ARCADA

Mobile Guide

An offline web application for mobile devices

Ted Mellin

Degree Thesis

Information technology

2011

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	
Författare:	Ted Mellin
Arbetets namn:	Mobil Guide
Handledare (Arcada):	Magnus Westerlund
Uppdragsgivare:	Art and Design Center Helsinki (ADC Helsinki)

Sammandrag:

Avsikten med examensarbetet var att skapa en tjänst som ger företag och organisationer möjligheten att skapa en digital guide för att förevisa ett geografiskt sett avgränsat område. Ett av de viktigaste kraven var att tjänsten bör vara både enkel att använda samt fungera utan Internet anslutning, detta på grund av dyra dataöverföringskostnader speciellt då man befinner sig utomlands.

Examensarbetet är uppdelat i teori och praktik. I teoridelen utreds ramverket Codeigniter som är stommen för tjänsten, dessutom analyseras HTML5 språket och möjligheterna det medför i utveckling av moderna webbapplikationer, speciellt med tanke på den mobila världen.

Den praktiska delen av examensarbetet beskriver administrationsverktyget och användargränssnittet, de tekniska lösningar som använts för utvecklingen och hur HTML 5 samt Codeigniter PHP-ramverket använts för att lösa problem och snabba upp utvecklingsprocessen.

Nyckelord:	HTML5, mobil, jquerymobile,offline,turism,guide, Codeigniter, cms, ramverk,php,karta
Sidantal:	48
Språk:	Engelska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Information technology
Identification number:	
Author:	Ted Mellin
Title:	Mobile Guide
Supervisor (Arcada):	Magnus Westerlund
Commissioned by:	Art and Design Center Helsinki (ADC Helsinki)

Abstract:

The goal of the thesis was to create a service which enables companies to create map-based mobile guides, restricted by a geographic area. One of the fundamental requirements was that the end-user web application had to be easy to use and also be functional even without an Internet connection. The reason for this is the high roaming costs for data transfer over mobile networks when visiting a foreign country.

The thesis is divided into two parts, the theoretical part and the practical part. The theoretical part describes the PHP framework Codeigniter. HTML 5 is also analyzed, specifically from a mobile web application perspective.

The practical part goes through the administration interface and the frontend application in detail. All problems and their solutions are covered as well as the usage of HTML 5 and Codeigniter to solve issues and speed up the development process.

Keywords:	HTML5, mobile, jquerymobile,offline,tourism,guide, Codeigniter, cms, framework,php,map
Number of pages:	48
Language:	English
Date of acceptance:	

OPINNÄYTE	
Arcada	
Koulutusohjelma:	Informaatiotekniikka
Tunnistenumero:	
Tekijä:	Ted Mellin
Työn nimi:	Mobiiliopas
Työn ohjaaja (Arcada):	Magnus Westerlund
Toimeksiantaja:	Art and Design Center Helsinki (ADC Helsinki)

Tiivistelmä:

Tämän opinnäytetyön tavoitteena oli kehittää palvelu jonka avulla yritykset sekä organisaatiot voisivat luoda karttapohjaisia mobiilioppaita maantieteellisesti rajoitetuille alueille. Yksi tilaajan keskeisimmistä vaatimuksista oli että palvelun täytyi olla helppokäyttöinen ja toimia yhteydettömässä tilassa. Syynä tähän on matkustaviin turisteihin kohdistuvat kalliit ”roaming” kulut.

Opinnäytetyö on jaettu kahteen osaan, teoreettinen osio kuvailee PHP runkoa Codeigniteria sekä HTML5, eritoten mobiili-näkökulmasta.

Käytännön osiossa esitellään itse palvelua sekä sen hallintapaneelia perusteellisesti. Kaikki kohdatut ongelmat sekä niiden ratkaisut on tarkasti kuvailtu tässä osiossa.

