter Text To

Cancel

Available Points of Interest

10-MAD, Category:Club, Type:Dance, Location:Gkazi, Rating:4
9-Nipiagogio, Category:Bar, Type:Dance, Location:Gkazi, Rating:3
11-SOHO, Category:Restaurant, Type:Greek, Location:Gkazi, Rating:3
8-Urban, Category:Bar, Type:Lounge, Location:Gkazi, Rating:4

Menu
1 View Details
2 Edit POI
3 View POI On Map

Back Menu

Figure 19: POI Service Screenshots

The user can choose between a range of *1 km, 3 km* and *unlimited* to reduce the results and make browsing easier. He/She can choose a POI from the list to view more details, edit it or view the location on a map relative to his/her location. Finally through the POI details screen, he/she can choose to add a review, or

read the *last 3, the last 10* or *all* the reviews written by others concerning that POI. Both range choices, and numbers of reviews were chosen in order to make the browsing easier and at the same time minimize the amount of data received from the server wherever is possible. So for example by choosing around "*My Location (1 km)"* in the range option it will show only the Points that are within 1 km, and if he/she is not satisfied he can try again with a longer range or even unlimited range.

7.6.5 Events

The Events service is very similar to the POI service. It provides users with information about where they can find a specific kind of place. So for example if a user is interested about a beach party or a festival this is the service he needs. It is more about the event rather than the place.

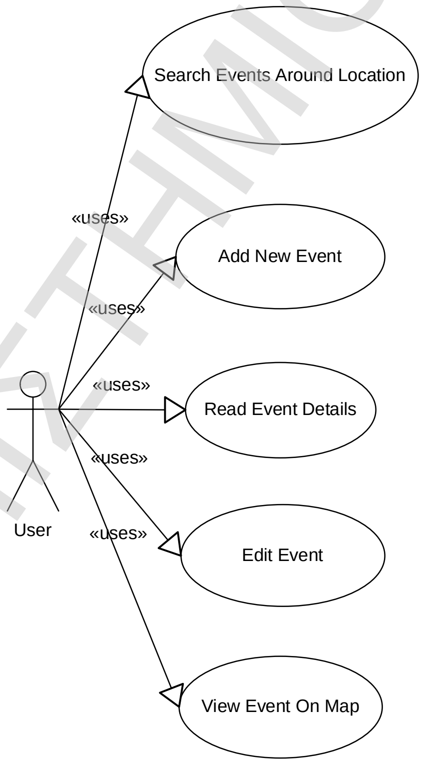


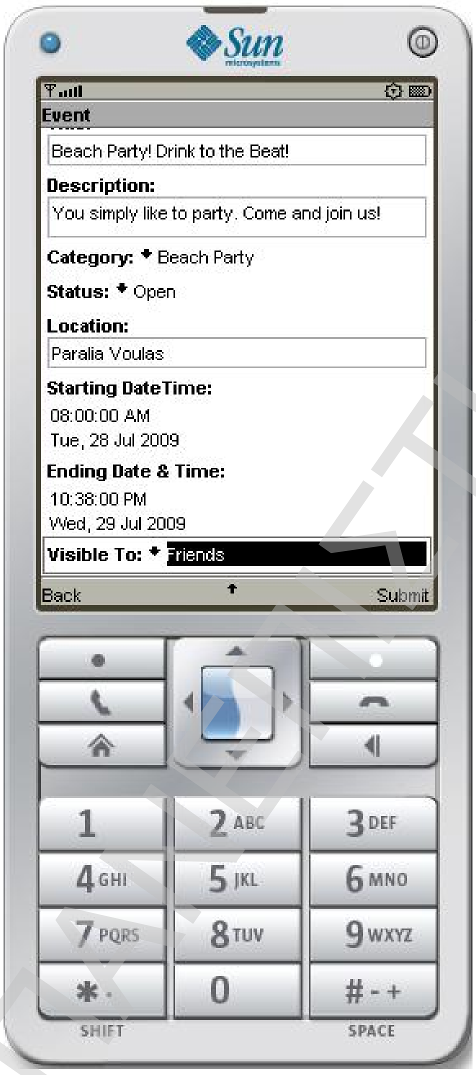
Figure 20: Events Service Use Case Diagram

The service provides the following methods:

getCategories: String[] : Returns the available event categories.

addNewEvent: String	**: Adds a new event to the database**
editEvent: String	**: Edits a specific event.**
getEventDetails: EventData	**: Returns an EventData object with the details of an event**
getEventsList: EventData[]	**: Returns a list of EventData objects.**

Just like the POI service, the user is able to search for events around his location, add a new event, read details about the events he/she is interested in, edit an event and finally view the event location on a map.


Event

Beach Party! Drink to the Beat!

Description:
You simply like to party. Come and join us!

Category: Beach Party

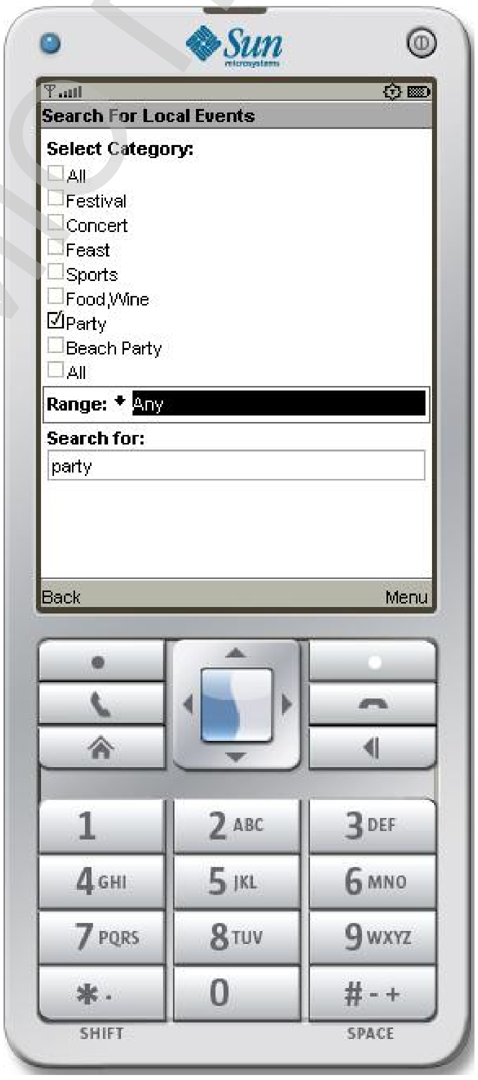
Status: Open

Location:
Paralia Voulas

Starting DateTime:
08:00:00 AM
Tue, 28 Jul 2009

Ending Date & Time:
10:38:00 PM
Wed, 29 Jul 2009

Visible To: Friends


Search For Local Events

Select Category:
☐ All
☐ Festival
☐ Concert
☐ Feast
☐ Sports
☐ Food,Wine
☑ Party
☐ Beach Party
☐ All

Range: Any

Search for:
party

Figure 21: Events Service Screenshots

The user can choose the category of the event which can be events like *"Festival", "Concert", "Beach Party", "Feast",* etc. The range of the search includes "Around My Location", which covers the area with radius of *1 km*, and "*Any*", to search for all the events on the database. That is used in combination with the search field, where the user can enter text to limit the browsing results or search for something in particular.

The event has different properties than the POI. It includes the usual *title, description, and category, the status, the Starting Date & Time, the Ending Date & Time*, and finally the *visibility*. The *visibility* determines who will be able to see the event. It can be either visible to "*All*" or "*Friends only*".

7.6.6 TrafficAlerts

The platform provides a service that can serve many purposes with some unique features. Since the exact location of the user is known, an estimation of his direction can be performed. This ability can help avoid problems or even hazards on someone's route. That, along with automatic notifications, makes a very helpful service.

