Hochschule Wismar
Fakultät für Ingenieurwissenschaften
Bereich Elektrotechnik und Informatik

Exam Paper Internship at M2Mobi

ENTWURF UND IMPLEMENTIERUNG EINER WEB-SCHNITTSTELLE FÜR MOBILE INFORMATIONSANWENDUNGEN.

DESIGN AND IMPLEMENTATION OF A WEB INTERFACE FOR MOBILE IT-APPLICATIONS

Handed in on:

by: Christian Kehl
born on 6th December 1986
in Schwerin

Task Description

Design and Implementation of a Web Interface for mobile IT-Applications.

Aufgabenstellung

Entwurf und Implementierung einer Web-Schnittstelle für mobile Informationsanwendungen.

Abstract

Currently mobile devices are steadily becoming more and more advanced. At the same time penetration (in western countries) of these advanced devices is reaching more than one device per person, creating a big market for mobile applications and services. Although mobile phone manufacturers recently are seeing a decline in sales of basic phones, the market for smart phones, like Blackberries for Business people or the iPhone produced by Apple is steadily increasing.

One of the companies trying to profit from this shift in the mobile landscape is M2Mobi. The company was founded in September 2006, led by Michiel Munneke and Michiel Baneke. M2Mobi is specialized in software development for all kinds of mobile phones. One department in the company is dedicated specifically to developing web interfaces for mobile phones. The biggest challenge this department faces is dealing with all the different devices that are currently used.

The core product of M2Mobi is Nulaz, which is a platform for finding out what is happening around you. The main pillars of the platform were a J2ME mobile client and a web page aimed at regular PCs. There was also a mobile site, but it was very basic. Together with the rest of the mobile web team, it was decided not to continue with that old mobile site, but start from scratch. That way it was possible to properly outline requirements and specifications, and produce well documented and manageable code. After thorough testing, the new site was put live and supporting a wide variety of handsets.

Nulaz is a location-based social network. People are using Nulaz to see where friends and favorite locations are. Additionally, Nulaz is a content distributor. Several location-based RSS feeds are stored in the backend to provide users with information about restaurants, cinemas, sights, general information and much more in the area the user is. Consequently, it gives an optimal overview of what is happening around the user. All this content is visualized in a map, which is the main aspect of the application.

Table of Contents

Definitions

Software

Computer software, or just Software is a general term used to describe the role that computer programs, procedures and documentation play in a computer system. […]

Software includes things such as websites, programs or video games, that are coded by programming languages like C or C++.

"Software" is sometimes used in a broader context to mean anything which is not hardware but which is used with hardware, such as film, tapes and records.

Application Software

Application software is a program that functions and is operated by means of a computer, with the purpose of supporting or improving the software user's work. In other words, it is the subclass of computer software that employs the capabilities of a computer directly and thoroughly to a task that the user wishes to perform. […]

Interface

Interface generally refers to an abstraction that an entity provides of itself to the outside. This separates the methods of external communication from internal operation, and allows it to be internally modified without affecting the way outside entities interact with it, as well as provide multiple abstractions of itself. It may also provide a means of translation between entities which do not speak the same language, such as between a human and a computer. Because interfaces are a form of indirection, some additional overhead is incurred versus direct communication. […]

Paradigm

[…] From the 1960s, the word has referred to thought pattern in any scientific discipline or other epistemological context. The Merriam-Webster Online dictionary defines this usage as "a philosophical and theoretical framework of a scientific school or discipline within which theories, laws, and generalizations and the experiments performed in support of them are formulated; *broadly*: a philosophical or theoretical framework of any kind.

API (Application programming interface)

In computer science, an application programming interface (API) is an interface defining the ways by which an application program may request services from libraries and/or operating systems. An API determines the vocabulary and calling conventions the programmer should employ to use the services. It may include specifications for routines, data structures, object classes and protocols used to communicate between the requesting software and the library.[1]

[1] Definitions taken from the English wikipedia articles at: http://en.wikipedia.org/

1. Introduction

This document is about the development of a mobile web interface for the location-based social network Nulaz. The practical tasks accomplished besides this document are six of the eight stages of the software development lifecycle (Planning, Specification, Design, Implementation, Testing and Deploy).

The following paragraph presenting the Nulaz system, which is the basic system the mobile interface was developed during the internship. Afterwards, another paragraph is introducing issues and specialties regarding the field of mobile web development.

1.1. Nulaz

The main product of M2Mobi is Nulaz. Nulaz is a location-based social network. Social networks are part of modern life in various kinds of fields. Social networks are web communities where people are participating to share events of their every-day life. Most of the big social networks have their own peer group. For example, there are existing Hyves, StudiVZ and Facebook targeting students who want to share their experience of parties and exhibitions. Flickr and Panoramio, for example, are communities where passionate photographers share their pictures. But there are also more business-people oriented social communities like Twitter or LinkedIn. All these examples share the fact that there users communicate about one specific topic that all of them concerns.

Nulaz is more than this. Nulaz is like a Meta social community because people can join Nulaz out of a variety of other social communities and connect there account with the Nulaz account. They can import and export the information that is part of the other community (for instance photos for flickr, audition & exhibition posts for Twitter etc.). This is particularly useful if one user has accounts in several social communities. In this case, the user only needs Nulaz to update information at several other communities at once.

Furthermore, Nulaz is a location-based social community. Users can interact with the system wherever they want. Nulaz also exists in the form of a classic web interface, but the main focus of Nulaz is on the users of mobile phones. They can use the mobile web interface or the phone application corresponding to their phone. M2Mobi provides their Nulaz-users with phone application for every phone: Symbian-based phones, Apple iPhone, Google Android and Windows Mobile.

The main advantage and feature of Nulaz is that every user has a location. Because of the location-based character, every user can be visualized on a map. But Nulaz is more: Nulaz is also a content provider. Nulaz shows not only users, but bars, restaurant, cinemas and much more interesting locations and content is displayed on the map to show the user what's happening around him.

Each Nulaz-user has its own profile. The profile contains information about gender, age, spare-time activities and all the features that are standard for each social community. Every user also has either a public profile picture or a pre-defined profile picture, if the user wants to keep this private. These public pictures are called avatar.

Additionally, each user has a photo gallery where he can upload photos and pictures. This especially is an essential feature for people connection their Flickr-account with Nulaz.

1.2. Issues and Specialities of mobile Development

The development of web interfaces for mobile devices differs to some extent from the development of web interfaces for standard devices. Some issues occurring only, or with much more importance, in development for mobile devices have to be taken into consideration. Besides this, there are also some special features that are only available for this kind of software development. These features are the main motivation behind mobile development.

Development of software for mobile devices takes advantage of the fact that the software is running on an easy moveable device. Most users are taking their mobile phone wherever they are going. Therefore, the mobile device can easily be used for localization tasks. This can be of great use for instance on festivals, to see where friends are, in foreign cities, to localize sights to visit, or in rather sparsely populated areas or the mountains, to find lost people. The task of localization can be accomplished with a variety of technologies.

Another positive aspect of mobile phones is that people can almost immediately share special moments. Modern mobile phones have integrated camera, being able to make pictures of where the user of the mobile phone actually is. Most cutting-edge-technology phones have enough memory space to even save whole videos. These media can be shared over mobile software.

In contrast to these unique feature making it very exciting working with mobile phones, there are also some issues that have to be taken into account when developing mobile software and mobile web interfaces. One, rather obvious issue, in mobile development is the small resolution of mobile phones, compared to other devices like netbooks, laptops or standard devices (Personal Computer, Apple Macintosh). Software has to be designed particularly for mobile phones to look nice and provide a good usability.

Another issue important for mobile web interfaces is the lack of standards in the market of mobile browsers. This issue particularly affects the field of the modern design technology CSS (Cascading Stylesheets). CSS exists in different versions. Although CSS is based on the basic internet meta description language and interchange format XML, therefore providing a relatively easy way to implement, only a few browsers of mobile devices support more than CSS 1.0. The design possibilities with CSS 1.0 are very limited, for instance supporting changing colors for font background.

Concerning design for mobile web interfaces, there is another issue to be aware of. Because of the limited internal memory of mobile phones, the available choice of colors is limited to some web colors of a color table. Only very few phones really support the full RGB color spectrum. This is a result of the lower technology used in that phones, because most of the low-end phones screens don’t support 24-bit-depth due to manufacturing costs.

The reason for using mobile web interfaces for older phones (low end phones) than developing Java applications is that these applications are often running with the same code only for a very few phones. There is a huge gap between low-end phones and high-end phones. The N97 has 128MB available heap size, full touch screen with a resolution of 360x640, the Samsung E250 has 300KB with a resolution of 128x160. It is just no longer justified (and also impossible) to invest time to try to develop something that uses the capabilities of the N97 but also runs well on the E250. Making a web interface for the E250 is a lot quicker and less maintenance than making a stripped down fully interactive J2ME client (Baneke, 2009). The application development for this kind of phones takes much more time (not standardized support for floating point operations etc.) and it is doubtful if that is worth it (because most of them will disappear from the market in the future).

Examples for phones of the target phone group are:

- LG Chocolate KG800
- Sony Ericsson W200i
- Motorola L6
- Samsung SGH-M200
- Nokia 6320i Classic
- Samsung E250

1.3. Structure of the main part – the software life cycle

In order to get the best output out the main project, the practical work of the project has been aligned to the standard procedure of the standard software life cycle (Balzert, Lehrbuch Software-Engineering). The procedure of the software life cycle is an approved principle in software development. While keeping the software development to the life cycle, an optimal output of the project can be provided. This does not only refer to the program, which is the usable output of the project. Moreover, it also provides clear steps to create documentation for maintenance aspects. Additionally, it provides also time for cost- and time estimation to easy up the software management so that client-side deadlines can be met.

Picture 1 software life cycle

Software Knowledge Requirements

260

The topic of this exam paper is the development of web software for mobile devices like the iPhone, Blackberries and many other mobile phones. The main focus is on the support of low-end phones mentioned earlier.

The development of web applications for mobile devices differs in several points from the web application development for Notebooks or Personal Computers. An important issue is the difference in available bandwidth. Only a few mobile devices are able to access Wireless LANs to connect to the Internet. As will be explained in more detail later, most phones access the web with far less bandwidth than is available to normal broadband devices. Therefore, more modern web features such as streaming videos and sound provide by technologies like SVG, AJAX, Adobe Flash or Silverlight cannot (yet) be used on current day mobile devices.

Due to the extremely large amount of different handsets on the market, the web software cannot possibly be tested on all phones. Therefore, to ensure that the website is able to be rendered and shown correctly, the HTML and CSS need to be validated.

Regarding the maintenance of the project in the future, the whole development process needs to be well-documented. The software design should be clear and easy to understand as well as be maintainable. That is why an object-oriented approach was chosen using UML to design the project as well as provide documentation. This is explained later in detail.

Although the result of this project is supposed to completely replace the old mobile site, it still needs to be integrated in a much larger existing IT-infrastructure. The back-end of all the web and server projects is based on the object-oriented Open Source PHP web framework CodeIgniter.

The rest of this chapter will provide more in-depth descriptions of the technologies used during development and implementation.

1.4. The CodeIgniter-Framework

This paragraph gives information about the object-oriented framework “CodeIgniter” which is used in most projects of M2Mobi.

CodeIgniter is an object-oriented Framework, based on the server-side scripting language PHP. It provides a framework for working with the MVC-Paradigm. There are classes for models, views, controllers, helpers and libraries as well as easy support for the implementation of own classes.