Avainsanat:	HTML5, mobiili, jquerymobile,yhteydetön,turismi,opas, Co-deigniter, cms, framework,php, kartta
Sivumäärä:	48
Kieli:	Englanti
Hyväksymispäivämäärä:	

CONTENTS

Abbreviations

POI	Position of Interest
ADC	Art and Design City
WDC	World Design Capital
HTML	Hypertext Markup Language
PHP	PHP: Hypertext Preprocessor
CSS	Cascading Style Sheet
URL	Uniform Resource Locator
CMS	Content Management System
ICSID	International Council of Societies of Industrial Design
MVC	Model-View-Controller
MySQL	My Structured Query Language
RFID	Radio Frequency IDentification
Wi-Fi	Wireless Fidelity
MAC address	Media Access Control address
GSM	Global System for Mobile communications
CDMA	Code Division Multiple Access
id	Identifier
API	Application Programming Interface

Figurer / Figures

1 INTRODUCTION

1.1 Background

World Design Capital is a worldwide project to encourage the use of design in the world's cities. Every other year a city is recognized for its commitment to design and the use of it to reinvent and improve its social, cultural and economic life. Once chosen, the city commits to a yearlong program of design related events, which are planned and organized by the WDC Organizing Committee (WDCOC), the working group of the International Council of Societies of Industrial Design (icsid) and the local WDC Project Management teams.

Once a city is recognized as the Design capital of the year, it is showcased on the international forum and promoted globally. For the city, and the country itself, this means growth within the tourism sector. As more tourists travel to the city seeking cultural and design attractions, there needs to be ways for them to discover and find these so called positions of interest.

After Helsinki was chosen as the World Design Capital of the year 2012, multiple projects within the region were initiated to improve the tourism experience. One of these projects was Mobile Guide, commissioned by Art and Design Center Helsinki (referred to as ADC hereafter).

1.2 The goal of the thesis

ADC, which is in charge of Arabianranta, one of the capitals most important design areas, wanted a way to present the area in a modern and effective way. Traditional tour guides usually consists of a group of tourists and a guide, which together walk around an area and admire its attractions. While this most likely is a great experience with conversations and interaction it has a few drawbacks or limitations. The tour is restricted by a schedule and a predefined route. Furthermore tourists are usually required to pay a fee to participate in the tour.

ADC wanted to reduce the need of traditional tour guides while maintaining or preferably improving the visitor overall experience. The solution was a pocket guide, an application that could be run on the tourist's mobile phone, containing all the basic information a traditional tour guide would have of the area. While this reduces the need for personnel it also solves all drawbacks mentioned earlier. The tourists no longer have to pay for the tour, and they can decide for themselves in which order and time to visit the attractions they have an interest in seeing.

The initial requirements were to develop and design a digital guide for the area Arabianranta. The guide had to be easy to administer and update as well as have an elegant and usable frontend. One of the big realizations made in the early stages of development were that the main target group, namely tourists, rarely have cheap Internet access when visiting a foreign country. This realization came to change the entire development process as all functionality built into the service had to work without an active Internet connection.

Apart from the offline support, ADC also insisted on making the guide as accessible as possible, meaning that the installation process had to be quick and easy. This was solved by removing the traditional installation process completely, by utilizing HTML5 functionality. This solution will be covered in great detail as it is one of the cornerstones for the whole system.

The theory part of this paper describes the three main building blocks of the system, namely the PHP framework Codeigniter, the hypertext language HTML5 and the mobile web framework jQueryMobile. The practical part will examine the admin interface as well as the end-user application and how the theory was applied in practice.

2 CODEIGNITER PHP FRAMEWORK

PHP frameworks exist to speed up the development and keep the project code organized. They also help the developer follow standards such as MVC, which is short for the Model–view–controller design pattern.

Codeigniter is one of the most popular PHP frameworks and this was part of the reason why it was chosen. Out of the 10 frameworks examined and compared, it quickly became clear that Codeigniter had the shortest learning curve. One reason for this is its loose approach to MVC it introduces, as it does not force the developer to use models for defining data structures. Models aid the developer in the database communication within the application. It is often better and faster to build your own helper libraries to handle database operations and the representation of your data.