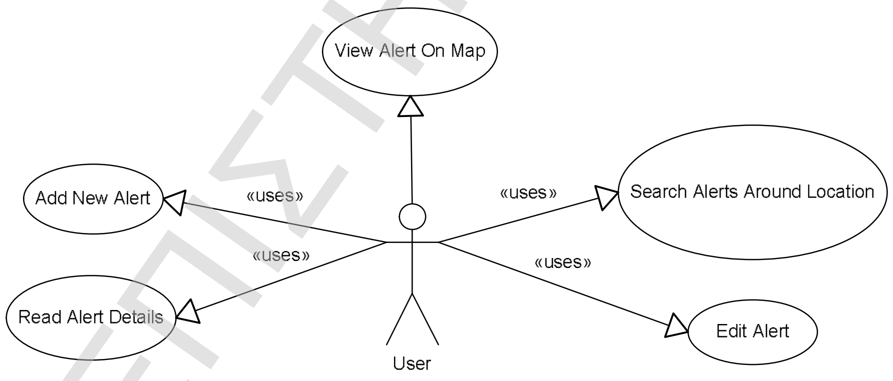


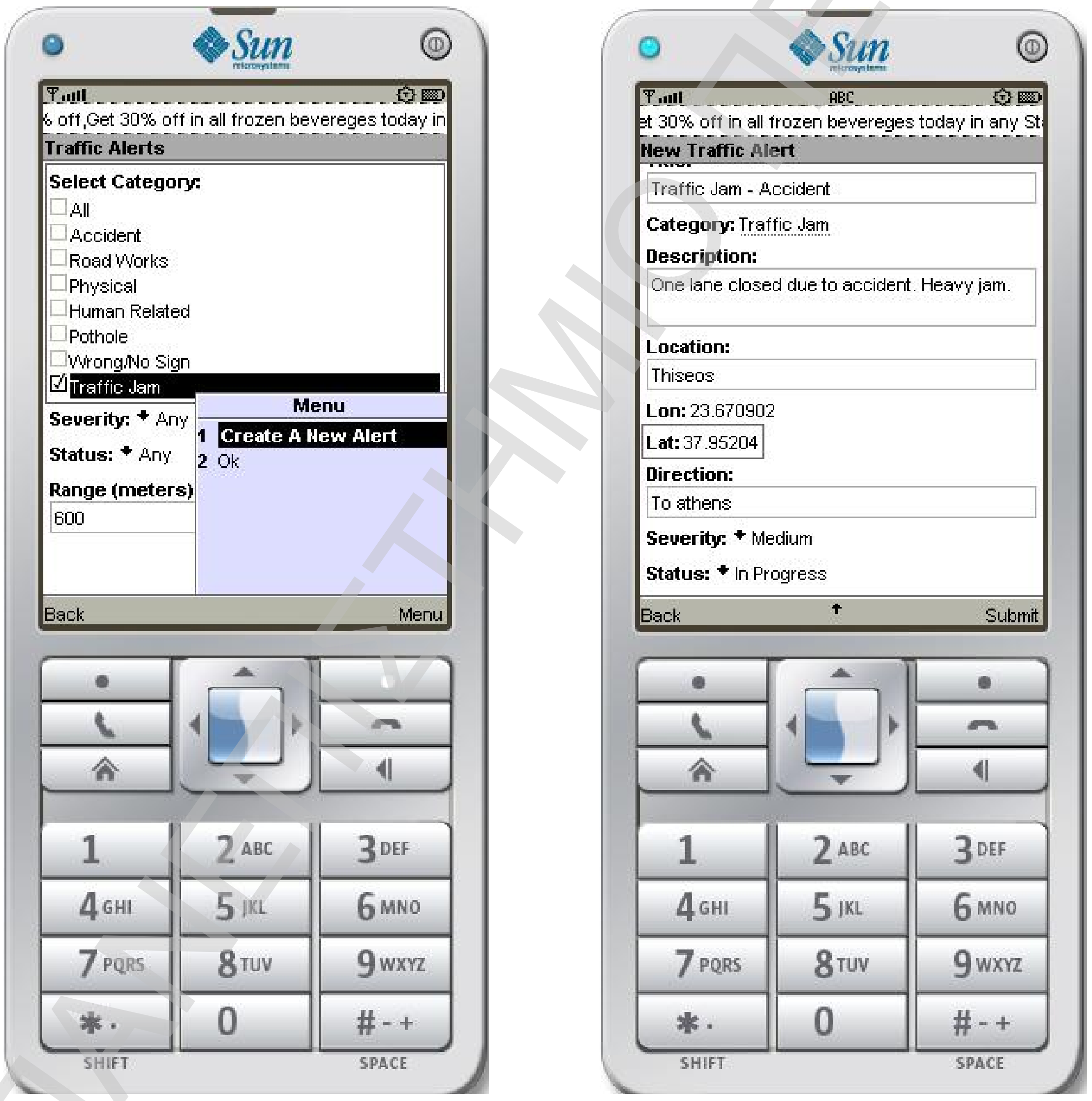
Figure 22: Alerts Use Case Diagram

The service provides the following methods:

getCategories: String[]	**: Returns the available alert categories.**
addNewAlert: String	**: Adds a new Alert to the db.**
editAlert: String	**: Edits a specific alert.**
getAlertDetails: AlertData	**: Returns an AlertData object containing the details of the alert**

getAlertsList: AlertData[] : Returns a list of AlertData objects containing the alerts.

Users can add or edit/change an alert that has come to their attention like a traffic jam they are stuck in, an accident they witness or were part of, a wrong sign or lack of sign that they believe it can be dangerous or event a pothole so that other people can avoid or be careful. The user can also search for alerts according to criteria of categories like the ones mentioned before, as well as road works, physical catastrophes, or human related problems. Finally he or she can view the location of the alerts on a map relative to their current location.



% off,Get 30% off in all frozen beverages today in

Traffic Alerts

Select Category:
- ☐ All
- ☐ Accident
- ☐ Road Works
- ☐ Physical
- ☐ Human Related
- ☐ Pothole
- ☐ Wrong/No Sign
- ☑ Traffic Jam

Severity: Any

Status: Any

Range (meters): 600

Menu
1 Create A New Alert
2 Ok

Back Menu

et 30% off in all frozen beverages today in any St

New Traffic Alert

Traffic Jam - Accident

Category: Traffic Jam

Description: One lane closed due to accident. Heavy jam.

Location: Thiseos

Lon: 23.670902

Lat: 37.95204

Direction: To athens

Severity: Medium

Status: In Progress

Back Submit

Figure 23: Alerts Service Screenshots

The alert can have the title that describes it in 2-3 worlds what is it about, a description that can explain in detail the problem, the location that this alert is on, as well as a helpful direction tag that can help people know for example if a danger is on a high way, in which direction is of the high way they will meet it. Finally a *severity* and a *status* tag to inform the users and the system how serious this danger seems and if it is still a threat.

The status tag could be used, so that the system can automatically clean the ended alerts. In contrary, the alerts that are in progress could be displayed on a big map on a computer, and with the contribution of the users, this map will be updated with all the problems and dangers of a road network, good enough even to save lives sometimes.

A unique feature that MEWS provides is the notification system. The settings of the client can be easily parameterised from the Home menu > Settings, and enable the "Automatic reminder on" for the alerts service. The user is even able to choose the time interval between each check, in order to allow him to save money. A 10 sec value is recommended. The system will check every 10 seconds for alerts that are located within a 700 m range from the user of any kind. If there are any, the same list of alerts will appear like the one that the user would see if it was chosen manually. The list will contain important only info for each alert. The user can select on and see the location on the map, see the details for the selected alert, or set the alerts as "*read*" and the system will not notify him or her again for these alerts.

7.6.7 BuddyLoc

BuddyLoc is a service where people can find, communicate and socialize with other people. It can replace the SMS texting with the bonus of location aware messages. The users don't need to ask "Where are you?" nor "Are you available? What are you up to?" People know what their buddies are doing and where (if they allow it) and they only need to ask if they are in the mood for doing something since there are near them. It is a revolutionary way to keep in touch with people on the go.

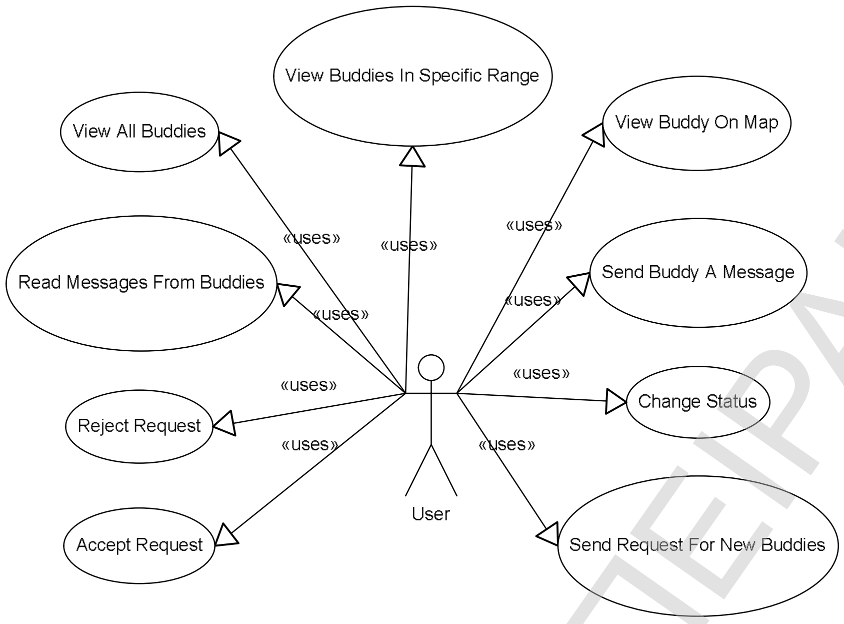


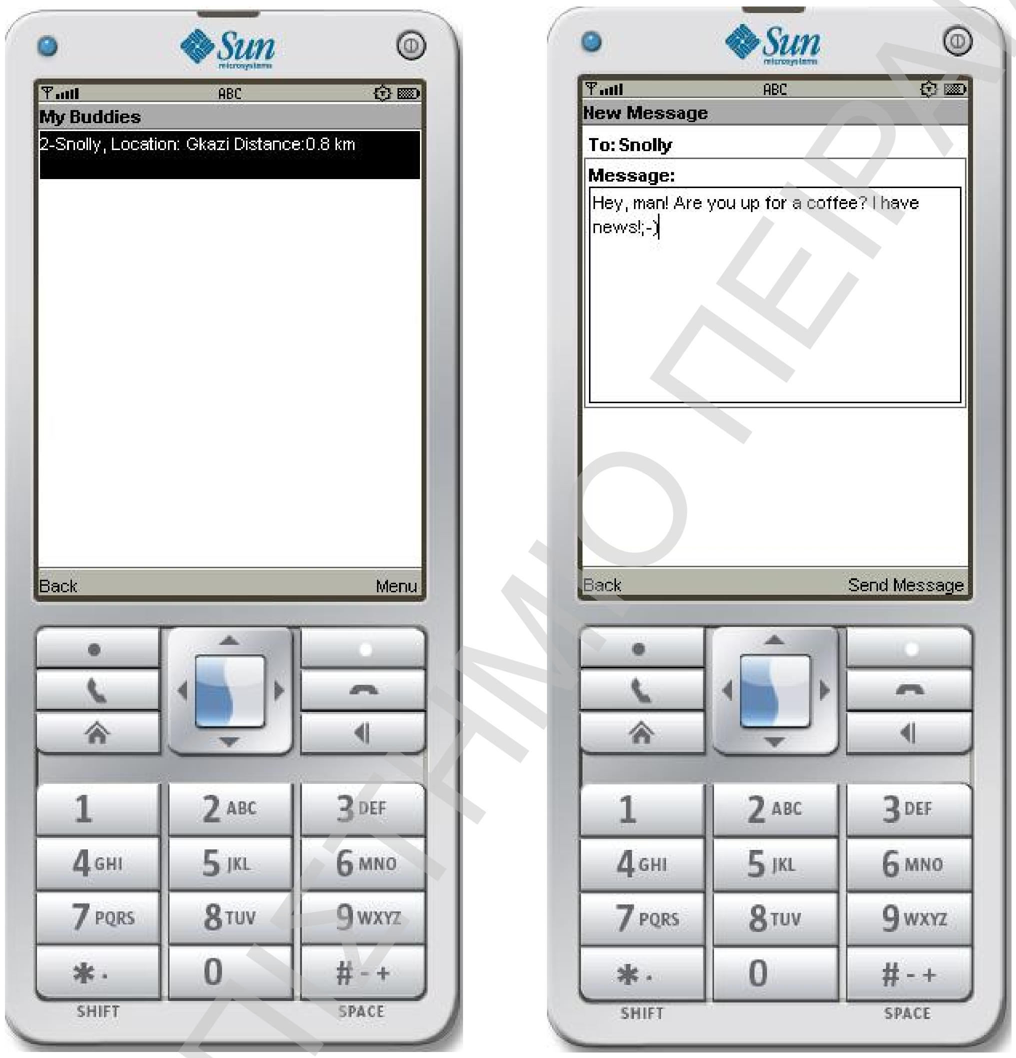
Figure 24: BuddyLoc Use Case Diagram

The service provides the following methods:

setCurrentLocation: String	**: Updates the Lon and Lat of the user to the db.**
getBuddies: BuddyData[]	**: Returns a list of BuddyData containing the details of friends in range.**
checkMessages: MessageData[]	**: Returns a list of messages of the user.**
getMessage: MessageData	**: Returns details of the specific message and the actual message.**
deleteMessage: String	**:Deletes a specific message.**
leaveMessage: String	**: Stores a message from a user to another.**
getLocation: String	**: Returns the location of a user.**
changeStatus: String	**: Changes the current status of the user.**
getStatus: String	**: Returns the current status of the user.**

It includes functionalities like changing the status, where people can write what they think or do, search for friends around them; view all their friends, their locations and their statuses. The users can also send and read messages to their

buddies, as well as send requests to people to add them to their buddy list. Of course a user can reject or accept an invitation.


My Buddies

2-Snolly, Location: Gkazi Distance:0.8 km

New Message

To: Snolly

Message:

Hey, man! Are you up for a coffee? I have news!;-)

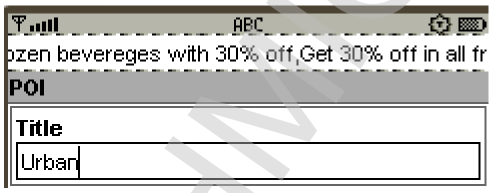
Figure 25: BuddyLoc Service Screenshots

The user can give the exact distance to search for friends. The list with the results will include the *nickname*, the *location*, the *distance* and the *status* of the buddy. The users can always prevent the application to publish their location since it is sensitive information. Someone can select a buddy to leave a message or see the location on the map relative to his/her.

7.6.8 Advertisement

Another service provided by the MEWS platform is the advertisements service. This service is different though. There is no interaction with the user other than the fact that the user sees advertisements on the top of the screen while using his mobile client.

The advertisements that someone sees are relative to his location. That means that if the user is near a starbucks coffee shop, he/she might see an ad concerning the products, the services and special offers about that shop. In case there is more than one ad that could be displayed to the monitor of the mobile phone, the system will randomly choose or can be configured by the administrators to give priority according to a variable that is in fact the "weight" of the ad. That of course depends on the clients that are the shops that want to be advertised. The weight would be larger for a shop that paid more than another that paid less. So that shop has more probabilities the user to see its ad.



ozen bevereges with 30% off,Get 30% off in all fr

Urban

Figure 26: Advertisements Service Screenshot

The service provides the following methods:

getAd[]:String[]	**: Returns an array containing an advertisement data.**

The user is not able to disable this feature, so he will always see ads scrolling on the top of the screen. The ad contains the title the description, along with contact info, and finally a link. More details about the advertisement service can be found in the implementation section.

7.6.9 Location Creation Algorithm

Since the MEWS platform is designed to be enriched mainly by users, at the beginning the data entries will be very few and the database will be populated while users use the application. Just like that, usable entries like locations will be automatically created and updated by the system, while users change and create Points of Interest.

While the user will add the first poi in the db the location that will enter, lets say "Neo Faliro" be added to the locations in order to be matched with the rest of the entries that will be entered later.

So the first time the POI that is located in Neo faliro will create an entry with start lon/lat and end lon/lat the same pair.

First POI
23.666267
37.949772

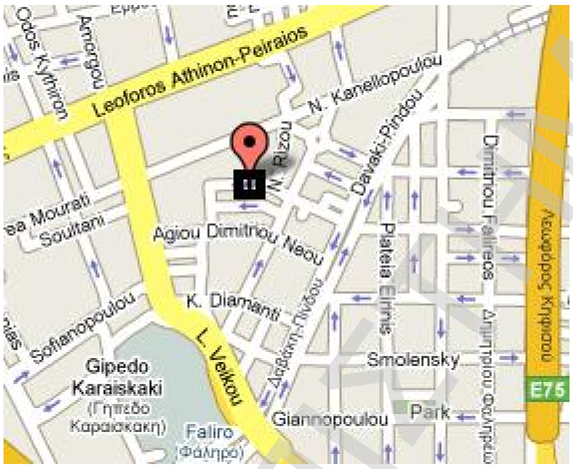


Figure 27: Location Expansion Phase 1

Neo Faliro Area
Lon Start :23.666267
Lat Start : 37.949772
Lon End :23.666267
Lat End : 37.949772

In the image above the map section you can see is the area of "Neo Faliro". The first time a POI was added with given location "Neo Faliro" the black rectangular was assumed as the area. This area is the same lon/lat pair for the bottom left corner (start) and top left corner (end) of the rectangular.

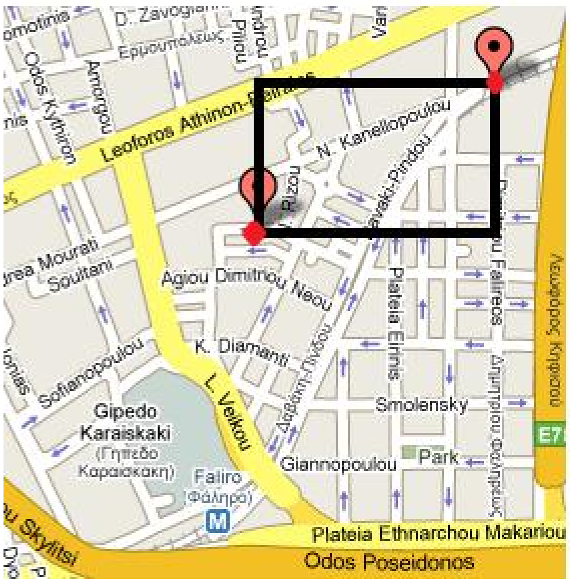


Figure 28: Location Expansion Phase 2

Second POI

23.670902

37.952040

Neo Faliro Area

Lon Start :23.666267

Lat Start : 37.949772

Lon End :23.666267

Lat End : 37.949772

The second time a POI with the same location will be added, the area will expand accordingly. The lon/lat pair for the bottom right will be the smallest candidate values and the pair lon/lat for the top left corner will be the largest candidate values.

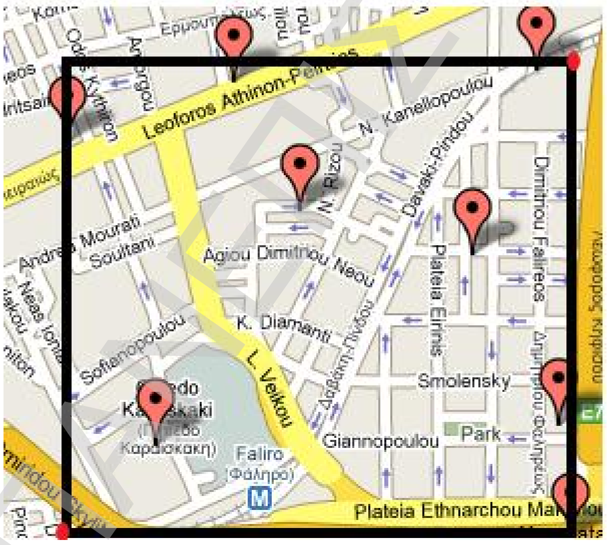


Figure 29: Location Expansion Phase 3

After many POI additions

Neo Faliro Area

Lon Start : 23,661246

Lat Start : 37,944595

Lon End : 23,672104

Lat End : 37,951397

Finally, the area will be large enough that can cover the actual area. All the POIs that are matched to be inside that rectangular are considered to be inside the "Neo Faliro" and no other area will be added in the database.

8 Implementation

8.1 Decisions

After a plan was arranged with the help of the Gantt chart, some decisions should be taken for the integration. As far as the coding standards concerned, a particular format was chosen for methods and variables. The names of the variables follow the pattern: "wordWordThirdWord".
No underscores are used and every word starts with capital. The methods and the variables start though with small letter at the very beginning. Classes on the other hand start with capital letter.
There is no static variables and the access of variables from other classes are done with the use `get()` and `set()` methods.
The documentation of the code is very important for maintenance and reusability. Having a piece of code that cannot be read and reused easily is like not having the code at all. Proper comments provide the required information to the programmers that will need to edit the code, making easy to find exactly what they need to give to the system additional or better functionalities.

8.2 Locating Technologies

A very important part of the system is the location determination. The platform's power is the fact that it provides location aware services. In order to do that, the client device needs to provide means to withdraw somehow at least the longitude and latitude of the user. As described in the background section, there are various ways, each one with their pros and cons.
The algorithm provides the option to the developer to give parameters as accuracy, power consumption, charging if required, etc. The java class then will decide the way to get the coordinates.
The class that is responsible for the LocationProvider is the Locator.java.