CodeIgniter is the central tool in the whole development of Nulaz and its derivates. Because of the server-client character of the projects, the use of a server-side language like PHP is essential in the development of this kind of software. Currently CodeIgniter uses PHP-Version 5.2.9. According to the W3C specification, the new PHP-version 5.3 should be distributed in the first quarter of 2009. At the University of Wismar, in the course of “Web-Programmierung”, PHP is taught using version 5.0, which is currently installed at the Hermes-Server. The main differences between the PHP which is taught in the course and the one used by CodeIgniter are the various new functions, which dominantly affect the structure and possibilities of the software developed. The version used in the lecture and practice part is normally used for programs with an SADT approach. In contrast to this, the version used with CodeIgniter expands the possibilities of PHP to use the better approach of Object-Oriented Analysis and Design.

The following paragraphs will give a general introduction in to the MVC paradigm and the, object-oriented techniques used by CodeIgniter. (CodeIgniter)

1.4.1. The MVC Paradigm

Using the MVC paradigm, the software will be structured in 3 main classes, and 2 helping classes. Model, View and Controller are the main functionality classes while being supported by Helpers and Libraries, the 2 supporting classes.

The idea behind this structure is that the 3 main classes are a collection of classes which work on coherent fields of tasks. The Model inside the software executes interface tasks for data access. The View performs tasks to display the particular selected data sets. The Controller is the interface between Model and View, calling the Model for specific data sets, manipulating them and sending the result to the particular view. Helpers and Libraries contain functions for general data manipulation. Helpers contain functions for manipulating one field (e.g. a helper for computer vision), while Libraries are collections of functions that work on various independent tasks (e.g. libraries with sort functions).

Model

The Model is the connection to the data; it manages the information saved in files as well as information stored in databases. In order to perform these tasks, the model-class provides the following functions:

- access single, specific storage devices
- adding data sets
- manipulating data sets
- retrieving data sets
- erasing data sets
- manage data transfer
- data synchronization
- manage data consistency

View

The View is concerned with displaying specific data sets and information. Each View is defined for a particular output device and associated with a specific use case (regarding the software design). It includes exact Design- and Layout information for the data, which are transferred from the controller to the view. The view architecture can also be used to differentiate between screen- and print media. For the projects at M2Mobi, the view architecture is used for an adequate display of data in the various classes of mobile phones.

Controller

The Controller is the interface between model and view. Input is sent to the controller, which then interacts with the corresponding model to retrieve the necessary data. When the data is transferred back to the controller, it modifies the data for output (e.g. selection of an

adequate data structure). Afterwards, the corresponding view is selected and the data is sent to it by the controller.

1.4.2. Object-oriented Analysis, Design and Implementation

The object-oriented paradigm is a principle for analysis, design and implementation of software. It covers several techniques such as structure and presentation diagrams that help during the specification, design, implementation and testing phases. Based on analysis of the real world, essential properties and behavior (data and functions) of single actors (named as "Objects") is identified and modeled. In conclusion, objects are modeled, and described by data and functions. In this process, homogenous objects are defined in one class. Consequently, they have the same properties and functions, but they can be differentiated by value of their member variables.

PHP as a programming language is more suitable to be used together with the "Structured Analysis and Design Technique" (SADT). According to SADT the value of variables is changed and used by sequentially calling functions. Data is stored independently from the functions.

During the evolution of the PHP programming language, extensions have been introduced which make it possible to form classes to bind data and corresponding functions. These extensions can be seen in the documentation of the present PHP version 5.2.

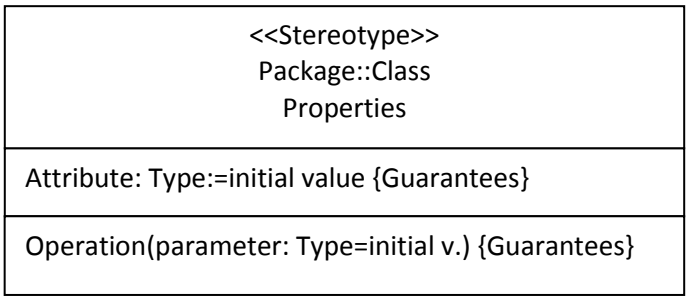
1.4.3. UML Standard in the CodeIgniter Framework

The CodeIgniter Framework uses the extensions of PHP 5.2, mentioned in the previous paragraph, to comply with object-oriented specifications. These extensions hold the main force of CodeIgniter. Because of the easy extendibility and the numbers of useful libraries and helpers, CodeIgniter was chosen from M2Mobi as their Backend solution for Nulaz. Another plus of CodeIgniter is the separation in Model, View and Controller. This makes it easier to provide different interfaces for different devices. In the following, it will be focused in which extent CodeIgniter covers particular Object-oriented specifications. Additionally, it will be pointed out in which form these specifications can be found in the running framework.

UML Base Model implementation:

The idea of the UML base model is the abstraction of the real world to create classes.

A class can consist of the following information:



Picture 2: General visualization of an UML class

With CodeIgniter, stereotypes, as well as a package connection, can be realized. The facility "abstract" of the UML class properties cannot be realized by CodeIgniter. In UML an abstract class has at least one operation where only the header is given and the parameters are only defined by data types. PHP doesn't have abstract variables or classes because variables don't have a predefined data type. The data type is a result of the stored value at a particular time. Therefore, an integer value can be loaded into a variable which before is stored as a string without crashing the program. The data type is adapted automatically. As a consequence of this, parameters don't have a defined type and abstract classes are not possible.

Member variables as well as member functions can be declared in CodeIgniter. Therefore, member variables are created in the constructor of a class. There are no other attributes possible because classes in general are implemented as function calls by CodeIgniter. Member functions are implemented as in every other object-oriented language, and initial values can be set. Guarantees exist to assure that certain pre-and post-operation states are set. Because these guarantees are only based on setting variables in certain states, and calling security functions, this can be done very easily in CodeIgniter.

The actual usage of classes, the creation of objects, works far different in PHP CodeIgniter than in other programming languages. This is based on the fact that there are no specific data types. Therefore, it is not possible to create a variable with a class as data type. Therefore, there always exists a variable storing the particular data of an object as an array or record, and also a function-based class where functions can be called. These functions need the variable as parameter and return a value based on this variable, which needs to be stored.

UML Static Model implementation:

One part of the UML static model is inheritance. This is the strong point of CodeIgniter. All classes in CodeIgniter are sub-classes of the core class "CI" or one of its inheritors. Therefore, the function of a class and its functionality is predominantly defined based on

what parent class it is inherited from. The main disadvantage of this technique is that the complexity of the inheritance structure rises exponentially with the addition of new classes. Another disadvantage is that with deeper inheritance, more unused code for subclasses is transferred.

Associations are possible in CodeIgniter. They are implemented in such a way that an object (the variable) stores an array of objects of other classes, which in turn are arrays/records of data by themselves. So, multiple associations are arrays of arrays/records of data. Because aggregations are associations modeling a part-whole-relationship, they work in the same way as normal associations.

Subsystems can be built through a directory hierarchy. These are easily recognizable in the browser due to the different URI segments.

1.5. Backend-Information for GoogleMaps

In order to use Google Maps as location information service, a Google Maps API key has to be retrieved first. Therefore, the possession of a Gmail/GoogleMail-Account is essential, because the API-key is bound to this Account.

Google Maps is a free Web-based API; every private person or institution can use the API. The API can be used for geo-coding addresses to latitude and longitude coordinates, and the other way around. Because the setup cost and maintenance of this system is a big expense for Google, the number of requests done by a single IP is limited to 15000 requests per day. This number is bound to the IP address of a single user, so the institution running the service is not affected by this. Because of the pretty high traffic consumed by sending the tiles of a map and because the use of the APIs is free, Google has set a limit to the number of requests. The number of request that can be done from one IP is limited to 500000 requests. If the running institution or company estimates more than 500000 requests per day, it is required to provide Google with this estimation. If being approved, Google might provide enough bandwidth and server capacity for an account. (Google)

Besides these restrictions, it is recommended to store the data sets of geo-codes in a privately owned data base. An advantage of this is that additional information can be stored attached to a place. The second advantage is that, if the same address is requested multiple times, data can be retrieved much faster. One geo-code request takes on average 30 milliseconds. A file or database request on the private server should be much quicker. (Roman, 2009)

In order to provide user with complex geographical information, the data format ".kml" is used to store these information. This data format is an xml-based data format for Geo-

information. It is used by the GoogleMaps API, as a data exchange format of some GPS transmitters, and connection format of news feeds and Geo-information (called GeoRSS).

1.6. XML Basics for RSS feeds and KML

New web technologies like Geo-Information or RSS news feeds are based on data structures which are usually inheritors of XML. XML is an open meta-description language where complex and nested data can be stored efficiently. Using XML as a language for new technologies in websites requires a valid XML doctype or namespace. Doctypes (files with a ".dtd"-extension) and namespaces are *rules* how the specific XML files has to be structured.

For preventing to save separate doctype files for each different technology, namespaces are the focus in new web technologies. Therefore, these namespaces are referred to by an URI at the top of an XML file. If a new technology is officially verified by the W3C, it can be updated.

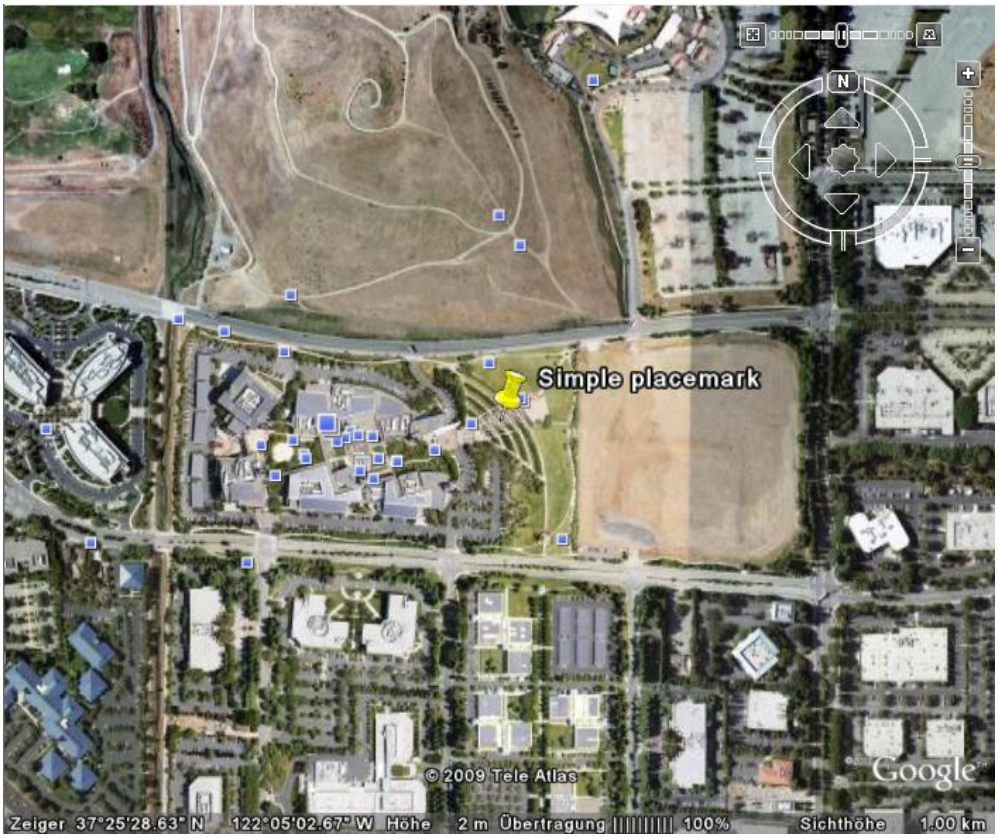
An example for the namespace-driven XML is the KML-format of Geo-information. In the following picture, an example is shown:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Simple placemark</name>
    <description>Attached to the ground. Intelligently places itself
       at the height of the underlying terrain.</description>
    <Point>
      <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

Picture 3: Code example of a KMLfile

The first two lines describe the XML-version used, character encoding in the file and the namespace used for the XML. The namespace used is the KML of the OpenGIS-Board (Geo Information System), which describes the XML structure of geographical XML files. The lines following contain the nested XML information. They describe a place on earth using some meta-data (name, short description) and the main geographical information, the position on the map.