2.1 How Codeigniter works

Every Codeigniter page is called through the index file. Here all the resources required for Codeigniter to function are loaded. It also accepts URL parameters which dictate which controller should be loaded.

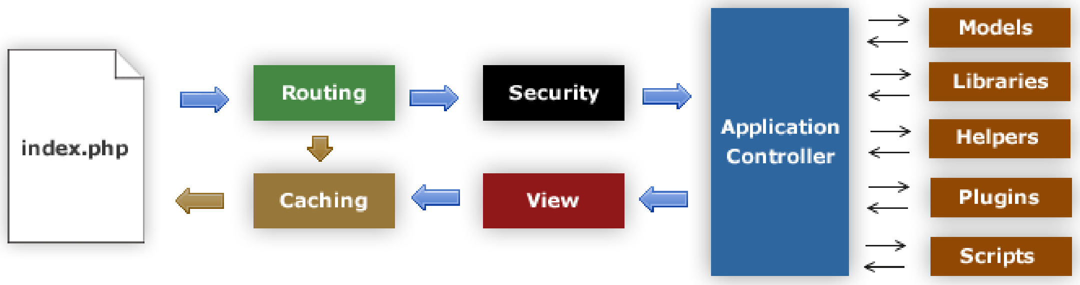


Figure 1 Codeigniter application flow chart (Codeigniter,2011)

As Figure 1 shows, the application controller then proceeds to load helpers and libraries which can contain added functionality, one example of a helper usually loaded in Codeigniter is an authentication helper which handles the system access control. Once the Controller has carried out given tasks a view (or views) can be loaded.

Upon loading the view(s), parameters can be passed from the controller. As an example, the controller could fetch and parse data from the application and pass the manipulated data to the view. This in turn means that the view only takes care of visualizing the given data, and keeps the application logic and interface code separated.

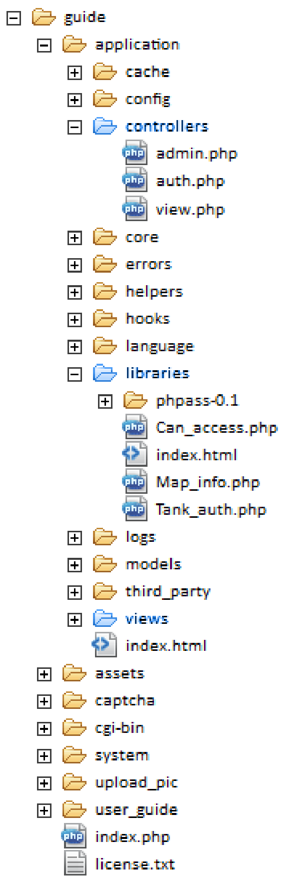
Aside from loading views, the controller is useful for handling data manipulation and background calculations through asynchronous communication.

```
class Load extends CI_Controller {
           function __construct() {
                      parent::__construct();
                      $this -> load -> helper('url');
           }
    function map(){
        $data['name'] =  $this -> uri -> segment(3);
        $this -> load -> view('map_view', $data);
    }
}
```

Figure 2 A simple example of a Codeigniter controller class

The example code in Figure 2 is a Codeigniter controller class in its most simple form, here the controller name is “Load”, which means the above code is located in “application/controllers/load.php”. The method “map” is programmed to load a view called “map_view” which in reality is a file called “map_view.php” located in the folder “application/views”. This method could be successfully run with the URL “index.php/load/map/helsinki”, where helsinki is the third URL segment expected by the method.