Criteria criteria = new Criteria();

```
            criteria.setCostAllowed(true);
            criteria.setAddressInfoRequired(false);
            criteria.setPreferredPowerConsumption(Criteria.POWER_USAGE_MEDIUM);
            criteria.setHorizontalAccuracy(500);
            criteria.setVerticalAccuracy(500);
        locationProvider = LocationProvider.getInstance(criteria);
```

The Criteria object contains the parameters given by the developer, and will determine the choice of the provider. So the developer, is not deciding which method to choose, but instead gives the requirements and according to them, the appropriate provider will be chosen automatically. Finally the LocationProvider is created by passing as argument the criteria.

```
public Coordinates getCurrentLocation(){

    Coordinates c=null;
   // c= new Coordinates(0,0,0);
   // c.setLongitude(0.2);
   // c.setLatitude(0.2);

    try {


          createLocationProvider();
          l = locationProvider.getLocation(60);

           c = l.getQualifiedCoordinates();


        } catch (LocationException ex) {
            ex.printStackTrace();
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }

         return c;
}
```

The above method returns the coordinates from the LocationProvider. The object returned is type Coordinates. Then, at any time in the application the methods:

locator.getCurrentLocation().getLongitude()
locator.getCurrentLocation().getLatitude()

Can be called to retrieve a set of doubles with the longitude and the latitude of the user respectively.

8.3 RSS Reader

The RSS Reader service provides feeds of news from sites chosen by the user. The service communicates with the web site web service to retrieve the xml file. The following method is responsible for that procedure with the help of the jdom, rome and rssutils libraries.

```
@WebMethod(operationName = "getRSSFeeds")
    public newsservice.RSSObject[] getRSSFeeds(@WebParam(name = "site")
    String site) {
        if (!connected) {
           connected = db.connect();
        }

        Vector feeds=  utility.rssRead(site);
         RSSObject[] feedsList =new  RSSObject[feeds.size()];

        for(int i=0;i<feeds.size();i++){
            RSSObject feed= new RSSObject();

            feed.setTitle(( (Vector)feeds.get(i)).get(0).toString()   );
           feed.setDescription(( (Vector)feeds.get(i)).get(1).toString()   );
            feed.setDate(( (Vector)feeds.get(i)).get(2).toString()   );

            feedsList[i]=feed;

         }
        return feedsList;
    }
```

```
public Vector rssRead(String site){
    RssParser parser;
            Rss rss =null;

      try {

          parser = RssParserFactory.createDefault();

           rss = parser.parse(new URL(site));

         //Get all XML elements in the fee
          } catch (RssParserException ex) {ex.printStackTrace();

      } catch (IOException ex) {

      }

         Collection items = rss.getChannel().getItems();

        int  c=0;
         Vector feeds= new Vector();
         if (items != null && !items.isEmpty()) {

             //Iterate over our main elements. Should have one for each article
             // for (Iterator i = items.iterator(); i.hasNext(); System.out.println()) {
             Iterator i = items.iterator();
            for (int j = 0;j<3;j++) {
             Vector feed= new Vector();
                 Item item = (Item) i.next();
                 feed.add(item.getTitle());
                 feed.add(item.getDescription());
                 feed.add(item.getPubDate());
                feeds.add(feed);

                 c++;
             }
         }
        return feeds;
}
```

A vector feeds is created and is populated with objects of type Vector containing Items. The method getRSSFeeds is producing these objects by communicating

with the remote web service through an rss parser. The parser returns a collection of objects that are feed into the vector.
Next the resulting feeds Vector is accessed to get each item and populate an array of RSSObjects with the same properties as the feed Item. The resulting array list is returned by the method. Then it is passed to the central server corresponding method to be passed to the mobile client that requested it. The method in the client side populateNewsList mewsclient.newsservice.rssObject[] newsList () populates a list of a form in the mobile client with items numbered, the feed title and the date & time published. The actual object already contains the description of the feed and it is retrieved when the user selects an item from the feeds list to "Read More...".

```
public void populateNewsList (mewsclient.newsservice.rssObject[] newsList){

   this.getNewsList().deleteAll();
   for (int i = 0; i<newsList.length;i++){

      news.put(Integer.toString(i), newsList[i].getDescription());
      this.getNewsList().append(i+"-"+newsList[i].getTitle()+ "\n "+newsList[i].getDate(),
null);



   }
      Font unreadFont = Font.getFont(Font.FACE_PROPORTIONAL, Font.STYLE_BOLD,
Font.SIZE_MEDIUM);
   for (int j = 0;j<this.getNewsList().size();j++){
  this.getNewsList().setFont(j, unreadFont);
   }
```

The RSS object passed from the external web service is an xml document with standard tags ready to be parsed by the RSS parsers.

8.4 Weather Forecasting

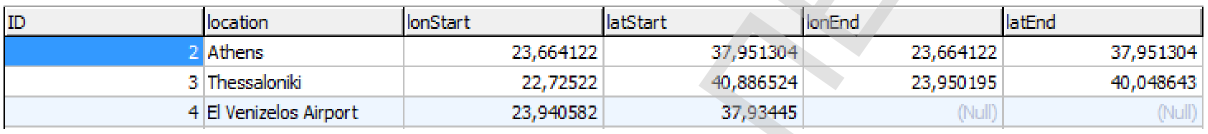
The coverage and rain probability data that are actually expressed with numbers, are part of a formula, that uses the percentage of the probability and the coverage with a scale from 1 – 100. In our case it was simplified. For the user

these numbers are useless, so they are converted to images. The following method lying on the client side, determines which gif image of coverage will so according to both these variables.

```
public String setCoverageImage(int coverage,int precipitation,String dateTime){
   String imageName="";



   if (coverage>0 && coverage <=25){
       imageName = "/sunny.gif";
   }
   if (coverage>26 && coverage <=50 && precipitation <70){
       imageName = "/partly-cloudy.gif";
   }
    if (coverage>26 && coverage <=50 && precipitation>70){
       imageName = "/shower.gif";
   }
    if (coverage>51 && coverage <=70 && precipitation<70){
       imageName = "/half-cloudy.gif";
   }
    if (coverage>51 && coverage <=70 && precipitation>70){
       imageName = "/half-cloudy-rain.gif";
   }
    if (coverage>71 && coverage <=100 && precipitation<70){
       imageName = "/full-cloudy.gif";
   }
   if (coverage>71 && coverage <=100 && precipitation>70){
       imageName = "/storm.gif";
   }

   try {
            coverageIcon = Image.createImage(imageName);
   getCoverageImageItem().setImage(coverageIcon);
   } catch (java.io.IOException e) {
            // e.printStackTrace();
         }

    return imageName;
}
```

The formula has been translated to 7 cases. The coverage alone is not enough. The precipitation will determine the rain probability. So for example in the third case if the coverage is between 26 and 50 with precipitation more than 70 the icon will be a partly cloudy image with raindrops, meaning showers. In contrary, the sixth case the coverage can be very high like between 70 and 100 and precipitation with less than 70 will mean fully clouded but without any raindrops.

8.5 Location Creation Algorithm



ID	location	lonStart	latStart	lonEnd	latEnd
2	Athens	23,664122	37,951304	23,664122	37,951304
3	Thessaloniki	22,72522	40,886524	23,950195	40,048643
4	El Venizelos Airport	23,940582	37,93445	(Null)	(Null)

Figure 30: Location Table Example

From the design a decision had to be made about how the locations would be created in the database. That decision was be used in all services that needed a way to store information about a location and link it to location or area. The implementation is the method that follows:

```
  double smallestLon = poiData.getLon();
        double smallestLat = poiData.getLat();
        double largestLon = poiData.getLon();
        double largestLat = poiData.getLat();

.....

query = "Select * from locations where location='" + poiData.getLocation() + "'";

            System.out.println(query);
            rs = db.execQuery(query);

            try {
                rs.next();
                locationID = rs.getString("ID");
                //Update the area range by changing the start coords or end coords as apropriate.
                //bottom left is start top right is finish.
                if (smallestLon > rs.getDouble("lonStart")) {
                    smallestLon = rs.getDouble("lonStart");
                }
                if (smallestLat > rs.getDouble("latStart")) {
                    smallestLat = rs.getDouble("latStart");
                }
                if (largestLon < rs.getDouble("lonEnd")) {
                    largestLon = rs.getDouble("lonEnd");
                }
                if (largestLat < rs.getDouble("latEnd")) {
                    largestLat = rs.getDouble("latEnd");
                }
                query = "Update locations set lonStart='" + smallestLon + "', latStart='" +
smallestLat + "', lonEnd='" + largestLon + "', latEnd='" + largestLat + "' where location='"
+ poiData.getLocation() + "'";
                System.out.println(query);
                db.execUpdate(query);
            } catch ( Exception e) {

                query = "Insert into locations values(null,'" + poiData.location + "','" +
poiData.lon + "','" + poiData.lat + "','" + poiData.lon + "','" + poiData.lat + "')";
                locationID = Integer.toString(db.execUpdate(query));


            }
```

The method tries to retrieve from the db a location entry with the given name. If it succeeds, it stores temporary the startLon, startLat, endLon, and endLat values from the db as both smallest and largest. Then it compares the new POI's values and replaces the smallest and largest values accordingly. The final result will be an expanded rectangular with the smallest Lon/Lat for the bottom left corner and the largest lon/lat for the top left corner.

If the SELECT query fails to return any results, means that there is no area with the same name and a new one will be created. The rectangular that will be considered as this area will have the same pair of coordinates for both corners. These coordinates will be the coordinates of the new POI just added.