The geographical output in a GIS (for this example "Google Earth Pro") looks like this:



Picture 4: Visualization of the KML file in GoogleEarth

The second XML-technology is an RSS feed. For RSS feeds, only a specific RSS version is required to build the namespaces so that the XML file is valid.

An example is given in the following picture:

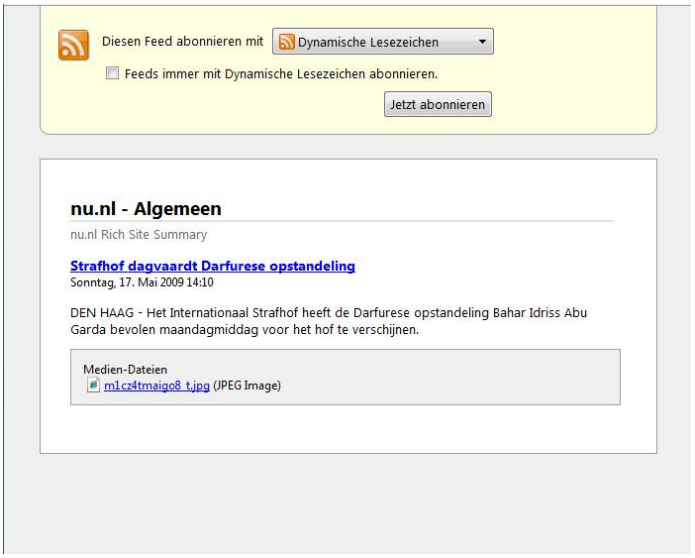
```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
  <channel>
    <title>nu.nl - Algemeen</title>
    <copyright>Copyright (c) 2009, nu.nl</copyright>
    <link>http://www.nu.nl/algemeen/</link>
    <language>nl</language>
    <description>nu.nl Rich Site Summary</description>
    <pubDate>Sun, 17 May 2009 14:10:23 +0200</pubDate>
    <item>
        <title>Strafhof dagvaardt Darfurese opstandeling</title>
        <link>http://www.nu.nl/algemeen/1965444/strafhof-dagvaardt-darfurese-
opstandeling.html</link>
        <guid>http://www.nu.nl/algemeen/1965444/index.html</guid>
        <description>DEN HAAG - Het Internationaal Strafhof heeft de Darfurese opstandeling
Bahar Idriss Abu Garda bevolen maandagmiddag voor het hof te verschijnen.</description>
        <pubDate>Sun, 17 May 2009 14:10:15 +0200</pubDate>
        <category>Algemeen</category>
                <enclosure url="http://media.nu.nl/m/m1cz4tmaigo8_t.jpg" type="image/jpeg"
/>
   </item>
  </channel>
 </rss>
```

Picture 5: Code example of a RSS feed

By looking at this example, it can be seen that the single version number, like in the original xml-heading, is enough to build the feed.

480 Lines three to the end describe the news feed itself. Therefore, the nesting indicates what is general information of the whole news feed (title, copyright, link, language, description, publication date) and what is the specific news item with its information.

The example output can be seen with a browser. This example is show in Mozilla Firefox 3.1:



Picture 6: Visualization of the RSS feed example

These XML technologies can also be combined to make a Geo-information based news feed: GeoRSS.

Nulaz, a location based social community site, has the functionality of a Geo-information based news distributor, using various kind of news feeds to display them in Google Maps.

2. General Solutions

Planning Phase

During the planning phase, the project requirements are specified. First it was necessary to analyze the state of the existing software. Wishes and Requirements of M2Mobi as well as their clients were collected to lead to a final vision of the new software.

In the following chapters, each step to this final vision is presented to give a detailed impression of the start of this project.

500

2.1. The present state

The present state describes the status of the mobile web page before this project. This last version was heavily focused on the requirements of one of M2Mobi's main costumers: Hier.nl. The project had very short specifications which only roughly detailed enough to base the software on. The main objective was to keep the costumer satisfied. Afterwards, problems started to reveal themselves. On the one hand, the software was not completely finished and never fully tested, so each day, bug reports came to the developers. On the other hand, the lack of specification gave too much space to the client to ask for implementation of new features without paying for them.

These issues also lead to some of the main requirements in the specification to avoid them from happening again.

2.1.1. Overview of the previous application

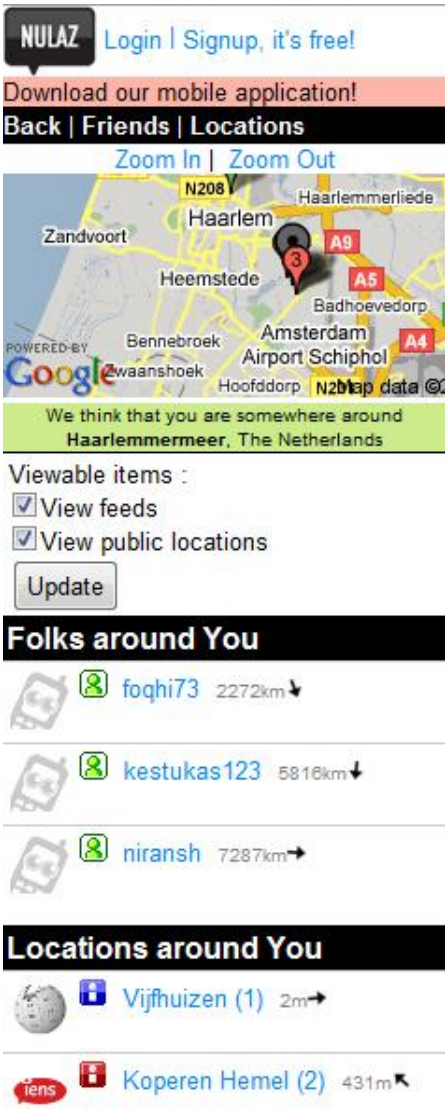
In this chapter, a short overview will be given of the old web page using screenshots of the web page and highlighting points which needed specific attention while determining the new requirements and why this was necessary.

A user visiting the site for the first time, as well as a user who was already registered, was directed to the same home page.

The worst aspect of this page, which led to the most important improvement for the new site, was the immediate loading of the map. This map was constructed via Google StaticMaps-Functionality. The download of this image had the most impact on download time and traffic. It was required to search for a fast alternative without leaving out the map completely.

A second point which was necessary to improve was the display of items. When visiting the website, it was attempted to determine the user's location via several technologies (GPS, Loki, Cell ID and IP). Because of several problems, which will be explained in detail below, an alternative was required.

Another point which is always under discussion for new versions is the introduction of new features into the software. All of Nulaz (available for web users and the mobile Java-Application) has much more features and as such is much more complex than the old site. So, decisions had to be made about what to implement according to the new aims of the software.

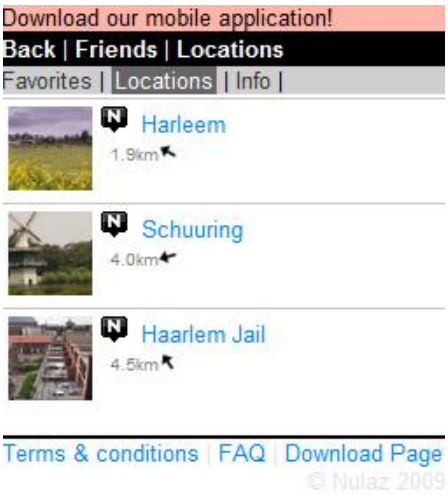


Picture 7: Homepage users strike when visiting the site via mobile phone

The second screenshot shows a listing of all nearby locations.

The biggest disadvantage on this page, as will be explained in detail later, is the loading of avatar pictures. Although a very good technology is used to prevent loading too many items, the page is slowed down remarkably by loading of avatars for items.

The final screenshot displays a detailed profile page of a user. On this page many improvements could be made in loading time and data traffic: A far too big avatar picture, the immediate loading of the map and an avatar list of friends.



Picture 8: the listing of nearby locations. Listings for users and feeds are looking pretty the same.

Another side issue was a slight change of the design, because many people disliked the rather pink background of options and error messages.

Picture 9: Example of a detailed profile page

560 2.1.2. Advantages & Disadvantages

There have been, besides the issues presented above, also some good parts of the former application. The general aim is to keep the advantages of previous versions while resolving issues and improving on the weak points of these former versions, as well as implementing new features.

To get a clear overview of what to keep, what to mark as deprecated and what to improve, the following table has been made.

Advantages	Disadvantages
clear, optical structure	optical difference between mobile and normal website
easy usability	long loading time and much traffic
usage of all available feed	optical overloaded
map feature	display of rather unaccurate information
extendability	disliked color sceme

Table 1: Advantages and Disadvantages of older version

According to this table, it might seem hard to meet all these expectations. With the help of the “devil square” (Balzert, Lehrbuch für Software-Management), it can be determined that the points of performance, layout and extendibility all together are rather contradictory aims. During development, focus was on getting a good performance and a nice layout (within the given limits) manually. Focus was not primarily on implementing new ways to facilitate better extendibility because CodeIgniter was already chosen to be used as a

Framework. The MVC-architecture itself serves as a good way to keep code maintainable and extendable.

2.1.3. **Aim for the new version**

Concerning the points which had been focused on before, the most important task was to cut down the data usage of the mobile webpage. It was far too much because the primary target-group of the mobile website is the group of low-end phone users with a slow internet connection.

There are existing versions of the mobile J2ME-Application for high-end phones, iPhone, Windows Mobile and Android phones. Because of the J2ME-Application has the full functionality of Nulaz, high-end phone user will be encouraged to use the Java-Application in stead of the mobile site.

In contrast to this, low-end phones might not have the capabilities needed for a mobile version of the J2ME Application. Another point that has to be taken into consideration is that these low-end phones have a bandwidth that is comparable to deprecated 56kbit/s Modem technology. In order to give the user acceptable access times to the site, the new mobile page should take not much more than 10kb (in average). Two steps to achieve this are the separation of the map and the reduction of avatar images. All images should be smaller then 50x50 pixels.

Besides implementing as many new features into the mobile website and separating some functionality, the website also had to provide a new clear way of navigation. This navigation should be clearly separated from the main content, and easily accessible no matter what the size and resolution of the display. Another important requirement for the navigation was the ability of the user to be able to easily determine, on all pages, where the user is at the moment.

The final issue was the new handling of user localization. As mentioned earlier, there are 4 types of localization: GPS, Cell-ID, Loki and IP.

GPS (Global Positioning System) is a satellite-based localization technology implemented in some high-end phones. GPS works by sending radio signals to 3 different GPS satellites. Via triangulation, the position is then determined, packed into a data structure and sent back to the mobile phone. Triangulation is a method of determining the relative positions of objects using the geometry of triangles [Trimble website, navigation point “How does GPS work”]. In the mobile webpage, GPS can’t be used because of the loss of access with PHP to hardware (like GPS transmitter) and because the main target phone group has no GPS.