One thing to remember when starting development with a PHP framework is, do not expect to get a fully functional CMS. A framework does not come with a working admin interface or templates. It only contains code to aid the developer in the creation of such interfaces. The folder structure of a Codeigniter project is presented in Figure 3, the most crucial folders are displayed in blue.



```
guide
  application
    cache
    config
    controllers
      admin.php
      auth.php
      view.php
    core
    errors
    helpers
    hooks
    language
    libraries
      phpass-0.1
      Can_access.php
      index.html
      Map_info.php
      Tank_auth.php
    logs
    models
    third_party
    views
    index.html
  assets
  captcha
  cgi-bin
  system
  upload_pic
  user_guide
  index.php
  license.txt
```

Figure 3 The tree structure of a Codeigniter project

2.2 Installation

The installation process of Codeigniter is quite similar to any other framework or Content management system. First you need to make sure the server requirements are met, these vary depending on version being installed, the latest version of Codeigniter (v. 2.0.3) requires a PHP version of 5.1.6 or newer and supports the databases MySQL (4.1+), MySQLi, MS SQL, Postgres, Oracle, SQLite, and ODBC (Codeigniter 2011).

Once the server requirements are met, the installation is as simple as downloading the installation package and extracting the package to a folder on your server. After this the config file needs to be edited, that is located in *application/config/config.php*. In the config file the projects base URL and encryption keys are set.

After this Codeigniter is up and running, however if a database is required, which is usually the case, the database login information needs to be set in *application/config/database.php*

On top of this, the Codeigniter community contains a lot of user-generated libraries and helpers which can be installed to add general functionality. An example of such a helper is Tank Auth which Mobile guide utilizes to control user authentication and their privileges. The installation of this helper function was pretty similar to the installation of Codeigniter itself. It involved extracting the helper package to the Codeigniter root and creating tables in the database. Some minor tweaks were required to get this working, but these were described in great detail in the libraries installation guide.

3 HTML5

HTML5 is the newest revision to HTML. First drafted six years ago it still remains at a stage of Working Draft according to the World Wide Web Consortium (W3C). Even though it is not considered an official standard yet, it is already supported in the latest versions of most modern browsers, both desktop and mobile ones.

HTML5 is much more than a structuring language for web pages. The greatest benefit of HTML5 is the introduction of numerous application programming interfaces (API), such as Canvas, Video, Local storage, Offline web applications, Geolocation and more. Mobile guide utilizes two of these HTML5 APIs, Offline web applications for storing data on the user’s device and the Geolocation API for locating the user. None of the APIs introduces new requirements on the server, as it does not involve server side scripting.

3.1 Offline web applications

HTML5 lets developers make their web applications work without an Internet connection. This has been possible in the past, e.g. by using Google Gears, but now it’s becoming a standard included in HTML5, which means that the same application can be run on a multitude of modern browsers, including Firefox, Chrome, Safari, Opera, Android browsers and more.

As any offline application, the offline web application has to be loaded on to the device. This means that in order to use the web application offline, an initial visit to the website while connected to the Internet is required. Once the page is loaded in the browser, the server tells the browser which files are needed for the website to function offline, these files are then downloaded. The files required are all listed in a file called the manifest, which can be modified by the developer. The next time the website is visited, even without an Internet connection, it will display the page as if the user was online, if all elements of the page are listed in the manifest. Sometimes a developer might want the page to look or function differently, which is also possible through some minor tweaks. Figure 4 contains the minimal amount of code to run a webpage in offline.

```
<!DOCTYPE html>
<html lang="en" manifest="somemanifest.appcache">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>HTML5 - offline</title>
</head>
<body>
<h2>HTML5 - offline</h2>
<img src="imagename.jpg" alt="imagename"/>
</body>
</html>
```

```
CACHE MANIFEST

CACHE:
imagename.jpg
```

Figure 4 Example of an html file and its manifest file

The manifest file is always defined within the html tag of the web page. The browser proceeds to download all files located under the CACHE line. The file which calls the manifest is automatically downloaded so it is not required in the manifest file itself.

The manifest is divided into sections, the CACHE section mentioned above being the most relevant one. Besides the CACHE files the manifest is also able to tell the browsers which files should never be cached, these go under the header NETWORK.