8.6 Authentication

The authentication system is the way for the user to be recognised and communicate with security. As previously mentioned, the user must enter a combination of username and password to login to the system. This combination will be matched with the encrypted data in the users table, and if the result is successful, the user entry will change to status "*online*" and a random 12 alpha-numeric string will be generated. This token is linked to this user and is translated as the session unique ID. Every request is performed through this string. The string will be extracted from the request, it will be matched with the user, and only if a connected user is requesting a service the server will reply. The service responsible for the authentication and management of the request is the "ManageUsers" service that is located in the central MEWS server.

```
public void login(){

    try{
   token=manageUsers.login(getLoginScreen().getUsername(),
getLoginScreen().getPassword());

                    if( !token.equals("") && !token.equals(null)){
                    showNotification("You are now logged in!", 2000,getMEWS());
....
....

                    }else{
                        showNotification("Login Failed!", 2000,getMEWS());
                        getTicker().setString("Login Failed!");

                    } }
    catch (Exception e){}
}
```

The first login method is on the client side. The combination is send to the authentication web service and the user is informed if the authentication was successful or not with the appropriate notification. No other service request is possible if the user is not logged in first.

```
@WebMethod(operationName = "login")
   public String login(   @WebParam(name = "username") String username,
        @WebParam(name = "password") String password) {

      String usernameLower = username.toLowerCase();
      String passwordLower = password.toLowerCase();

      String token = "";



      token = db.connect(usernameLower, passwordLower);
      db.execUpdate("update users set connection='online', token='" + token + "'" + " where username = '" + usernameLower + "';");
      return token;
   }
```

The manageUsers service provides the login method that will use the db class to try to match the combination in the db using the connect() method. If it succeeds the token will be returned and the entry will be changed to "*online*" and the token will be stored for future authentication.

```
public String connect(String username, String password) {
   boolean logged_in = false;
   String token="";

   try {

     Class.forName("com.mysql.jdbc.Driver").newInstance();
     System.out.print("Driver found...");

   }

   catch (Exception ex) {
     System.out.print("Cannot connect to driver");
   }

   try {

    // conn = DriverManager.getConnection(URL, username, password);
```

```
      conn = DriverManager.getConnection(URL, "root", "");
      System.out.print("\n connected!");
      System.out.print("\n Checking user credentials...");


      try {

        stmt = conn.createStatement();
      }
           catch (Exception ex) {
             System.out.println("Statement creation error");

             logged_in = false;
      }

        if (stmt != null ) {
            token = checkCredentials(username,password);
          if(!token.equals("")){

          logged_in = true;
          }
        }


    }
    catch (SQLException ex) {
    //System.out.println("SQLException: " + ex.getMessage());
    //System.out.println("SQLState: " + ex.getSQLState());
    //System.out.println("VendorError: " + ex.getErrorCode());
      //ex.printStackTrace();
    logged_in = false;
    }
System.out.print("\n Credentials OK");
    return token;

  }
```

The *db.connect()* method will use the JDBC driver to connect to the database URL.

```
public String checkCredentials(String username,String password){
  String token="";
  String query= "";
  query= "Select * from users where users.username='"+username+"' AND
users.password=+MD5('"+password+"');";
  //System.out.println(query);
  ResultSet rs = execQuery(query);
  try {
    if (rs.next()) {
        Random r = new Random();

   token = Long.toString(Math.abs(r.nextLong()), 36);

 query= "UPDATE users set token='"+ token+"'where users.username='"+username+"'
AND users.password=+MD5('"+password+"');";
 execUpdate(query);

 return token;
    }else{
      return token="";

    }
  }
  catch (SQLException ex) {
    return token="";

  }
}
```

Finally, the check ***credentials()*** method executes the query to the database using the MD5 encryption algorithm. If the result is successful, the method generates a token using the ***Math.abs()*** and ***Math.Random()*** methods. The result is a Long number that is converted to String.
Now every request is escorted by this token. So a request that arrives to the server must be checked for its authenticity. The following code is performed for every request to the server:

....

```
....
...
 if (!connected) {
          connected = db.connect();
      }

      try { // Call Web Service Operation
          // mews.buddyloc.BuddyLocService port = service.getBuddyLocServicePort();

          java.lang.String userID = "";
          String query = "Select userID from users where token='" + token + "'";

          System.out.println(query);
          ResultSet rs = db.execQuery(query);
          if (rs.next()) {
             userID = rs.getString("userID");
```

If the resultSet(rs) returns null, then the user is not authenticated, and the rest of the code inside the if statement is never executed.

8.7 Maps

The application provides the feature to display maps with the location of the user and the location of another point of interest. This point can be an event, a POI, a buddy, or an alert. The mobile device communicates directly with the Google Maps API web service to retrieve a PNG image.

```
mapRetriever=new
MapRetriever(this.currentLon,this.currentLat,poiCoords,map_key,"b",this.getMapForm());
             mapRetriever.start();
             switchDisplayable(null, getMapForm());
```

The code above is executed to call the MapRetriever class, by passing as arguments, the current longitude and latitude of the device, the coordinates of the POI, the map_key that is retrieved when use register in the Google Maps API web site, the symbol that has as label on the POI point on the map, and finally the form the map will be displayed on.

```
public class MapRetriever extends Thread {

    String lon;
    String lat;
    String POIsCoords[][] = null;
    String mLon;
    String mLat;
    String key;
    Image image;
    Form form;
    String symbol;

    public MapRetriever(String lon, String lat, String[][] POIsCoords, String key,String
symbol, Form form) {
        this.lon = lon;
        this.lat = lat;
        this.POIsCoords = POIsCoords;
        this.symbol=symbol;
        this.key = key;
        this.form = form;

    }

    public void run() {
        getMap(lon, lat, POIsCoords, key,symbol, form);
    }

    public byte[] getMap(String lon, String lat, String[][] POIsCoords, String key,String
symbol, Form form) {
        HttpConnection httpConnection = null;
        DataInputStream dataInputStream = null;
        byte[] buffer = null;
        String markersString = "";

        for (int i = 0; i < POIsCoords.length; i++) {

            if (POIsCoords[i][0]!=null)
            markersString = POIsCoords[i][0] + "," + POIsCoords[i][1] +
",yellow"+symbol+"%7C," + markersString;

          }
  try{
```

```
 markersString = markersString.substring(0, markersString.length()-4);
  }
 catch (Exception e){}

 String                                        url                                        =
"http://maps.google.com/staticmap?&size=240x240&maptype=mobile&markers=" + lat +
"," + lon + ",blueu%7C" +markersString + "&key=" + key;

       try {
          System.out.println("Initializing Connection...");
          System.out.println(url);
          httpConnection = (HttpConnection) Connector.open(url);
          httpConnection.setRequestMethod(HttpConnection.GET);

          dataInputStream = httpConnection.openDataInputStream();
          int length = (int) httpConnection.getLength();
          System.out.println("Connection Initialized");
          if (length != 0) {

             buffer = new byte[length];
             dataInputStream.readFully(buffer);

          } else {
             int c;
             ByteArrayOutputStream byteArray = new ByteArrayOutputStream();
             System.out.println("Lenght 0");
             while ((c = dataInputStream.read()) != -1) {
                byteArray.write(c);
                System.out.println("C:" + c);
             }
             buffer = byteArray.toByteArray();
             byteArray.close();
          }
       } catch (Exception e) {
          System.out.println("Connection Failed");
       } finally {
          try {
             if (dataInputStream != null) {
                dataInputStream.close();
             }

             if (httpConnection != null) {
                httpConnection.close();
            }
        } catch (Exception e2) {
            System.out.println("IO Failed");
        }
    }

    try {
        form.delete(form.size() - 1);
    } catch (Exception e) {
    }
    form.append(Image.createImage(buffer, 0, buffer.length));

    return buffer;
  }
}
```

The map retriever class extends a Thread. It is highly recommended to use threads for a task like that in order to avoid dead-locking the application. The *run()* method of the thread will call the *getMap(lon, lat, POIsCoords, key,symbol, form)* method that retrieves the graphics.
Firstly, a string with the coordinates of the points of interest is created in a specific form. Then this string is concatenated with the *URL* string along with the size of the image, the type of the map, and the markers string.
Then an http connection is initialized and the request is send just like it would be using a browser. The request returns the PNG image with the map. The *dataInputStream* is loaded on a buffer with the same size and finally in a *ByteArrayOutputStram*. The bytes are then converted into an image using the *Image.createImage()* method and appended to the form.

8.8 Charging System

The charging system is part of the manageUsers service and is used in order to charge the user for the usage of the services if applicable. The values of each service can differ, so the system asks for the charge for each request from the database. When a request arrives to the central server, before it forwards the request to the back-end services, it authenticates the user, and checks his balance. It compares the value of the request with the balance and only if there is

enough credit, it proceeds. Otherwise, the user gets a notification, informing him that there is not enough credit in his account.

```
public boolean chargeService(String token,double charge){
        boolean charged= false;

       String userID="";
       double credit=0;

         String query = "Select userID,credit from users where token='" + token + "'";

            System.out.println(query);
            ResultSet rs = db.execQuery(query);

            try{
            rs.next();
            userID = rs.getString("userID");
            credit = rs.getDouble("credit");
            }catch (Exception e){}
              System.out.println(charge);

           if(credit>0){

            query = "update users set credit=" + (credit - charge) + " where userID='"+
userID+ "'";

            System.out.println(query);
             int key = db.execUpdate(query);
             charged=true;
           }
        return charged;
    }
    public double getPrice(String service){
        double price =-1;
 String query = "Select * from charges where serviceDescription='" + service + "'";

            System.out.println(query);
            ResultSet rs = db.execQuery(query);
            try{
            rs.next();
            price = rs.getDouble("charge");
            System.out.println(price);
        }catch (Exception e){}

      return price;
    }
```

The class responsible for the charging of the user is the CashDesk class. The chargeService() method gets the user info from the db and uses the getPrice() method to get the price of the service the user wants to use. The user balance is then updated to the new balance = oldBalance-price. The method returns a Boolean which is true if the user was charged for the service or false if he was not.