Loki is a cross-platform browser plug-in that allows developers to easily add accurate location to a website with a few lines of JavaScript (Inc.). According to the mobile Nulaz-page, Loki can't be used. On the one hand-side, Loki is focused for desktop browsers. Browsers of mobile phones are not among their target audience. On the other hand-side, the target phone group does not support JavaScript; therefore Loki-usage is impossible.

Cell-ID is basing on terrestrial broadcast stations. Several Base transceiver stations (short: BTS) cover an area. The area is divided so that each BTS covers 3 separated areas (Cells). Each phone connecting via a BTS is automatically localized at the position of the BTS. The accuracy varies, depending on the country the user is. Most detailed Cell-ID networks are in the Netherlands, Germany, the United Kingdom and Finland. (Vitaletti, 2004). Concerning the Nulaz website, Cell-ID is only used by one of M2Mobi's costumers. It's not used in the webpage.

The last localization technique left for the project is IP-Localization. IP-Localization works like this: Every mobile phone entering the World Wide Web gets an IP-address, the sub-information of the address can be used to track where the device that is entering the site, is localized. One big disadvantage of IP-Localization is that this is the least accurate technology of localization. Under the best of circumstances, it is only possible to get the country of the user. It can also happen that a user is changing their position while staying in the same net. In this case, because the IP is not updated with a new position if the connection remains intact, even country level information is incorrect.

In order to deal with these problems, creative solutions had to be found for determining position information and providing localized content based on this position.

2.2. Comparative software

In order to increase the value and stay on top with the software, information has to be retrieved about comparative and concurrency products. The idea is, especially for web software, to get an impression of the cutting-edge user interfaces used on the market. Another point is to perhaps find problems and bad points in these software to avoid making the same mistakes or even solve problems other companies have.

In the following subchapters, most fitting comparative software solutions are presented. Later drafts and parts of the design are based on solutions which have been found here.

2.2.1. M.last.fm

m.last.fm is the mobile solution of the pretty popular web software “Last.fm”. The webpage provides users all over the world with music based on their style of music. It’s an internet radio broadcasting center which determines user preferences and configure then the music choice for every user.

This webpage have been chosen as a comparative because of the nice design as well as for the lightweight of data which is transmitted. The site is accessibly very fast, although there has to be displayed much less content then in Nulaz.

2.2.2. m.facebook.com

m.facebook.com is the mobile solution of the pretty popular web software “Facebook”. Facebook is a web application and, as Nulaz, a social service. Therefore, users log in to the page to connect with other people, share experiences via various kinds of media (pictures, videos etc.) and show there interest to attract other users. The unique factor of Facebook is the user-based advertisement distribution. Website advertisement is based on the user’s location, interests and friends.

This web-application is a good comparative because it’s the same category of web-application as Nulaz. Additionally, Nulaz is connected to Facebook via open Interfaces of Facebook. Structuring, content reduction, usability and navigation are advantages of this page, which also have been taken into consideration for the new version of the mobile Nulaz webpage.

2.2.3. Weather.mobi

Weather.mobi is an only-mobile available weather forecast solution. Weather.mobi gives information about temperature, climate, sun times, humidity and other weather issues.

This web solution has been compared and presented because of the unique structure and navigation on this site. In order weather is one of the most complex issues in science, there are a lot of information to display on a professional weather forecast site. The creators of this webpage have chosen the generally most interesting information in weather and put them in a small “info-screen” on the top of the page. There, the main information are easily accessible. After this, there is displayed the menu, where each sub-category is separated with a big bar to really navigate the user. According to Nulaz, a similar structure has to be found.

2.2.4. **m.nbc.com**

m.nbc.com belongs to the broadcasting company of NBC. This mobile page contains news of several subtopics, media information, sub-information about NBC-produced series etc. Regarding this mass of information that has to be displayed on a mobile phone, m.nbc.com has a deep tree-structured navigation. In order displaying highly-multimedia content, NBC had to think of a good strategy to cope with this mass off traffic. NBC focuses with their mobile page the cutting-edge phones and don't support older phones.

Concerning the new mobile webpage of Nulaz, it has also to be thought about a strategy dealing with images and video, in order the full Nulaz is an equal highly-interactive webpage like m.nbc.com. In comparison to NBC, the mobile page is focused to low-end phones, therefore, there has to be found a different strategy as NBC.

3. Requirements and Aims of the Solution

Specification Phase

In the specification phase, plans are made what things there have to be realized with the new software. Requirements have to be found, set and proven. The requirements are the functionalities of the product. At the end of specification phase, a specification document should include what functionalities should be realized, what functionalities are not realized. This document is the basis of further design and implementation to meet the requirements and ideas that company and costumer have agreed. Another aspect why the specification is important is that this can reduce bugs in later stages of development. Therefore, it can prevent the fail of the software project due to misunderstandings and the loss of detailed description of the aims of the particular product.

Concerning the software project of the new mobile webpage, the specification ware made between the developers and the project manager. In order Nulaz is on one part a software for the anonymous market and on the other part sold to other companies (known as “costumers”), the project manager, being in charge of general business and merchandise, knows best what the users want, what the costumers want and what definitely to improve in comparison to the old version.

The specification has been made in an iterative process. Meetings have set new requirements that have been afterwards set into requirements and written down in the specification document. In the following meeting, these aspects have been discussed and either signed as a fixed requirement or iteratively improved.

An evolutionary prototype (pilot system) has been developed to prove the requirements. This was the basic description of the requirements and another basis, discussions have been made.

In fact this is a maintenance project; the general feasibility has been proven before. Consequently, there was no need for a cost and time estimation in the planning phase, where this is normally done. Instead, after specifying the new software, cost and time estimations have been made for software design, implementation and testing phase.

Regarding the history of the software, the specification was done very carefully in contrast to former times to avoid misunderstandings and to reduce the bug rate in the final system. In former times, the developers got per day several bug reports about the running system, preventing them of working on present projects. In addition, these bugs made users dissatisfied with their platform and costumers partly angry about their product. This has to be improved.

In the following chapters, the single steps of the iteratively-done specification will be presented. For each step, a short introduction will be given of what has been changed in comparison to the former version and, if there is a special reason, why this change has been made. Afterwards, the different stages of time and cost estimation will be shown.

3.1. Mobile Website Specification V1

The first version of the specification was a collection of ideas, based on the problems that have been figured out in the planning phase. The sitemap of this web-project was in this stage nearly the same of the former one.

The biggest difference to the old website is the introduction of the main menu and the options menu. This point was one of the first improvements that have been set as a requirement.

A second big decision that was made even in this early stage was the separation of the map. Instead of showing it on nearly every screen, it has been decided to create links to a specific map-site, where the map is needed.

Concerning the start page and the localization problem, it was decided to show the best rated items of the system for the specific country, the transmitted IP belongs to. In Addition, the user will be able to type in his momentarily address. Based on the accuracy of the input, we show the items sorted by rating (for low accuracy) or sorted by distance (for high accuracy). Consequently, it's possible to get on the one hand-side a rough overview of good locations (if the user doesn't really have a clue what he/she wants) and on the other hand-side a clear instruction where to go (if the user wants to have detailed direction instruction of the area nearby).

The "About Us" page was pretty vacant at this time because the company website only exists as a normal website, not really viewable on a mobile phone.[2]

3.2. Mobile Website Specification V2

The only thing that has been changed in the second version is the point of "About Us". It has been confirmed to start a new project after the new mobile Nulaz website to build up a new mobile company site. It was remarked that the company page, which was online at this time only has a pure web interface without any support for low end phones. Regarding the field

[2] Mobile Website Specification, Annexes

of software development of M2Mobi, a mobile web interface for the company page could improve the general reputation on the market. In advance of this, a link will be provided to inform users about the company with the new site.

Generally, it has to be said that this second meeting was done far too early. Although, most functionality shown in the first version have been set as a requirement, the project manager was not very satisfied with this version. Consequently, the page had to be improved for the final version.[3]

3.3. Mobile Website Specification V3

The third version was also the final version on which contracts with costumers are made.

The first new aspect in comparison to the previous specification was the plan of a versioning. Considering the amount of phones, the focus of this mobile website is first of all on the group of low-end phones. Due to the high technical difference between the mobile phones on the market, there have to be made more versions for better phones. This will be accomplished, on first place, via choice of CSS-Files (Cascading Style Sheets).

A second new feature of the final specification is the change of the home page. In contrast to former specification versions, where items have been shown “in the whole” (the system cannot determine sophisticatedly the user’s position), it was decided to start with a plain home page, forcing the user to type in his position. Basing on this information, which are in most of the cases more accurate than IP-Localization, items in his area are shown. The concept of sorting depending of the accuracy of input stays as an idea for later versions of this software. In the first version, items are, at first, sorted by popularity.

A smaller change, more or less contributed to design issues, is the change from horizontal to vertical menus. This has been done to reduce the space the menu will need on the small displays. Therefore, more content can be shown.

Referring to the second meeting, several smaller options have also been included in the specification. One example is the friend invitation, where the corresponding screen has been included in all versions of the specification, while this one has been overseen by the developers in previous versions. Another example is the functionality to create new locations to provide more interaction with the system for the user.

Main changes have been made regarding the profiles. Having a look on the feature list, it has been decided to introduce two new sections which show the latest activities and comments of a user or a location. With the help of this information, the user should visiting another user’s profile should stay at the profile, because it gives him all the information he wants.

[3] Mobile Website Specification V2, Annexes

This is a very important requirement because character of the application has changed. Referring to the old mobile website, the software was more a map distributor to the user. Due to the rising user numbers, the system gets more and more the character of a traditional social community. With the separation of the map, the new mobile page should also provide the ambience of a traditional social community site.

The second change concerning the profiles was a clear separation of all headlines. Therefore, headings are now getting a big bar as a background.

While having a closer look to the specifications of the map, a general conclusion was that the map will have a key impact in implementation phase. It has been decided to make the requirement that the user as well as the item of the visited profile is shown on the map. Both items should be listed, including the distance, at the bottom of the map. Additionally, a form should be provided to give the user the possibility to center the map at a point he wants.

Another strong aspect where discussions have taken much time was the photo gallery. Having a closer look to the focused phone group (low end phones), the developers wanted to strike out this requirement because internet connections are too low for this. The target phone group neither has a 3G-connection, nor WiFi, Bluetooth or W-LAN. These phones have a standard GPRS connection with a bandwidth of 9 Kbit/s to the maximum of 21 Kbit/s. Therefore, the developers think it is senseless to send for a single thumbnail 4x2 Kbyte to the phone, because the traffic will cost much money to the user and the user will also have to wait a lot longer for this page, in comparison to the rest of the pages, until the site has loaded. On the other hand, the project manager thought to implement this to give interested people the chance to share their experiences with other users. At the end, this feature has been considered not to take out, as it has been the will of the developers, and to leave it as a fixed requirement.

The last pages of the specification also includes the final time and cost estimation. This process will be explained in the following chapters.[4]

3.4. General Use Case Diagram

In conclusion to completely understand the requirements of the new page, a general user case diagram has been created.[5]

[4] Mobile Website Specification V3, Annexes
[5] General Use Case Diagram, Annexes

3.5. Sitemap

Regarding the further steps of software development, a sitemap has been created to give the developers a clear structure which pages have to been accomplished.[6]

3.6. Approach of Empiric Time and Cost Estimation

The process of empiric time and cost estimation is based on the use experience of professionals. One or more professionals get the plans for the software. They give their estimations of time and cost. By using more people in this process, the result can be improved. Therefore, either all professionals estimate the whole software and afterwards the average of these estimations is taken, or each professional estimate a different part of the software, according to their special field. One of these empiric estimation procedures is the Delphi method. It includes the participation of several professionals who discuss together about the approximation. It is named after the working procedure in the ancient oracle of Delphi, where the old wise men sat together to discuss about the fate of people who wanted to have advise in special turning points of their lives.