3.2 Geolocation API

The Geolocation API can be used to retrieve the device position. The common source for location data is GPS, however sometimes the device or the browser does not support GPS, in this case other sources are automatically used such as RFID, Wi-Fi and Bluetooth MAC addresses, and GSM/CDMA cell IDs. Most browsers require the user to authorize the use of location services. If they deny the browser access then no location data will be available through the Geolocation API.

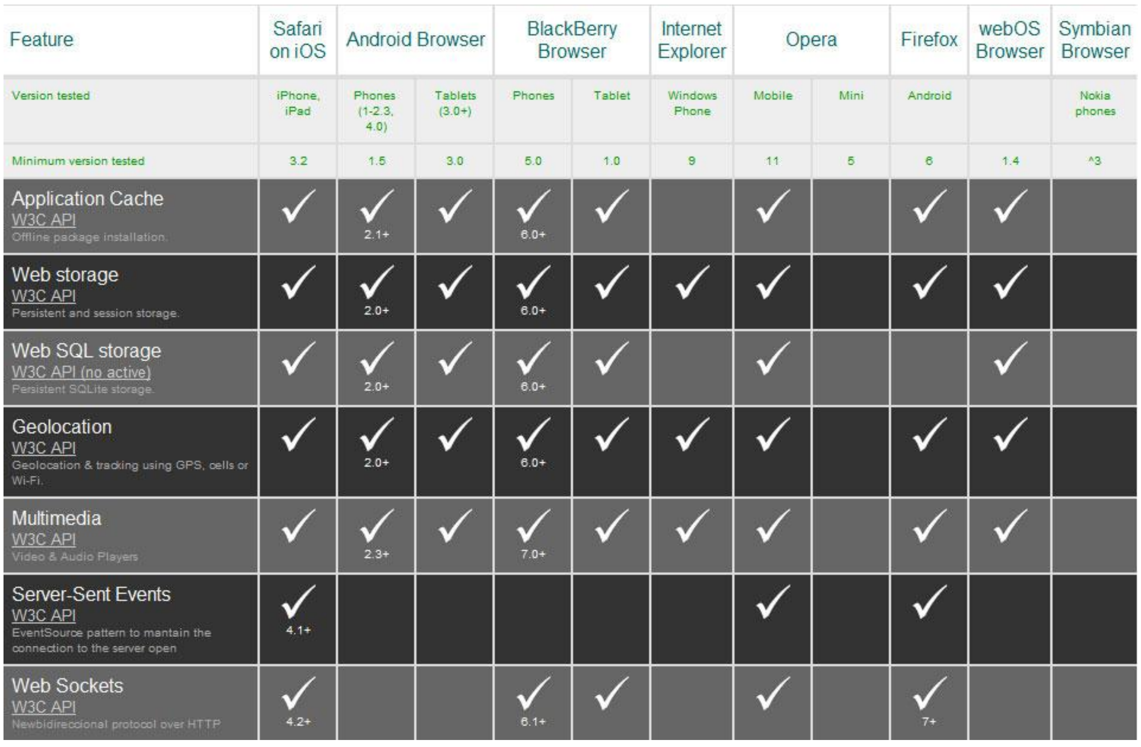
The API can do a one-time location query or alternatively be set to retrieve the location on a defined interval. The method for retrieving the location only once is “navigator.geolocation.getCurrentPosition(callback_function)”. The given callback-function then takes the result as a parameter and can read the longitude and latitude values from it. The code for fetching the location continuously is usually a bit more complex, as it involves starting and stopping the location querying and handling potential errors. Figure 5 shows a simple example of how to repeatedly fetch the location and use the data.

```
function scrollMap(position) {
// Scrolls the map so that it is centered at (position.coords.latitude, posi-
tion.coords.longitude).
}

// Request repeated updates.
var watchId = navigator.geolocation.watchPosition(scrollMap);

function buttonClickHandler() {
// Cancel the updates when the user clicks a button.
navigator.geolocation.clearWatch(watchId);
}
```

Figure 5 Example code for repeated location querying using the Geolocation API

3.3 HTML5 support in modern browsers

Even though HTML5 is not an official standard, it has been widely accepted and implemented by both desktop and mobile browsers. Some of the HTML5 functionality however can be implemented differently depending on the browser, and it is crucial to have a good understanding of these differences when targeting multiple platforms and browsers.