8.9 Alerting Thread

The alerting mechanism is complex due to the fact that it uses multiple threads that call other threads in order to retrieve the location and direction and avoid deadlocks of the GUI. The alerting notification works on the background and its job is to pop up the list of the alerts according to the criteria and the direction of the user.

```
public class AlertsThread extends Thread {

    RecordStore MEWSSettings = null;
    Utilities utility = null;
    boolean finished = false;
    TrafficAlertsService trafficAlertsService = null;
    MewsMIDlet midlet;
    String status = "";
    String severity = "";
    String lon = "";
    String lat = "";
    String token = "";
    String categories[] = new String[1];
    String range = "";
    int interval = 10000;
    mewsclient.trafficalertsservice.alertData[] alerts;
    Locator locator=null;
    public AlertsThread(MewsMIDlet midlet, RecordStore MEWSSettings) {
        utility = new Utilities(MEWSSettings);
        this.midlet = midlet;
        this.MEWSSettings = MEWSSettings;
        categories[0] = "Any";
        locator = midlet.getLocator();
    }

    public void run() {

        finished=false;
        while (finished!=true) {

            try {

                try {
                    interval = Integer.parseInt(utility.searchSetting(MEWSSettings,
"AlertsTimeInterval"));
                } catch (Exception e) {
                }
                Thread.sleep(interval);

                try {
alerts = trafficAlertsService.getAlertsList(utility.searchSetting(MEWSSettings, "Token"),
categories, utility.searchSetting(MEWSSettings, "AlertsSeverity"),
utility.searchSetting(MEWSSettings, "AlertsStatus"), utility.searchSetting(MEWSSettings,
"CurrentLon"), utility.searchSetting(MEWSSettings, "CurrentLat"));
                    System.out.println("Checking ofr alerts notification
on:"+utility.searchSetting(MEWSSettings, "AlertsOn"));

                    if (utility.searchSetting(MEWSSettings, "AlertsOn").equals("true")) {

                        alerts = midlet.getAutomaticAlertsListWS();

                        if (midlet.populateTrafficAlertsList(alerts, true) != 0 &&
!midlet.getDisplay().getCurrent().equals(midlet.getTrafficAlertsForm()) &&
!midlet.getDisplay().getCurrent().equals(midlet.getTrafficAlertsList())) {



midlet.getTrafficAlertsList().removeCommand(midlet.getEditAlertDetails());
midlet.getTrafficAlertsList().removeCommand(midlet.getSetAlertAsRead());
                    midlet.alertData = midlet.getAutomaticAlertsListWS();

                    midlet.getTrafficAlertsList().addCommand(midlet.getSetAlertAsRead());

                    if
(!midlet.getDisplay().getCurrent().equals(midlet.getTrafficAlertsList())) {
                       midlet.switchDisplayable(null, midlet.getTrafficAlertsList());
                    }
                 }
              }

           } catch (Exception e) {
             // e.printStackTrace();
           }
        } catch (InterruptedException ex) {
        //ex.printStackTrace();
        }
     }
     finished= false;
}
```

The **AlertsThread** class extends a Thread and the run overridden contains a while loop in order to keep running with a time interval sleep. Every time the thread fetches from the server the alerts for the specific criteria and the new location of the user. The alerts should not be displayed if the user is going away from the alert. So instead of the location the thread calls a method **getAutomaticAlertsListWS()**.

```
public alertData[] getAutomaticAlertsListWS() {

     alertData[] alertsList = null;
     if (!isLoggedIn()) {
        return null;
     }

     String[] allAlertsCategories = new String[1];
           allAlertsCategories[0] = "All";
```

```
   try {

          if(directionAutomaticAlertThread==null){
          directionAutomaticAlertThread = new
DirectionAlertThread(this,MEWSSettings,token, allAlertsCategories,
utility.searchSetting(MEWSSettings, "AlertsSeverity"), utility.searchSetting(MEWSSettings,
"AlertsStatus"),"600");
          directionAutomaticAlertThread.start();
          }
         alertsList = directionAutomaticAlertThread.getAlertsList();
      } catch (Exception ex) {
          ex.printStackTrace();
      }
      return alertsList;
   }
}
```

The method will not simply get the current Lon and Lat, but instead will call another thread that will return the alerts for 2 pairs of Lon and Lat.

```
public class DirectionAlertThread extends Thread {

    RecordStore MEWSSettings = null;
    Utilities utility = null;
    boolean finished = false;
    TrafficAlertsService trafficAlertsService = null;
    MewsMIDlet midlet;
    String status = "";
    String severity = "";
    // String lon = "";
    // String lat = "";
    String token = "";
    String categories[] = new String[1];
    String range = "";
    int interval = 10000;
    mewsclient.trafficalertsservice.alertData[] alerts;
    Locator locator = null;
    String lon1, lat1, lon2, lat2;
    mewsclient.trafficalertsservice.alertData[] alertsList = null;

    public DirectionAlertThread(MewsMIDlet midlet, RecordStore MEWSSettings, String
token, String[] categories, String severity, String status, String range) {
        utility = new Utilities(MEWSSettings);
        this.midlet = midlet;
        this.token = token;
        this.categories = categories;
        this.severity = severity;
        this.status = status;
        this.range = range;

        this.MEWSSettings = MEWSSettings;
        locator = midlet.getLocator();
        lon1 = Double.toString(locator.getCurrentLocation().getLongitude());
        lat1 = Double.toString(locator.getCurrentLocation().getLatitude());
        lon2 = Double.toString(locator.getCurrentLocation().getLongitude());
        lat2 = Double.toString(locator.getCurrentLocation().getLatitude());
        try {
            alertsList = midlet.trafficAlertsService.getAlertsList(token, categories, severity,
status, range, lon1, lat1, lon2, lat2);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    public void run() {
        System.out.println("DEBUG: - Started direction thread.");
        //Get the current lon lat in case it is the first time and
        //the thread has not got the pairs yet and someone asks for them

        while (true) {
            int i = 0;
            finished = false;
            while (i != 2) {
                try {
                    lon1 = Double.toString(locator.getCurrentLocation().getLongitude());
                    lat1 = Double.toString(locator.getCurrentLocation().getLatitude());


                    try {
                        interval = 1000;
                    } catch (Exception e) {
                    }
                    Thread.sleep(interval);
                    //System.out.println("__________");
                    try {
                        //get the position twice with 2 seperate pairs of coords.

                        lon2 = Double.toString(locator.getCurrentLocation().getLongitude());
                        lat2 = Double.toString(locator.getCurrentLocation().getLatitude());

                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                } catch (InterruptedException ex) {
                    ex.printStackTrace();
                }
                i++;
            }

            try {

                alertsList = midlet.trafficAlertsService.getAlertsList(token, categories, severity,
status, range, lon1, lat1, lon2, lat2);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    }
```

The thread is also using its run method in combination with an infinite "while" to keep getting the Lon and Lat twice with a 4000 mills delay. Then the getAlertsList(token, categories, severity, status, range, lon1, lat1, lon2, lat2) is called and the alertsList array is populated. Then the consumer AlertsThread class can ask for these alerts and display them in the client's GUI.

```
....
....
....
 if (lat1 != null && lon1 != null & lat2 != null && lon2 != null) {

                    // utility.calcDistance(alertLat, alertLon, Double.parseDouble(lat1),
Double.parseDouble(lon1), "K") <= Double.parseDouble(range)
```

```
                    double dist1=  utility.calcDistance(alertLat, alertLon,
Double.parseDouble(lat1), Double.parseDouble(lon1), "K");
                    double dist2=  utility.calcDistance(alertLat, alertLon,
Double.parseDouble(lat2), Double.parseDouble(lon2), "K");

                    if (dist1>=dist2) {

                         tempList.add(alertData);

                      }
                  } else {
                     // tempList.add(alertData);
                  }
                  i++;
              }
              alertsList = new AlertData[tempList.size()];

              for (int j = 0; j < alertsList.length; j++) {
                  alertsList[j] = (AlertData) tempList.get(j);
              }
.....
....
....
```

The portion of code above is the part of the backend TrafficAlerts service that decides what alerts to return according to the user's direction. In order to do that the algorithm calculates the distance between the alert location and the first pair of the Lon, Lat and then the distance using the second pair. That way it can guess if the user is moving towards or away form the alert. If first distance is larger that the second on then the user keeps getting closer so the alert is added to the tempList array to be return to the client.

9 Testing

Testing is a fundamental part of the software development. It is used to discover potential errors and abnormal behaviors of the system. The soonest the problem is found the lower the cost end effort for programmers and managers will be. It takes a lot of time so a reasonable time period should be designated for that purpose. It usually takes place during the implementation and after it. The system must be tested a long time before the deadline, in case there are major problems with the application.[1]

Testing is more like an art than a science. That's because it not determined when a software is enough tested neither how a software must be tested. The second matter varies depending on the nature and the size of the software. [1]

There are many methods, approaches and techniques to test software and the MEWS platform is not a common one. It consists of multilayered web services and interfaces that exchange information both ways. A testing strategy should be applied. [1]

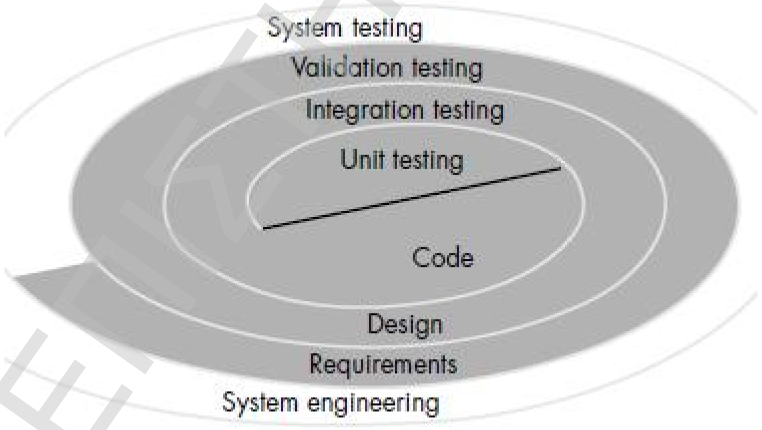


Figure 31: A Spiral Testing Strategy

A strategy can be viewed in the context of the spiral. Unit testing begins as the vortex of the spiral and concentrates on each unit (web service or method) of the software as implemented in source code. Testing progresses by moving outward along the spiral to integration testing, where the focus is on design and construction of the software architecture. Taking another turn outward on the spiral, we encounter the validation testing, where requirements established as part of the software requirements analysis are validated against the software that

has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole. [1]

The fact that the services are distinct pieces of code by design, it makes the testing process easier. In order to make sure every component of the web services works as planned, each method it provides is examined, and each one of this method would be tested separately. So by going from smaller to bigger, we would ensure the integrity and stability of the data and the platform. [1]

Two methods of testing are taking place during all steps of the strategy. The black–box, the white-box or sometimes they are both applied. In the first one, "we are more interested for the input and output relationship". We provide input to the module and we compare the output to the expected one. If the output is not what it should be, we sub-divide the system and we repeat the test for each subsystem. It is a test case design method that uses the control structure of the procedural design to derive test cases. Using white-box testing methods, the software engineer can derive test cases that (1) guarantee that all independent paths within a module have been exercised at least once,(2) exercise all logical decisions on their true and false sides, (3) execute all loops at their boundaries and within their operational bounds, and (4) exercise internal data structures to ensure their validity. [1]

The use case diagrams demonstrated in the previous sections worked as checklists for the individual tests needed to be run.