3.6.1. Delphi method

For the new Nulaz website, the Delphi method was chosen as the final method of estimation. The general procedure works like this: one of the long-experienced developers made the estimation of the implementation time. Afterwards, these estimations have been reviewed by another developer. For each point to develop, a testing difficulty was selected. Out of these difficulties, a testing factor has been introduced to approximate the general testing time while programming. So, at the end, there was a testing time for each point that has to be developed. These have been summarized to get a total implementation time.

The problem according to this approach is that it ignores the fact that M2Mobi consists mainly of interns. Consequently, new developers have been introduced to the project. In addition, the developers had to communicate a lot because of teaching and system introduction issues. Another aspect which complicated that was the fact that the developers had to communicate mainly not in their mother language. Therefore, the total amount of time was taken as a raw number and then has been multiplied by 1.2, as a kind of communication overhead.

[6] Sitemap, Annexes

In Order 2 people are working on the project, this result was at the end divided by 2.[7]

3.6.2. **Testing approximation**

For the final system integration test approximation, a new approach was approved. For every view in the site, there are in average 4 points to test (a right input, the “Back”-option and in average 2 other points). Because bugs will occur during the testing phase, this number is multiplied by 3 (the average repletion time because of bugs). This number, 12, is the raw complexity per view to test. Because there are 21 views to test, the raw total complexity is 264. Basing on the estimation that each feature needs 2 min of testing, the first raw total testing is 528 min (8.8h). This is the time consumption for one test on one phone on the whole mobile interface. After each test run, a test report has to be written, which takes around 1 hour. So, the final time one test will take is 9.8 hours. According to the target to have this website run bug-free on every low-end phone, this test has to be done on every low-end phone. Therefore, this number has to be multiplied by the total number of low-end phones that are available for testing, which are actually 15. It should be mentioned that there are a lot more low-end phones on the market. So even after this testing, it is not verified that the website will run bug-free on every phone. But the testing for these 15 will ensure a bug-free version for the vast majority of phones. Due to this aspect, the whole system integration test will take around 147 hours. Regarding the computation complexity of the approximation, the computation has been written a bit differently in the specification.[8,9]

[7] Implementation Approximation, Annexes
[8] Testing Approximation, Annexes
[9] Mobile Website Specification V3, Annexes

4. Design of the approved Solution

Design Phase

The design phase of the software development is used to form a specific plan what to implement, based on the goals that has been set in the specification.

Due to the usage of the object-oriented framework, the software has been designed with the help of UML diagrams. Because of the increased interactive character of the web software, especially regarding the chat and the map, it was also sometimes necessary to specifically use UML 1.1 diagrams to keep the content of one part in one diagram without decreasing the understandability. This was the case on all communication diagrams.

900

4.1. Use Case Diagrams

The application is controlled via each single sub-page as one view. Consequently, the basic approach was to design a Use Cases as a diagram for each single view. In contrast to this, there are also some very easy views with less interaction. These Use Cases have been designed using a table with the description of entry, exits and interaction possibilities.

For the four newest and most critical parts of the software, also more complicated, detailed diagrams have been developed to get an idea how to implement these parts of the software and, in consequence of this, decrease later the implementation time. Regarding former projects, it can be said that a strong design phase decreases the amount of faults and, with this, also the implementation time.

4.2. Overview Class Diagram

A general use case diagram had to be developed because no diagram was available for the software development. The old structure was the basis and has been extended by the new, needed classes. Concerning the fact that it should give a basic guideline to not forget something and to get the right controller-model links, the diagram don't include any details of the member functions.[10,11]

[10] General Class Diagram Models [Annexes]

4.3. Recent Activities

The feature of the activity feed was the easiest of the new features to include in the new mobile page, because it had been used in the normal web page before.

It was decided consentaneously by all involved developers that this feature should be the first new one to program for the new mobile page. To prevent troubles at the start of the project, it has been decided to create a class, communication and activity diagram for this feature, although the feature is well-known.[12]

4.4. Chats

A chat is generally an asynchronous process. Messages are distributed spontaneously. The chat feature is therefore a pretty tricky thing in point of synchronization, reloading and computation complexity as well as the possible amount of recipients. This is, for a real-time chat, generally also somehow limited by the available bandwidth, because the control overhead rises with amount of members of a chat. This can be seen in comparable software projects like StudiVZ with a web interface for standard devices.

Another problem that hits all aspects mentioned above is the change of technology and programming language, which was obvious from the start of the project on. Generally, chats are developed in programming languages that support asynchronous data transfer. For desktop applications, this concerns every modern programming language like Java, C++, C# and so on. For standard web interfaces, this is approved with the client side language JavaScript and its server side additional technology AJAX. Because these technologies are generally not support on low-end phones and only supported and correctly executed by a few high-end phones, it was for sure that these technologies and possibilities are not available.

To keep at least the impression of a real chat, which is normally a feature with real-time character, it was decided to provide the user with “Refresh Chat”-link so the user can decide himself how often the chat is refreshed. This is then only limited of the connection speed of his mobile phone.

Concluding all these reasons, class, communication and activity diagram has also been designed for this feature.[13]

---

[11] General Class Diagram Controllers [Annexes]
[12] Recent Activities Activity Diagram,
Recent Activities Communication Diagram,
Recent Activities Class Diagram,
Recent Activities Use Case Diagram [Annexes]
[13] Chats Activity Diagram,

4.5. Photo Gallery

The problem of the photo gallery on mobile pages is, as discussed in previous chapters, the low connection speed. Therefore, it has been thought of how to optimize the code to load the pictures with best quality-capacity ratio.

Due to the fact that the photo gallery doesn't use an external service or needs to be developed for real time character, there was no need for a communication diagram. Beside this, class and activity has been designed.[14]

960

4.6. Map

The map was, since beginning of the project, the most important and hardest feature to implement.

One problem is that the map itself is provided by an external service – Google Maps. For non-JavaScript-phones, the only possibility to get a map is via Static Maps, where the ways of configuration and interaction are very limited.

For phones with JavaScript-support, the normal Google map can be used, while being aware of far less connection speed compared to the standard web interface. To speed this up, it has to be decided what to show on the map.

For all the technologies, a basic activity, class and communication diagram is vital for development. Beside this, a diagram visualization of the Geocoder is useful. Due to these idea, it has been designed an activity diagram as well as a state chart diagram for the geocoder.[15]

Chats Communication Diagram,
Chats Class Diagram,
Chats Use Case Diagram [Annexes]

[14] Photo Gallery Class Diagram,
Photo Gallery Use Case Diagram,
Photo Gallery Activity Diagram [Annexes]

[15] Map Class Diagram,
Map Use Case Diagram,
Map Activity Diagram,
Map Communication Diagram,
GeoCoder State Chart Diagram,
GeoCoder Activity Diagram [Annexes]

5. Realization of the Solution

Implementation Phase

5.1. Overview of adapted and new classes

The following tables visualize what classes have been changed or added and, in case of a change, what has been changed generally.

Controller	**Modification**
about	Treatment of sub-points in the FAQ has changed
activity	New
chats	Completely new chat system based only on PHP
errors	New
locations	Modifications for mobile website e.g. “edit_location”, “add_location” etc.
mobile	Re-written
users	Modifications in “get_comments”, “get_friends” and “get_favorite_locations”
Photos	Modifications in uploading pictures and loading smaller pictures for mobile website
Nulaz	New treatment for index page of mobile website
Settings	Added new interaction and modification possibilities for the own profile of a user
Search	New for mobile page
Model	**Modification**
Chat	Set-up for a paginated, PHP-driven chat & inbox
Location	Pagination for comments
user	Pagination for comments

Table 2: Overview of modifications

5.2. Recent Activities

The recent activities exist, as well as the comments, in two different versions. The first version is implemented as an include feature for the corresponding profile, showing the last 3 activities. The second version can be entered via “All”-link, showing a paginated overview of all activities that are related to the particular person or location of the profile.

A special point of the activities is the possibility to show additionally a global overview of all actions that have been done in the system.

This feature is implemented in the following way.

```
{
    // Setting up Pagination

    //page-site (start item number) is transmitted in the third URI segment
    $config['uri_segment'] = 3;
    //base-url on which this pagination is based (function which is called)
    $config['base_url'] = base_url().'activity/get_events';
    //total number of items is counted with count-SQL statement in Database
    //[look in library "events", function count_newsfeed() ]
    $config['total_rows'] = $this->events->count_newsfeed();
    //8 items are displayed per page
    $config['per_page'] = '8';
    //pagination is initialized with former set-up values
    $this->pagination->initialize($config);

    //per page, 8 items are displayed
    $limit = 8;

    //present start item is computed:
    //3rd URI segment is empty: start is 0
    //else: start = number in URI segment 3
    $offset = $this->uri->segment(3) == "" ? 0 : $this->uri->segment(3);

    //items from offset (start) to limit (end) are loaded via library
    $this->data['newsfeed'] = $this->events->get_events($offset, $limit);

    //Setup for View

    //Data is displayed as a single site, NOT as an include of the profiles
    $this->data['singlesite']=TRUE;
    //header is set-up with tanslation system: "Recent activities [link]Back[/link]"
    $this->data['title_text'] = lang('head_activity').' (<a href="'.base_url().'">'.
    //page title for browser is set: "Recent activities"
    $this->data['document_title']=lang('head_activity');
    //mobile website view is loaded
    $this->template->load('includes/newsfeed', $this->data);
}
```

Picture 10 Code example of the general "Recent activities"

The recent activities of a user, shown as a standalone-page, have been implemented in quite the same way. The following lines are the implementation of that.

```php
//determine the user_id of the called loginname
$user_id=$this->user->get_id('login_name',$loginname);
//if user exists (user_id is given back), then set up the recent activities, e
if($user_id)
{
    //determine the username (nickname) of the user
    $username=$this->user->get_info('login_name', $user_id);
    // Setting up Pagination
    //page-site (start item number) is transmitted in the fourth URI segment
    $config['uri_segment'] = 4;
    //base-url on which this pagination is based (function which is called)
    //--> in this case, URL is converted with a routing file
    $config['base_url'] = base_url().'user/'.$loginname.'/activity';
    //total number of items is counted with count-SQL statement in Database
    //[look in library "events", function count_newsfeed($user_id) ]
    $config['total_rows'] = intval($this->events->count_newsfeed($user_id));
    //8 items are displayed per page
    $config['per_page'] = '8';
    //pagination is initialized with former set-up values
    $this->pagination->initialize($config);

    //per page, 8 items are displayed
    $limit = 8;

    //present start item is computed:
    //3rd URI segment is empty: start is 0
    //else: start = number in URI segment 3
    $offset = $this->uri->segment(4) == "" ? 0 : $this->uri->segment(4);

    //Data is displayed as a single site, NOT as an include of the profiles
    $this->data['singlesite']=TRUE;
    //header is set-up with tanslation system: "<user>'s activities"
    $this->data['title_text'] = $username.lang('profiles_activity_title');
    //page title for browser is set: "<user>'s activities"
    $this->data['document_title']=$username.lang('profiles_activity_title');
    //mobile website view is loaded
    $this->template->load('includes/newsfeed', $this->data);
}
```

Picture 11 Code example of "Recent Activities" of a particular user

The first version of the recent activities, as one including part of a profile, is integrated in the function which sets up the data for the profile. It is located in the “users”-controller.