Feature	Safari on iOS	Android Browser		BlackBerry Browser		Internet Explorer	Opera		Firefox	webOS Browser	Symbian Browser
Version tested	iPhone, iPad	Phones (1-2.3, 4.0)	Tablets (3.0+)	Phones	Tablet	Windows Phone	Mobile	Mini	Android		Nokia phones
Minimum version tested	3.2	1.5	3.0	5.0	1.0	9	11	5	6	1.4	^3
Application Cache W3C API Offline package installation.	✓	✓ 2.1+	✓	✓ 6.0+	✓		✓		✓	✓	
Web storage W3C API Persistent and session storage.	✓	✓ 2.0+	✓	✓ 6.0+	✓	✓	✓		✓	✓	
Web SQL storage W3C API (no active) Persistent SQLite storage.	✓	✓ 2.0+	✓	✓ 6.0+	✓		✓			✓	
Geolocation W3C API Geolocation & tracking using GPS, cells or Wi-Fi.	✓	✓ 2.0+	✓	✓ 6.0+	✓	✓	✓		✓	✓	
Multimedia W3C API Video & Audio Players	✓	✓ 2.3+	✓	✓ 7.0+	✓	✓	✓		✓	✓	
Server-Sent Events W3C API EventSource pattern to mantain the connection to the server open	✓ 4.1+						✓		✓		
Web Sockets W3C API Newbidireccional protocol over HTTP	✓ 4.2+			✓ 6.1+	✓		✓		✓ 7+		

Figure 6 HTML5 support on a range of modern browsers (Mobile HTML5,2011)

There are many sources online for a quick overlook of HTML5 support on different browsers. Figure 6 contains a comparison of a few, but not nearly all functionality defined within the HTML5 structuring language.

Making sure the target platforms supports the required functionality is important, but it is also crucial to know exactly in what way it is supported. As an example, Android implements Application Cache without any limitations on the total size of the files to be cached. Mobile Safari on the other hand has a 10MB limit per website. After this limit is reached the user is prompted with a message asking them for permission to increase the cache limit up to a total of 25MB for the active website. The best possible research is to actually test the applications on as many browsers as possible yourself, although this might not be feasible. Browser manufacturers usually provide developers with extensive documentation over their HTML5 implementations.

4 JQUERYMOBILE

jQueryMobile is a framework built on jQuery with a focus on mobile devices. The framework helps the developer create mobile-optimized web applications which work and look the same on all major mobile operating systems, such as Android, IOS, Blackberry, Bada, Windows phone, Palm web OS, Symbian and Meego.

Touch optimization is one of the most important features found in jQueryMobile. The framework consists of many different widgets and layouts which mimic the usability found in the most popular mobile operating systems and applications.

4.1 Single page design

One big difference with jQueryMobile compared to traditional web design is that a page is not identified by a file on the server, but as a div element with a unique ID so that all pages are defined within the same file and body tags.

It might take some time to get used to this way of working, especially since it limits some functionality widely used in traditional web development. As an example, it is not possible to exchange data between pages using the traditional methods GET and POST. It is however possible to pass variables around from page to page using JavaScript variables.