The second method is not opaque as the first one. The box this time is made out of "glass" and we try to peek inside it. We care this time about the internal structure of the module. , black-box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black-box testing is not an alternative to white-box techniques. Rather, it is a complementary approach that is likely to uncover a different class of errors than white-box methods. Black-box testing attempts to find errors in the following categories: (1) incorrect or missing functions, (2) interface errors, (3) errors in data structures or external data base access, (4) behavior or performance errors, and (5) initialization and termination errors. [1]

The order of testing must be done following the data flow of the module. When a previous module gives incorrect output to the next one that might confuse us while testing the second module. The fact that the platform is actually a client/server system, makes tracing of errors harder. An exception can occur in the client and the problem can actually be in the backend service because for example for some reason it returns null where it shouldn't. [1]

In order to test the system 2 tools, SOAPUI and JMeter, were used along the classical techniques that specialize at testing and tuning systems that are based on Web Services. These tools can be used to test performance, simulate heavy loads, test functional behaviour, monitor servers, as well as find problems and troubleshoot issues with the servers by simulating client requests. In the following part the criteria of testing and the testing process of the most basic modules is outlined. Both white and black box testing was used for all of them.

9.1 Unit Testing

Considering the process from a procedural point of view, testing within the context of computer software engineering is actually a series of four steps that implemented in sequential order. Initially, tests focus on each component individually, ensuring that it functions properly as a unit. Hence, the name "unit testing". Unit testing makes use of white-box testing techniques, exercising specific paths in the module's control structure to ensure complete coverage and maximum error detection. [1]

During the unit testing of the MEWS platform all services where tested using test tools and the JSP test client constructed easily by the IDE. The modules were given inputs with the intention to trigger the error handling mechanisms. [1]

9.2 Integration Testing

Someone could ask "If they all work individually, why do you doubt that they'll work when we put them together?" The problem, of course, is "putting them together"—interfacing. Data can be lost across an interface; one module can have an inadvertent affect on another, or sub-functions, when combined, may not produce the desired major function. Individually acceptable imprecision may be magnified to unacceptable levels, global data structures can present problems. Sadly, the list goes on and on. [1]

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. [1]

The objective is to take unit tested components and build a program structure that has been dictated by design. Black-box techniques are the most prevalent during integration, although a limited amount of white-box testing may be used to ensure coverage of major control paths, like scenarios. [1]
In the MEWS platform the integration test were performed to groups of services that interface with a specific client consumer. So actually a bottom-up depth-first approach was used. For example, the News service client on the mobile device is interfacing with the central server's news service, user management service, charging module, and these components with the corresponding news service on the back-end server. So various scenarios were used to ensure that all the components involved in handling the requests from the client, were working as they should as a whole separate system. All other services were tested the same way during this phase.

9.3 Validation Testing

After the software has been integrated, a set of high-order tests are conducted. Software validation is achieved through a series of black-box tests. A test plan outlines the classes of tests to be conducted and a test procedure defines specific test cases that will be used to demonstrate conformity with requirements. [1]
Both the plan and procedure were designed to ensure that all functional requirements are satisfied, all behavioral characteristics were achieved, all performance requirements were attained, documentation is correct, and human engineered and other requirements were met (e.g., transportability, compatibility, error recovery, maintainability). The final results for the platform were 100% successful.

9.4 System Testing

The last high-order testing step falls outside the boundary of software engineering and into the broader context of computer system engineering. System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that system elements have been properly integrated

and perform allocated functions. the software engineer should anticipate potential interfacing problems and (1) design error-handling paths that test all information coming from other elements of the system, (2) conduct a series of tests that simulate bad data or other potential errors at the software interface, (3) record the results of tests to use as "evidence" if they are blamed for a problem, and (4) participate in planning and design of system tests to ensure that software is adequately tested. In the sections that follow, we discuss the most significant types of system tests that where performed on the MEWS platform. [1]

9.4.1 Stress Testing

During earlier software testing steps, white-box and black-box techniques resulted in thorough evaluation of normal program functions and performance. Stress tests are designed to confront programs with abnormal situations. In essence, the tester who performs stress testing asks: "How high can we crank this up before it fails?" [1]

Stress testing executes a system in a manner that demands resources in abnormal quantity, frequency, or volume. For example, special tests may be designed that generate ten interrupts per second, when one or two is the average rate, input data rates may be increased by an order of magnitude to determine how input functions will respond, test cases that require maximum memory or other resources are executed, test cases that may cause thrashing in a virtual operating system are designed, test cases that may cause excessive hunting for disk-resident data are created. Essentially, the tester attempts to break the program. [1]

The tools that were mentioned at the beginning of this section were very useful for this purpose. Each service provided accepts one or a set of inputs and returns values like Strings, Booleans or complex objects and arrays of them. During the normal execution the amount and sizes are also normal. By injecting code to the client or server we can force the execution to behave abnormally and produce very large amount or size of data. For example the RSS News Service was injected with text and recursions to return to the client abnormal large data. That caused huge delays but not crashing of the system. The list of friends and reviews of POIs that was returned during the test was close to a thousand and the client GUI was annoyingly slower but the still managed to perform the commands of the tester.

All these kind of problems would only be caused if someone would have 2000 friends in range, or a web site would publish hundreds of book size news articles per day. Still that would be easily solved by predicting it, and changing the client to fetch parts of data each time. Maybe the "previous page/next page" would be a solution to implement it.
The performance testing was coupled with the stress testing of the system in our case. These kinds of tests utilize black-box tests to measure if the performance of the system is acceptable and within the requirements' specifications. A user for example, should not wait 1 minute on a Wi-Fi connection to retrieve the latest news. Neither should an alert be displayed on the screen after the user has passed by it. Of course, both stress and performance tests must be carried out from the early stages of unit testing.

9.4.2 Security Testing

Security testing attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration. During security testing, the tester plays the role of the individual who desires to penetrate the system. The security testing on the MEWS platform performed with various ways. The tester attempted to acquire, attacked the system with custom software, overwhelmed the system, thereby denying service to others, etc. The only problem that seems to appear is the overloading caused by extensive service requests, even though they were rejected because they where not escorted with a valid token. [1]
Given enough time and resources, good security testing will ultimately penetrate the system. The role of the system designer is to make penetration cost more than the value of the information that will be obtained.

10 Evaluation

10.1 Introduction

When designing something, a house, software, a car, the final product must meet the expectations of the client. In the house case it is easier. In the software case it is not that easy. When all the stages of the implementation of the system finished, the result is compared to the requirements. A final evaluation must be done by both the developers and the clients (users) to make sure the product is as close as possible to the one that was asked for. It is common, that software ends up being different for the initial requirements, but sometimes this is not necessarily bad, because it was changed in the process of the design or the development. That can happen because some of the involved parties realized that the product should be different since it would perform better like that, or look nicer or would be more user friendly with some changes. This is a natural result while testing the software.

10.2 Evaluating the System

MEWS in the final form could perform all the required tasks with 100% success and it includes even more small features to make the life of the user easier. It also obeys to the non- functional requirements like hardware requirements, performance, energy consumption etc. More details about the non-functional requirements can be found in section 6.3 Non-functional Requirements.

10.3 User Evaluation

After the project was finished users were invited to use the application. They were not given any specific guidelines neither a manual. Since it is well known that people don't read manuals as they should, it would be interesting to test how easy would it be to use it without having any help at all. The scenario that they

would follow was up to them. Three people undertook the testing. The testing feedback and bugs are described in the following table.

User	Experience	Feedback	Bugs Found
George	IT Professional	" Very promising and really handy" -Lack of speed. - Needs more user error handling.	Bug in buddyLoc person location found (fixed).
Angela	Inexperienced User/ Basic Mobile usage knowledge	"I like it. It was hard to use it at first, but after 15 minutes it was piece of cake. I enjoyed the buddyLoc a lot."	No bugs found.
Fannis	Average IT Skills/ experienced mobile user	"It is something I wanted on my phone. I liked it a lot. I would like it to be fasted though, and be able to add POI on map without being at the place."	Bug in POI review addition found (fixed)

Table 2: User Feedback Table

In general all three users found the application very interesting and promising. It was clear to them that it needs some tuning regarding the speed and the user feedback. For example the loading of news if there are many new feeds or from some web sites that the RSS feed brings the whole article takes a lot of time. The experience of the users was covering all three levels, from no IT skills and basic mobile usage, to IT professional and expert mobile usage.

Two of the users were using the buddyLoc service to communicate and find each other on the move. They exchanged a lot of messages without any problems. The third person used a lot the POI service to add places while using his bicycle downtown. Two of them found some bugs and they were fixed right away.