The following code is the corresponding passage of the mentioned controller.

```php
//[...]
// Get newsfeed
//if page is entered from a mobile phone and this is not an iPhone (because iPhone page is seperated
 if($this->agent->is_mobile() && !$this->agent->is_iphone())
 {
        //newsfeed appears as an Include of the profile; activities are not shown as own page
        $this->data['singlesite'] = FALSE;
        //Header of the passage: "<user>'s activities"
        $this->data['title_text'] = $this->data['buddy_name'].lang('profiles_activity_title');
        //load the latest three activities related to the user
        $this->data['newsfeed'] = $this->events->get_events_user($user_id, 0, 3);
}
else
//treatment for normal website
{
        //load the latest eight activities related to the user
        $this->data['newsfeed'] = $this->events->get_events_user($user_id, 0, 8);
}
//[...]
```

Picture 12 Code example of "Recent Activities" in "users"-controller

```php
<?php
//if newsfeed is displayed as a standalone-page, add a headline
if($singlesite==TRUE){ ?>
<h1><?=$title_text;?></h1>
<?php } ?>
<p class="main_content">
    <?php
    //if activities are available, show all transmitted activities
    if(isset($newsfeed) && $newsfeed != false){ ?>
    <ul id="last_activity">
        <?php
            foreach($newsfeed as $event){
                if($event['main_message'] != -1){ ?>
                <div class="item_list">
                    <li class="<?=$event['action'];?>">
                        <div>
                            <?=$event['main_message'];?><br />
                            <small class="type"><?=$event['ago_message'];?></small>
                        </div>
                    </li>
                </div>
        <?php   } } ?>
    </ul>
    <?php
    //if newsfeed is shown as standalone page, add links for the pagination
    if($singlesite==TRUE){ ?>
    <div class="pg_links">
        <? echo $this->pagination->create_links();?>
    </div>
    <?php } ?>
    <?php  }else{
    //display that no activities have been done
    echo "<div>".lang('user_no_activity')."</div>";
    }?>
</p>
```

Picture 13 View of "last activities" include in user profile

5.3. Chat

The chat was done in a three-step-way. This can be seen in the designing diagrams of the chat.

The most important functions have been implemented in the chat controller. As an example for the complexity of the PHP chat, the function of adding new persons to a chat have been chosen as code example and is presented in the following passage.

```
function new_recipient($username="")
//start of a new chat in mobile website -> PHP-driven, no JavaScript
{
    if(!$this->auth->is_logged_in())
    {
        //if user is not logged in, redirect the user to the login page --> Chat o
        redirect('login?return=chats/inbox');
        exit;
    }

    //addition of users is based on the user model
    $this->load->model('user');
    //for veryfication of the form, validation library is used
    $this->load->library('validation');

    //get the user_id out of the session variables
    $user_id=$this->session->userdata('user_id');
    //create a new arry of recipients for later output
    $this->data['recipients']=array();
    //set the number of loaded recipients to 0
    $this->data['number']=0;

    //load buddies for the user's friendlist
    $this->data['friends']=$this->user->get_friends($user_id);
    $userlist=array();

    //if any submission was done, start adding recipients or start chat
    if(($this->input->post('submitted')=='yes') || ($this->input->post('submitted2
'yes') || ($this->session->userdata('profile_chat')))
    {
        //if populated userlist was sent with the post, fill the userlist
        if($this->input->post('userlist'))
        {
            $userlist=explode(",", $this->input->post('userlist'));
        }
    }
```

Picture 14 Code example "New Recipient" to start a new chat - Part 1

```
//if chat was done from a profile, add the recipient to the list
//further information see: "function recipient_from_profile($loginname)"
if($this->session->userdata('profile_chat'))
{
    $new_loginname=$this->user->get_info('login_name', $this->session->u
    array_push($userlist, $new_loginname);
    $this->session->set_userdata('profile_chat',"");
    $this->session->unset_userdata('profile_chat');
}

//perform post-actions, validation
$rules['general_user'] = "trim|required|min_length[3]|max_length[64]";
$this->validation->set_rules($rules);

$fields['general_user'] = lang('chats_user_input_field');
$this->validation->set_fields($fields);
//input field submission of typed in username
if($this->input->post('submitted')=='yes')
{
    if($this->validation->run() == FALSE)
    {
        // Validation not passed.
            $this->errors[] = 'errors_chats_recip_user';
    }
    else
    {
        $new_username=$this->input->post('general_user');

        //if input was empty, give error message
        if(!$new_username || $new_username=="")
        {
            //Error handling for left out input
            $this->errors[]=lang('errors_chats_recip_input');
        }
        else
        {
            $new_id=$this->user->get_id("user_name", $new_username);
```

Picture 15 Code example "New Recipient" to start a new chat - Part 2

```
            //if input was invalid (no existing user), give error message
            if (!$new_id)
            {
                //Error handling if no such user exists
                $this->errors[]=lang('errors_chats_recip_input');
            }
            else
            {
                $new_loginname=$this->user->get_info('login_name', $new_id);
                //check if user has already been chosen for this conversation
                if(in_array($new_loginname,$userlist))
                {
                    //Error handling for adding a user more than once
                    $this->errors[]=lang('errors_chats_recip_double');
                }
                else
                {
                    //add user to recipients array
                    array_push($userlist, $new_loginname);
                }
            }
        }
    }
}

//select-box submission for chosen friends out of the friendlist
if($this->input->post('submitted2')=='yes')
{
    $new_loginname=$this->input->post('friendly_user');
    //check if user has already been chosen for this conversation
    if(in_array($new_loginname, $userlist))
    {
        $this->errors[]=lang('errors_chats_recip_double');
    }
    else
    {
```

Picture 16 Code example "New Recipient" to start a new chat - Part 3

```
        //add user to recipients array
        array_push($userlist, $new_loginname);
    }
}

//Build up recipients list if userlist has entries; if not, set number o
//(to avoid nonsense data)
if(!(empty($userlist)))
{
    $i=0; $l=count($userlist);
    for($i=0; $i<$l; $i++)
    {
        if($userlist[$i]!="")
        {
            //array_push($this->data['recipients'], $userlist[$i]);
            $this->data['recipients'][$i]['loginname']=$userlist[$i];
            $this->data['number']++;
        }
    }
}
else
{
    $this->data['recipients']=array();
    $this->data['number']=0;
}


//start chat submission
if($this->input->post('submitted3')=='yes')
{
    //If no recipients are chosen, display error and do not allow the ch
    if($this->data['number']==0)
    {
        $this->errors[]='chats_recip_no_selection';
        $this->data['document_title'] = lang('head_chats_recip');
        //var_dump($this->data); exit;
        $this->template->load('chats/new_recipients', $this->data);
    }
```

1020 **Picture 17** Code example "New Recipient" to start a new chat - Part 4

```
        $this->template->load('chats/new_recipients', $this->data);
    }
    else
    //If recipients are there, start a chat
    {
        $participant_id=$this->user->get_id("login_name",$this->data['recipi

        //start a chat - first recipient is automatically added
        $chat_id=$this->chat->create($user_id, $participant_id);

        //own account has to be added manually
        $this->chat->add_recipients($chat_id, $user_id);

        //add all other recipients
        $i=0;
        foreach($this->data['recipients'] as $recipient)
        {
            if($i==0)
            {
                $i++;
                continue;
            }
            else
            {
                $participant_id=$this->user->get_id("login_name",$recipient[
                $this->chat->add_recipients($chat_id, $participant_id);
                $i++;
            }
        }
        $this->data['chat_id'] = $chat_id;
        //get the chat
        redirect('chats/'.$chat_id);
        exit;
    }
}
else
//if chat should not start, add more recipients to the future chat
```

Picture 18 Code example "New Recipient" to start a new chat - Part 5

```
    //if chat should not start, add more recipients to the future chat
    {
        if($this->data['number']==0)
        {
            $this->data['recipients']=array();
            $this->data['number']=0;
        }
        $this->data['document_title'] = lang('head_chats_recip');
        $this->template->load('chats/new_recipients', $this->data);
    }
}

//PHP-driven chat deletion, No JavaScript
//needed parameter: Chat ID
//Initial Value: -1 -> "Delete All"
//Other value: particular chat
function delete_chat($chat_id=-1)
{
    if(!$this->auth->is_logged_in())
    {
        redirect('login?return=chats/inbox');
        exit;
    }
    else
    {
        $user_id=$this->session->userdata('user_id');
        //If chat_id=-1 -> "Delete All" -> clear the inbox
        if($chat_id===-1)
        {
            $chats=$this->chat->get_inbox($user_id);
            foreach($chats as $chat)
            {
                $chat_id=$chat['chat_id'];
                $this->chat->leave($user_id, $chat_id);
            }
            redirect('chats/inbox');
```

Picture 19 Code example "New Recipient" to start a new chat - Part 6

5.4. Photo Gallery

In the old version of the mobile interface, pictures have been transmitted to the browser in full size of the uploaded image. To reduce the loading time of the gallery smaller images, called thumbnails, have to be used to increase the loading speed. The possibility of resizing the image via CSS command provides a smaller image output whereas the data size stays the same. Consequently, CSS width and height parameters are no option in compressing the image.

For the creation of thumbnails the CodeIgniter Framework is a helpful tool because it provides a library for image handling such as upload, resize, primitive picture processing methods and thumbnail creation. Therefore, only the code of the controller had to be changed to provide small images (50 pixels by 50 pixels) for mobile phones while the normal website gets the images with its original size.

The code for this choice is represented with the following line out of the photo controller to load the gallery.

```
$this->data['pic_size']='';
if($this->agent->is_mobile() && !$this->agent->is_iphone())
{
    $this->data['pic_size']='sq50';
}
else
{
    $this->data['pic_size']='';
}
```

Picture 20 visualization of image output separation between mobile device and normal web for photos

5.5. Errors

Concerning the handling of 404 Page “page not found” errors, the user can be provided with background information why the error occurred. The corresponding cases are included in the activity diagram for the errors.[16]

The determination of faults coming from the Nulaz page is done by the parameter of the index function. If any content is not reachable because the values are not in the database, the source of this error is, with a very high chance, a broken link on the page leading to non-existing content. Therefore, everywhere when content is not reachable while a get-function is called, the following line of code will be executed.

[16] Errors State Chart Diagram
Errors Activity Diagram

```
if(! <content>)
{
        redirect('errors/index/yes');
        exit;
}
```

Picture 21 Example for internal 404 error detection

This line of code calls the index function in the errors controller with the parameter “yes”. This will give, as it can be seen on the whole controller code, an error message that the user followed an internal broken link and a “Back”-Button to go back to where the user came from.

For the search engine error determination, an array has been created with the most popular search engines in the web. If the parameter is not “yes”, the source of the 404 error is external. Therefore, the “HTTP_REFERER”, a global environment variable, is checked if it contains one of the search engine domains. If this is the case, the search engine has indexed a broken link the user followed. The user will be provided with this information and motivated to stay on the page and sign up an account.

In case the user followed a broken link of a partner site in the web, the URI is saved in the “HTTP_REFERER”. The error can be logged. If this broken error is confirmed manually (those kinds of 404s are relatively rare), the host of the partner site can be informed to update this link. The user is provided with this information and, additionally, with a back link to the page he came from.

The last possibility is that the user entered a wrong address by himself which is routed to the domain of Nulaz. Although it can be determined correctly (the “HTTP_REFERER” is not set), it has been implemented by process of elimination. Consequently, if the reason hasn’t been one of the former explained, it is probably a typing fault leading to the error.