A page can, but doesn’t have to contain a header, content and footer container. When using one of the default themes that comes with jQueryMobile these give the website a look and feel of a real mobile application. There is also endless of styling possibilities available using traditional CSS. The default themes are mainly there to give the developers something to start off with. Figure 7 contains a simple example of the document structure for a jQueryMobile website.

```
<body>
<!-- Start of first page -->
<div data-role="page" id="index">
		<div data-role="header">
			<h1>Mobile Guide</h1>
		</div><!-- /header -->
		<div data-role="content">
			<p>I'm first in the source order so I'm shown as the page.</p>
			<p>View internal page called <a href="#mapview">Map view</a></p>
		</div><!-- /content -->
		<div data-role="footer">
			<h4>Page Footer</h4>
		</div><!-- /footer -->
</div><!-- /page -->
<!-- Start of second page -->
<div data-role="page" id="mapview">
		<div data-role="header">
			<h1>Map view</h1>
		</div><!-- /header -->
		<div data-role="content">
			<p>This is where the map view is displayed.</p>
			<p><a href="#index">Back to main page</a></p>
		</div><!-- /content -->
		<div data-role="footer">
			<h4>Page Footer</h4>
		</div><!-- /footer -->
</div><!-- /page -->
</body>
```

Figure 7An example of a simple jQueryMobile page

One of the reasons why jQueryMobile is designed to run within a single page is page transitions. This would not be possible if the user was redirected every time the page is changed. The page can be changed programmatically and traditionally using traditional web links. Both approaches accept a page identifier as a value, furthermore a transition effect such as “flip”, “slide”, “pop” can be given to make the transition more smooth and modern looking. Figure 8 shows how transition effects are applied.

```
<a href="#mapview” data-transition=”slideup”>Clickable link</ a>

$.mobile.changePage( "#mapview", { transition: "slideup"} );
```

Figure 8 An example of how internal linking works within jQueryMobile

5 THE FRONTEND

The frontend is a simple to use tool for exploring the area and its positions of interest either through a list view or a map. It is particularly intended for mobile devices but does support stationary devices, even though these come with a few limitations, such as no real geo-positioning.

5.1 Device support

One of the most important requirements for the application was that it had to work on as many devices as possible. Many cross-platform development frameworks such as Phonegap, Titanium Appcelerator and pure HTML5 were considered. HTML5 was eventually chosen as all modern phones, tablets and computers support it. Furthermore a web application does not have to but can be installed on a device, greatly speeding up the usage. HTML5 suited all of the applications needs perfectly, such as the ability to programmatically store data on the device and look up of the user's location using various localization methods such as GPS. As seen in Figure 9, Mobile Guide can be run on both desktop and mobile devices thanks to jQueryMobile and HTML5.