10.4 Usage Evaluation Study

A group of services like the ones that MEWS provides, could be integrated into our everyday routine. In order to test the platform in the best way possible, as it was already demonstrated, 3 users installed the client on their mobile phones and

used it for 4 days. The feedback was indeed very useful, but another perspective could be examined also. The perspective includes the financial aspect, and power consumption aspect.

10.4.1 Power Consumption

In order to evaluate the actual consumption of modern devices while using the MEWS services, all three users' devices were loaded with utilities, "pyBattery" and "Energy patrol", which are both freeware and easy to use, to measure the battery drain and give detailed statuses. The users were told to monitor these utilities before and after the use of the MEWS client. The following table describes the information the users gave after these 5 days, including the power consumption they believe the application caused, by comparing it with a normal day without the application. We can only assume these are estimates and not exact numbers.

User	News Usage	Weather Usage	POI Usage	Events Usage	Alerts Usage*	BuddyLoc Usage*	Minutes/ day	Extra Consumption
1	15%	5%	30%	0%	30%	20%	40	10%
2	0%	0%	10%	10%	20% (W/N)	60%	30	40%
3	10%	10%	30%	20%	0%	30%	90	30%

Table 3: Usage and Power Consumption Comparison Table

Users 2 and 3 chose to use mostly the BoddyLoc service in order to play a hide and seek game while communicating with messages sent by the same service. User 1 used the service for reading RSS news. He added his own RSS sites, and MEWS was keeping him informed for the latest headlines and articles.
There is significant difference of the minutes spent from user to user. The first one used it for 40 minutes and had an extra 10% of consumption. The second one used it for 30 minutes and had an extra 40% of consumption. Finally, the third one used it for 90 minutes and had a 30% of extra consumption. It is obvious that the duration of usage is not the criteria for measuring the drain of the battery. It is important what services and how the users used them.

The first user used the News services, while spending time adding and Reviewing Points Of Interest. The Traffic Alerts service was used with auto notifications OFF. So there was no extra GPS utilization or data transmission. The News service doesn't use the GPS at all. It only transmits and receives data on request. Similarly the POI service only uses the GPS when necessary, and the user spends a lot of time writing.
User 2 spent only 30 minutes using the application. He turned on the automatic notification for the Traffic Alerts Service. That means the user was using other services while the Traffic Alerts service was sending and receiving constantly data to and from the server, while updating the position of the user using the GPS.
The third user spent 3 times the time the previous user spent, and had less extra power consumption than him. Both users had the "allow my friends to know where I am" option ON.
From the data gathered form the users, we confirm that the MEWS platform is as energy-inefficient as all application that use GPS and remote transmission of data via 3G. It is also important to mention that depending on the services used, the consumption varies significantly. Services like Traffic Alerts and BuddyLoc that constantly update the location and have a periodic communication with the server are consuming much more battery that applications like POI and News that the GPS usage varies from none to rare, and the data transmission are only executed on user request.

10.4.2 Investigation of Financial Aspects

If MEWS would go on the market, users would be charged with two different ways. First for using of the service, and secondly for the data they would send and receive. Let's examine both cases and have an estimate of the actual usage.
Since it provides services, it could have 2 options for charging. Either with subscription, were people would pay a standard fee and would have unlimited access, or with per-service-request, where people would be charged nothing for adding and contributed in the database, and a specific amount for retrieving data like the POIs near them for example. For MEWS the developer chose the second option. The user can top up his account and start using the services. The user will not be charged for adding a POI but will be charged for retrieving a list of events, or Points of interest around him.

So according to the previous table, 3 users used the services for an average 50 minutes per day. That estimates about 100 requests. It is assume 70% would be chargeable, and 30% not. That totally depends of course from the services the user uses and for what cause. The news service for example has a lot of data traffic but can get all the news with one request. If the user wants to read the article, he will make a new request and even more data traffic.

In one of the mobile phones used for the user evaluation a data counter was used to measure the data exchanged. The user number 3 used the phone for 90 minutes, with about 100 requests to the services. Each service is estimated to charge 2 cents in average. The data counter showed 700 kb usage. So if the user has no data plan, meaning the user has to pay in average 0.5 Euros/MB, the user has to pay in total:

0,35 Euros (data) + 2 Euros (service)=2.35 Euros

That is only estimation of heavy usage with no data plan and using the services that have required a lot of requests. This number can be easily dropped to 1 euro per day, if the usage is lighter and the user has a data plan.

So as a conclusion the service would be feasible from the financial point of view, with both the users being satisfied paying for services they use a reasonable amount, and the company that could own the services to have also a reasonable profit.

11 Conclusion

The MEWS platform ended up being a system that with some tuning could be out on the market and be a hit. It manages to unify services that are actually needed by someone on the go and make his life easier and more enjoyable. The platform is by design expandable to allow more services to be hosted. Of course, since it is the dissertation of an MSc degree it can always be even more professional and more secure, while more features could be added to be ready for the competitive market.

Information and content is and will be for a long time the most important aspect of the electronic mobile world. Applications and services that provide it in smart and friendly way will always be on the scene.

11.1 Future Work

The MEWS platform is a framework of services with some of them interconnected. So each one of them is a small application providing its service and can be enriched with even more. Also tweaks and changes on the current ones can be done according to user feedback. Some of the ideas for future work are described bellow.

- Stock Service: A service that provides stock prices live from third party services could be added.

- Search for people: allow users to search with filters for people they are interested in.

- Allow users to upload pictures: The POI Service would much more helpful if the users could see a picture of the POI they are interested in. So the feature of uploading pictures would be a nice touch.

- Display turn by turn information: A not so easy task. There is no need to have turn by turn directions in real time, but a list of the directions for start point to destination would be useful for people that are completely unfamiliar with the area, like tourists.

- Allow pinpointing on Map: It was asked by a user. It would make the application a lot more functional by allowing the user to add any POI, event, or alert by pointing using the finger or a mouse pointer on the map. That requires a lot of effort and time. The coding would be quite complex.

- Make advertisement smarter and more targeted: The advertisement business is quit complex. There is no specific recipe for success. However, having smarter algorithms and making the advertisements more targeted would make the ads worth much more. For example coupons could be offered when people are right outside the stores. The ads would refer to people that are actually interested in, by demographic information aggregated in advance about the users from the system.

The platform is scalable. These are only some of the ideas that could be useful. A lot more could be added to make the platform more functional and helpful.

11.2 Lessons Learned

I learned a lot from this big project. It is clear to me how important the user opinion is, providing evaluation and feedback to make the software more functional and easier to use. The ideas are nothing if you cannot apply them in a way to be 100% usable and accessible. The rules of the Software Engineering are important in order to make a solid system, and have a final result that performs and looks like (or even better in that case) than the one that you used to imagine 1 year ago.

12 References

Books

[1]Software Engineering, A practitioner's Approach, Fifth Edition, Pressman and Ince, ISBN 0-07-709677-0.

[2]Database Systems, A practical Approach to Design, Implementation, and Management, Third Edition, Thomas Connolly – Carolyn Begg. ISBN 0-201-70857-4

Web Sites

[3]"http://www.directionsmag.com/article.php?article_id=394", "A Brief History of LBS and How OpenLS Fits Into the New Value Chain – Articles"

[4]"http://www.mobilemastinfo.com/information/jargon_buster.htm", "Mobile Phone Mast Jargon - 3G, GSM, MTAG, NRPB Limits, SAR"

[5] "http://gsmserver.com/articles/gsm_overview.php", "What is GSM. GSM Overview. GSM Technology"

[6] "http://en.wikipedia.org/wiki/Cellular_network Cellular network", "Wikipedia, the free encyclopedia"

[7]"http://www.raddcomm.com/E-911%20overview.htm", "E-911 Overview"

[8] "http://www.networktutorials.info/mobile_technology.html", "Mobile Technologies Learning - Mobile Communication Introduction"

[9] "http://en.wikipedia.org/wiki/Mobile_telecommunications", "Mobile telecommunications - Wikipedia, the free encyclopedia"

[10] "http://www.scribd.com/doc/243325/Locationbased-services-History", "Location-based services History"

[11] "http://developer.openwave.com/omdtdocs/location_studio_sdk/pdf/Intro_to_Location_Technologies.pdf, Intro_to_Location_Technologies.pdf", "Location-based service - Wikipedia, the free encyclopedia"

[12] "http://www.geo.unizh.ch/publications/cartouche/ lbs_lecturenotes_steinigeretal2006.pdf","lbs_lecturenotes_steinigeretal20

[13] http://www.techcrunch.com/2008/06/04/location-technologies-primer/, "Location Technologies Primer"

[14] "http://www.technocom-wireless.com/pdf/COMDEX_Fall2001.pdf", "COMDEX_Fall2001.pdf

[15] "http://www.phonescoop.com/glossary/term.php?gid=126", "AFLT definition"

[16] "http://en.wikipedia.org/wiki/Web_service", "Web service - Wikipedia, the free encyclopedia"

[17] "http://www.w3schools.com/webservices/ws_intro.asp", "Web Services Introduction - What is Web Services"

[18] "http://www.w3schools.com/webservices/ws_why.asp", "Why Web Services"

[19] "http://www.w3schools.com/webservices/ws_platform.asp", "Web Services Platform Elements"

[20] "http://www.developer.com/services/article.php/1485821", "Introduction to Web Services"

[21] "http://gdp.globus.org/gt4-tutorial/multiplehtml/ch01s02.html", "1.2. A short introduction to Web Services"

[22] "http://www.locationet.com/LBS_app/intro.shtml", "Locationet - LBS Applications"

[23] "http://www.gisdevelopment.net/technology/lbs/mwf09_vinit.htm", "Building POI LBS and its Indian Market Potential"

[24] "http://bdnooz.com/lbsn-location-based-social-networking-links/", "A list of Location Based Social Networking sites | Location Based Services Business Only"

[25] "http://www.gisdevelopment.net/magazine/global/2008/june/60.htm", "Location aware APPLICATIONS"