The 404 error handling is done in the following code.

```php
<?php

class Errors extends Nulaz_Controller {

    function Errors()
    {
        parent::Nulaz_Controller();
        $this->load->library('events');
    }

    function index($our_page)
    {

        $search_engines = array(
            "www.google.com",
            "www.altavista.com",
            "www.lycos.com",
            "www.yahoo.com",
            "www.alltheweb.com",
            "www.go2net.com",
            "http://infoseek.go.com/",
            "http://www.looksmart.com/",
            "http://www.hotbot.com/",
            "http://www.mckinley.com/");


        $this->data['document_title']=lang('head_404');
        $this->data['additional']='';
        if($our_page!='yes')
        {
            if($our_page=='no')
            {
                //comes from the server - try the normal errors ;-)
                if(in_array(getenv('HTTP_REFERER'),$search_engines)==TRUE)
                {
                    $this->data['connection_error']='search_page';
                    //provide a message that the error occured by a wrong link of a searchengine
```

Picture 22 Code of "Errors"-controller - Part 1

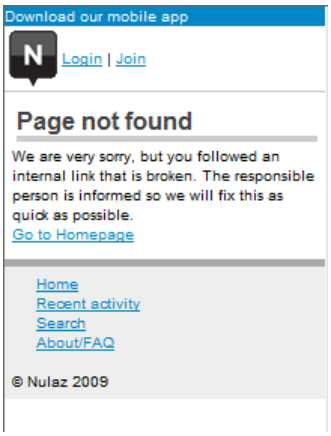
```
                    $this->data['notice']=lang('404_search_notice');
                    //provide a big sign-up button to link the sign-up site
                    $this->data['additional']='<p>'.lang('404_search_additionals').'</p><a hr
<span>'.lang('404_signup_link').'</span></a>');
                }
                if(getenv('HTTP_REFERER'))
                {
                    $this->data['connection_error']='link_page';
                    //provide a message that the error occured by following a broken link of
                    $this->data['notice']=lang('404_link_notice');
                    //inform via mail to Nulaz of the broken external link
                    //no additionals
                    $this->data['additional']='<p>'.lang('404_link_additionals').'<a href="'
'HTTP_REFERER').'</a></p>';
                }
                else
                {
                    //propably users fault
                    $this->data['connection_error']='user_fault';
                    //provide a message that fault is blaming to a wrong typed-in address or
                    $this->data['notice']=lang('404_user_notice');
                    //scan for perhaps meant sites - leave that out for the moment
                    //use $_SERVER["REQUEST_URI"]
                    //provide at max. 5 alternative soultions as additional data
                    // right data - no additionals
                    $this->data['additional']='<p>'.lang('404_user_additionals').'<br />'.la
base_url().'signup" class="action"><span>'.lang('404_signup_link').'</span></a></p>';
                }
            }
        }
        else
        {
            //call came from one of our controllers - wrong data - chance is very high that
            $this->data['connection_error']='own_page';
            //inform the user that he followed a broken link from our site
            $this->data['notice']=lang('404_m2m_notice');
            //inform via mail to Nulaz of broken own Link
```

Picture 23 Code of "Errors"-controller - Part 2

```
            //advice for next step
            $this->data['additional']='<p>'.lang('404_m2m_additionals').'</p>';
        }
        $this->data['newsfeed'] = $this->events->get_events(0,8);
        $this->template->load('errors/404', $this->data);
        //exit;
    }
}

?>
```

Picture 24 Code of "Errors"-controller - Part 3



1080

Picture 25 Visual example of the 404 Page

5.6. Map

Taking a closer view on the implementation of the map, there have been two possibilities to realize the display and interaction with the map. On the one hand there could have been written a variety of slightly different functions because there are only slightly changes in the content to display, depending on the way of entering the map. On the other hand, there could have been made one function to build the map and for each different way of interaction another function. This way has been chosen because the structure is very simple and clear. This is very important for building extensions, although each function itself will have more lines of code. Due to the chosen architecture, it is possible to add a new way of interaction to the interaction function and not touch the display code, and to add a new content to display by extending the display function, without modifying the computation-intensive, very mathematical function for interactions.

Because the interaction of the map is realized by member variables, these variables have to be set in the constructor. Very important are the zoom level as well as the vertical and horizontal extends of the viewport. These parameters of a viewport can be send to Google to get a guaranteed image size. This is needed because some low-end phones resize big images automatically. To prevent this behavior, the viewport is essential. These extends where chosen to be 5/6 of the original width and height of the currently viewed map. The viewport itself is based on geographical degree instead of graphical units. In addition to this issue, the extend values are depending on the present zoom level. That's why, a way of computation had to be developed.

Therefore, this computation has been applied:

$$extend\ value = \frac{\frac{mapwidth/height}{10000} - \frac{mapwidth/height}{10000 * 6}}{2^{zoom\ level}}$$

Referring to this formulae, the map extends scale was determined by experience so that this represents the width and height of the map in degree. For the original movement, this extend value was divided by 2 regarding the positive and negative movement in each direction

Taking all these needed information into account, the constructor of the class which includes the map has to look like this:

```
        //pre-defined values for height and width of viewport - general expands of the map
        if(!($this->session->userdata('zoomlevel')))
        {
            $this->session->set_userdata('zoomlevel',13);
        }
        else
        {
            $this->session->userdata('zoomlevel');
        }
        $zoommod=$this->session->userdata('zoomlevel');
        //$this->data['viewport']['width']=100;
        //$this->data['viewport']['height']=100;
        $this->data['viewport']['width']=$this->session->userdata('resolution_width')-70;
        $this->data['viewport']['height']=$this->session->userdata('resolution_width')-70;
        $this->data['viewport']['vextend']=(($this->data['viewport']['height']/10000)-(($this->data['viewport']['height']/10000)/6)))
/ pow(2,$zoommod) ;
        $this->data['viewport']['hextend']=(($this->data['viewport']['width']/10000)-(($this->data['viewport']['width']/10000)/6))) /
pow(2,$zoommod) ;
        $this->session->set_userdata('current_center',"");
```

Picture 26 Code of map variable initialization

The main display function is too big to be completely presented at this point. It is recommended to take a look at the code on the CD for detailed information.

Generally, the code itself consists of “if-else” statements, switching depending on the type which is transmitted by the controller. In each command tree, the profiles which are available (depending on login status and entering function) are determined and prepared for presentation. If two profiles are presented on the map and the function is called for the first time or the “resize”-option has been used, the new center is computed. If latitude and longitude values have been transmitted, the map will be centered in this position, otherwise it will be focused on the presently saved center. Afterwards, the map string for the map request is build and the corresponding view is called.

[Reference “controller/mobile” on the disc, Annexes]

The movement on the map is realized by one function. The coding was basically done exactly according to the design. For the two-dimensional movement, the steps have to be computed new. Afterwards, they are added to or subtracted from the presently centered point. According to the 3-dimensional parameters, it has only to be checked if they are inside the allowed borders.

```
    function move($type, $direction, $prof_type="", $identifier="")
    /**
     * default moving function for static map
     * type - differenciate between zooming and moving (also place for later functionality like rezising and so on)
     * direction - differentiate betwen the mode of movement, according to the chosen type
     * prof_type - type of profile where you came from -> needs to fit to the cases of show_position
     * identifiert - unique identifier of the chosen profile -> needs to fit to the identifier for show_position
     */
    {
        switch($type)
        {
            case "move":
                {
                    $zoommod=$this->session->userdata('zoomlevel')-13;
                    //Computation of stepping:
                    //(((viewportsize/10000)-((viewportsize/10000)/6[one step = 1/6 of screen]))/2[left and right]) *
pow(2,$zoommod)[zoomdependency]
                    $latmod=((($this->data['viewport']['height']/10000)-(($this->data['viewport']['height']/10000)/6))/2) / pow(2,
$zoommod) ;
                    $lonmod=((($this->data['viewport']['width']/10000)-(($this->data['viewport']['width']/10000)/6))/2) / pow(2,
$zoommod) ;
                    $this->data['viewport']['vextend']=$latmod*2;
                    $this->data['viewport']['hextend']=$lonmod*2;
                    $mylat=$this->session->userdata('present_lat');
                    $mylon=$this->session->userdata('present_lon');
                    switch($direction)
                    {
                        case "left":
                            {
                                $this->session->set_userdata('present_lat',$mylat);
                                $this->session->set_userdata('present_lon',$mylon-$lonmod);
                                $this->session->set_userdata('current_center',"");
                                $this->show_position("","",$prof_type,$identifier);
                                break;
                            }
                        case "right":
                            {
```

Picture 27 Code example map interaction - Part 1

```
                                $this->session->set_userdata('present_lat',$mylat);
                                $this->session->set_userdata('present_lon',$mylon+$lonmod);
                                $this->session->set_userdata('current_center',"");
                                $this->show_position("","",$prof_type,$identifier);
                                break;
                            }
                        case "up":
                            {
                                $this->session->set_userdata('present_lat',$mylat+$latmod);
                                $this->session->set_userdata('present_lon',$mylon);
                                $this->session->set_userdata('current_center',"");
                                $this->show_position("","",$prof_type,$identifier);
                                break;
                            }
                        case "down":
                            {
                                $this->session->set_userdata('present_lat',$mylat-$latmod);
                                $this->session->set_userdata('present_lon',$mylon);
                                $this->session->set_userdata('current_center',"");
                                $this->show_position("","",$prof_type,$identifier);
                                break;
                            }
                    }
                    break;
                }
            case "zoom":
                {
                    switch($direction)
                    {
                        case "in":
                            {
                                if($this->session->userdata('zoomlevel')>=18)
                                {
                                    $zoommod=18;
                                }
                                else
                                {
```

Picture 28 Code example map interaction - Part 2

```
                        $zoommod=$this->session->userdata('zoomlevel')+1;
                    }
                    $this->session->set_userdata('zoomlevel',$zoommod);
                    $this->session->set_userdata('current_center',"");
                    $this->show_position("","",$prof_type,$identifier);
                    break;
                }
            case "out":
                {
                    if($this->session->userdata('zoomlevel')<=1)
                    {
                        $zoommod=1;
                    }
                    else
                    {
                        $zoommod=$this->session->userdata('zoomlevel')-1;
                    }
                    $this->session->set_userdata('zoomlevel',$zoommod);
                    $this->session->set_userdata('current_center',"");
                    $this->show_position("","",$prof_type,$identifier);
                    break;
                }
        }
        break;
    }
case "resize":
    {
        $this->show_position("","",$prof_type,$identifier, TRUE);
        break;
    }
}
}
```

Picture 29 Code example map interaction - Part 3

6. Proof of operability

Testing Phase

1140

After finishing the implementation, during which the module tests have been done, the system integration test has to be done. A testing strategy had to be found for this procedure.

Regarding the testing at M2Mobi, a testing system for Java- and web applications has been developed by a different department. This system was finished at the same time as the new mobile website. That's why, there had to be developed a manual testing plan (to not be completely depended on a new testing system) as well as the backend integration of this manual test, as a modified version, into the testing system. Database SQL-statements had to be sent directly to the database. The relations between the tables had also to be done manually.

For a representative result of the test the most low-end phones have been chosen to test this website. This step was vital in order to determine the technological border for this website as well as the degree of support of the layout technology CSS.

In this chapter, the testing phones will be presented at first. Afterwards, the testing plan will be presented and explained. At the end, the corresponding test results will be shown.

6.1. Overview of testing phones

6.1.1. LG Chocolate KG800

The LG Chocolate is a designer mobile phone. Apart from the serious design and the good touchpad-interaction buttons, this mobile phone is the low-end border of all the phones that are tested at M2Mobi. The mobile phone is not able to handle float point operations in Java correctly and the integrated browser does not even support a session. Therefore, even the login was impossible. Consequently, this phone was marked as "deprecated" and excluded later from testing.

6.1.2. **Motorola MotoSLVR L6**

The Motorola L6 is a good, solid mobile phone for web browsing. The browser itself is pretty fast. To focus the problems on this phone, it has to be said that the browser has several problems with W3C CSS, although the backend technologies are working without problem. Another issue which makes this to a low-end phone is the very slow reaction time of the navigation cross. Another big problem is, as even in higher Motorola mobile phones, that the resolution is not native. The resolution of 128x160 is resized by a much less resolution.

6.1.3. **Sony Ericsson Walkman W200i**

The Walkman W200i is an low-end phone for active people. It has a solid hull and supports various multimedia data formats.

Concerning web issues, this mobile phone is pretty slow in connection. In addition, it has various problems with very simple CSS options like background-image, pseudo-formats and even the color command for font color.

6.1.4. **Nokia 6230i**

The Nokia 6230i was determined as the best of the low-end phone while testing. It has a solid technology, a native 208x208 resolution, camera and a web browser which can come along with most of the basic CSS 1.0 commands as well as will backend technologies like cookies and sessions.

Concluding all this, it was sure that things that are not running on this phone won't run on other low-end phones either.

The point that makes it to a low-end phone is the limited connectivity, the relatively small resolution, the basic input, and, to only be mentioned, the old-style design.

6.2. Testing Plan

The testing plan was developed with focus on the interaction possibilities the user has for every use case. Every use case has to be tested and give the right output.

The test of the functional site should reveal system errors or software crashes. These bugs are crucial and have to be fixed immediately. Another point of the functional test is the correct linking to each page by forcing the tester to visit each link. Although link tests can be done automatically by testing software like Selenium, the complexity to configure this for only this one project is too much, especially if manual testing has also to be done, without caring about a link test.

In addition, web tests need a user interface test to guarantee usability. The aim is to uncover user interface traps or not obvious link guiding in the software. If this is not done, user interface bugs lead to less users of the system. Users get easily angry if not finding immediately what they are searching for. If this is not prevented, the user will leave the site.

Although the final testing procedure is very long, it can be done in an acceptable amount of time. The fact that the web software is used and tested many times of many different users, they will run different branches of the test, depending on environmental parameters, for instance the login status. The diversity of the test runs then covers the whole test to guarantee correct results.[17]

The testing system has the same structure as the printed out version. Nevertheless, there are some advantages about the test and bug-teacking system. At first, back-jumps indicated in the manual procedure as numbers in brackets are realized as links in the system. Therefore, back-jumps are much easier to realize which provides a better overview during the test. At second, in the system there can also be selected only specific parts of the full test. Therefore, the system is also very good to be used for module test. As the main aspect, it must be said that the bugs found and saved in the systems are automatically distributed to the responsible developer. This improves the bug-fixing process enormous.[18]

6.3. Testing results

It is recommended to have a closer look to the testing results in the Annexes.[19]

[17] mobile_website_test_printout, Annexes
[18] (Scheipl, 2009)
[19] Alejandro's test, Almudena's test, Nicolas' test [Annexes]

7. Evaluation of the software

Conclusion

After completing the testing phase, the software is distributed to users and clients. The maintenance phase includes improvement and minor extensions to developed software. New requirements are set and will be implemented until the changes are massive enough to start a new software project with another major extension.

In addition, at the end of the project, the analytic steps of the development also have to been reviewed to improve the analytic capabilities. Therefore, a comparison between the approximated time of the planning/specification phase and the real used time should be done to determine where there have been the parts which used more or less time than estimated. This improves on the one hand the approximations of the next software project and on the other hand, the software engineer can determine in which technology further education of the staff could be useful to improve productivity and experience.

Consequently, this chapter first covers a comparison between the estimated and real used time. Subsequently, the working result referring to the task will be evaluated. Afterwards, ideas will be presented which came up that could lead to new requirements and minor improvements.

7.1. Comparison of approximated and used time

The values and conclusions taken in this chapter refer to the approximations done in the specification and the really used time written down in the time tables for implementation and testing.

The first feature that has been accomplished was the search. The time difference, consuming 10.5 hours more used time than estimated, is extra-ordinary high. The reason for this is the low amount preparation time. Additionally, the pagination of the search was much harder to accomplish than expected because the search load every kind of profile instead of only locations or users. Therefore, the existing search controller had to be modified very much. This explains the high time difference.

The Login and Sign-Up process has been, in contrast to the search, much more overestimated. The code of the controllers and models has not been modified, so only the views had to be constructed from the scratch.

The time consumption for the profiles has been underestimated with around 6.25 hours more used than approximated. The whole functionality compressed in the profiles has been overseen. A point to improve estimations for profiles in the future is to count the sub-features of the profiles more instead of estimating this from existing code. The reason for this conclusion is that most of the core changes in each version are done in the sub-features of profiles.

The overestimated time for the photo gallery hasn't been expected. The use of the CodeIgniter framework and its own image library improved the implementation speed a lot. Therefore, it took less time to implement that. In future estimations, the analyst should be more trained in the used technology to calculate this. Regarding to this project, it can be considered as a one-time-mistake.

Although, in the tables it seems as if the map took also much less time, this is a different problem. The configuration of the map and its interaction options are not very easy. Another point is that, when the developer of the map leaves the company, the new developer responsible for this has to read a lot of documentation and diagrams to be able to modify this feature without breaking the existing version. The estimation of the map has fit to its really used time.

The following features have even been forgotten in the approximation or their requirements have been made in the implementation phase:

- Set my location
- Edit my location

These are 5.5 hours of work that haven't been calculated. Referring of some minor or major overestimations in the calculation, this little amount haven't had a big impact on the time.

All the other, not mentioned features were accomplished in more or less the expected time (estimation +/- 2 hours).

Finally, the total time needed was around 10 hours less than approximated. This shows a quite good approach in estimation that should also be used in future developments.

Regarding the testing approximation it's hard to take some conclusions on the comparison because the estimation was done before the decision have been taken to also introduce this project as the first project in the new testing and bug-tracking system.

Regarding to the total used and total approximated time, the difference is not too much considering the former mentioned reason. The difference of both times is 15 hours. Considering also that testing approximation is always very hard to do, especially for graphical user interface tests for web projects, the difference is acceptable.[20]

[20] Implementation Approximation,
Testing Approximation,

7.2. Evaluation of the process and result

The assignment was the design and implementation of a web interface for mobile IT-applications. Nulaz was chosen as the example IT-Application.

In conclusion of this assignment, it can be said that mobile web interfaces are a good possibility to serve low-end mobile phones with interactive web content without spending extra time on adaption for every available phone on the market. The web interface was successfully created and run on the vast majority of mobile phones. It doesn't only serve the maximum usability for low-end phones, but it also gives a basic, very fast and uncomplicated interface useable by high-end phones.

During the development, some issues of mobile web development have been faced. For most of them, nice solution has been found, as, for instance, it can be seen in the map feature. For some issues due to the available technology, workarounds have been found, as can be viewed in the chat feature. As a result of the in-depth testing phase, the boundaries possibilities for mobile web interfaces developed for low-end phones have been determined. One of these boundaries is the correct support of PHP sessions on mobile browsers. Without that, it is not possible to provide dynamical content. One example phone that is, because of this reason, not supported is the LG Chocolate KG800.

A big amount of effort in this project was given to the process of software engineering and development itself. It is not worth to do software engineering such detailed only for a small mobile web interface. In the case of this project, Nulaz, this big effort on the process was worth because of several reasons. Nulaz is a fast-developing and even at this stage high-developed software which complexity is also quickly increase. Until the beginning of the project, necessary documentation and design of the system was missing. This can lead to bigger problems in future development. That's why a lot of effort was taken into design and documentation to solve this issue. Due to the detailed time-and cost approximation, deadlines haven't been exceeded and the project was accomplished in time, which is unfortunately more an exception than the regular result.

Counting all this together, the project was completely successful and led to some new cognition about this field of software development.

---

time_consumption,
time_consumption_tests [Annexes]

7.3. Approaches for future versions

In foresight of future versions some new features have been found during the development that are nice to have in the website. This chapter gives an overview of possible future feature.

One feature that is possible to do in future versions is the integration of other web services and social community sites like “Facebook” or “Hives”. These web services also provide mobile web pages for their users. Consequently, a smooth connection between them is possible. In addition, the APIs for these services are even included in the normal web page. Therefore, the integration complexity as a maintenance feature is not so high.

Taking a look the newly introduced feed system, there can be much more done with the location-feed-interaction. A reasonable feature is the implementation of an option in the location profile to the corresponding feeds.

If the integration of other mobile web services will be done, the “Recent Activities” will also include a broader diversity of news.

The search and profiles can be extended with a new popularity measurement method. This will provide the user with even more accurate information when searching for specific data in Nulaz.

Finally, the map can be extended to provide the user with additional path information when entering the map with two profiles shown. This will improve the user’s experience according to the location based service.

Bibliography

Balzert, H. *Lehrbuch für Software-Management.*

Balzert, H. *Lehrbuch Software-Engineering.*

Baneke, M. (Juli 2009). About special aspects of mobile devices. (C. Kehl, Interviewer)

CodeIgniter. (n.d.). *CodeIgniter User Guide*. Retrieved April 2009, from http://codeigniter.com/user_guide/

Google. (n.d.). *Google Code FAQ*. Retrieved May 2009, from http://code.google.com/support/bin/topic.py?topic=10028

Inc., S. W. (kein Datum). *Loki - Precise location for Websites*. Abgerufen am 20. Juli 2009 von Loki - Precise location for Websites: http://www.loki.com

Roman, P. (2009, April). Google Maps Backend Information. (C. Kehl, Interviewer)

Scheipl, P. (2009). Testing System of Location based Services. In P. Scheipl, *Testing System of Location based Services.* Vienna.

Vitaletti, E. T. (Juni 2004). Cell-ID location technique, limits and benefits: an experimental study.

Index of Pictures

Index of Tables

Annexes

Mobile Website Specification
Mobile Website Specification V2
Mobile Website Specification V3
Sitemap
General Use Case Diagram
Implementation Approximation
Full Function Point maintenance projects
Full Function Point Method Web-Projects
General Class Diagram Models
General Class Diagram Controllers
Recent Activities Activity Diagram
Recent Activities Communication Diagram
Recent Activities Class Diagram
Recent Activities Use Case Diagram
Chats Activity Diagram
Chats Communication Diagram
Chats Class Diagram
Chats Use Case Diagram
Photo Gallery Class Diagram
Photo Gallery Use Case Diagram
Photo Gallery Activity Diagram
Map Class Diagram
Map Use Case Diagram
Map Activity Diagram
Map Communication Diagram
GeoCoder State Chart Diagram
GeoCoder Activity Diagram
Errors State Chart Diagram
Errors Activity Diagram

mobile_website_test_printout
Alejandro Test Result
Almudena Test Result
Nicolas Test Result
Implementation Approximation
Testing Approximation
time_consumption
time_consumption_tests

Statement of Self-Employment

Herewith I declare that I created this exam paper independently, without illegal external help and only by use of listed sources and tools.

Location, date Signature

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die hier vorliegende Arbeit, ohne unerlaubte fremde Hilfe und nur unter Verwendung der aufgeführten Hilfsmittel angefertigt habe.

Ort, Datum Unterschrift