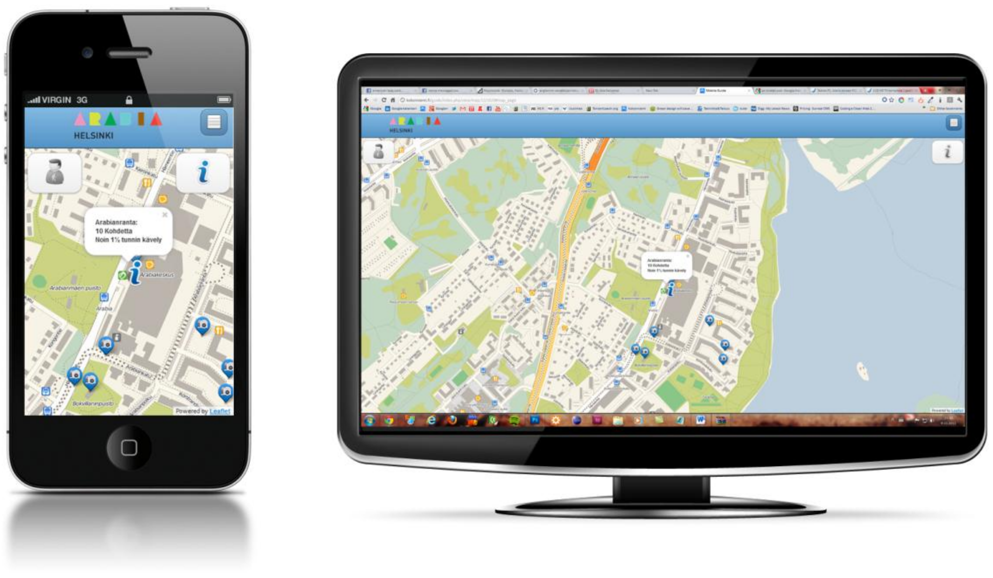


Figure 9 Mobile Guide ran in a mobile phone and on a PC

5.2 Multilingual

Mobile Guide is primarily designed for tourists, to aid them in finding and exploring new areas. This means that localization is of great importance. That is why the first step when the guest visits the web application is to choose the desired language from a list of available languages, see Figure 10. The system is built so that administrators have the alternative to define multiple languages and provide each position of interest with translations in desired languages. In the case of a missing translation for a specific position of interest, the info is displayed in the maps default language, e.g. English.

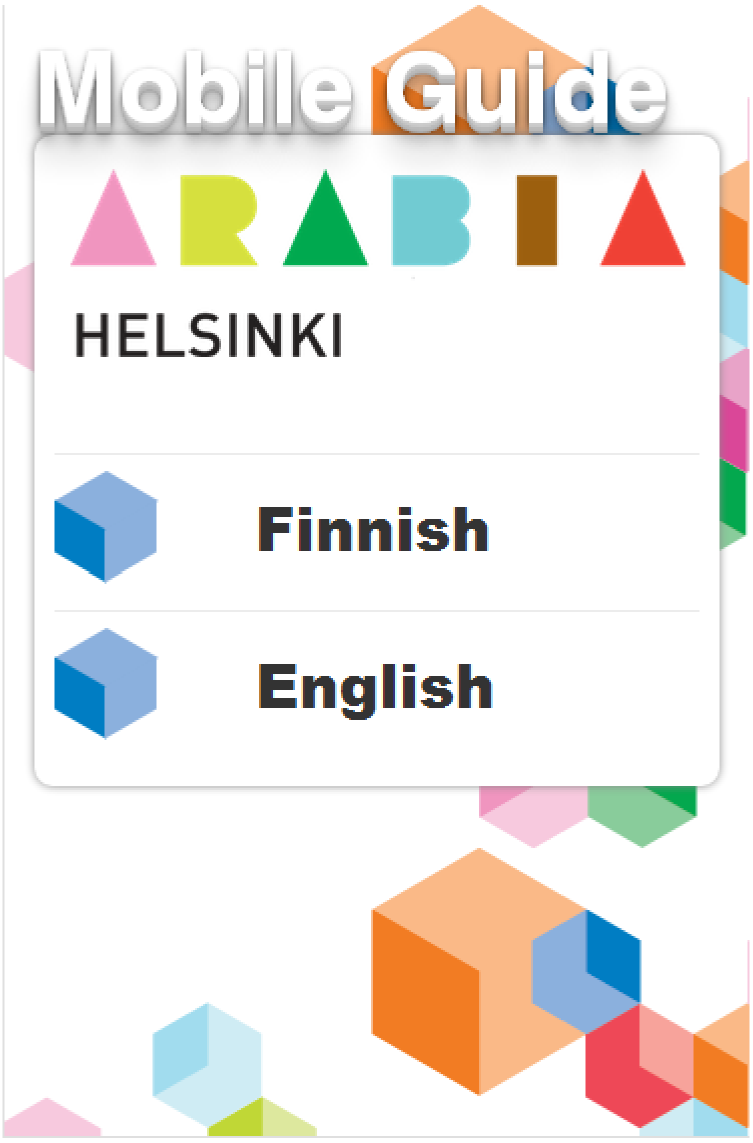


Figure 10 A view of the front-end language select

5.3 Going offline

After choosing a language the user is presented with a selection of operating mode, see figure 11. Going offline means that all “Positions of interest” (POI) along with required map tiles are downloaded and stored in the browser cache using a HTML5 manifest file. As mentioned in the manifest file chapter, in its most simple form the manifest contains a list of files required for the website to function without an Internet connection. Once the user visits the website for the first time, all files listed in the manifest are downloaded. After this users can visit the website again without an active Internet connection, and use the website as if they were connected to the Internet.

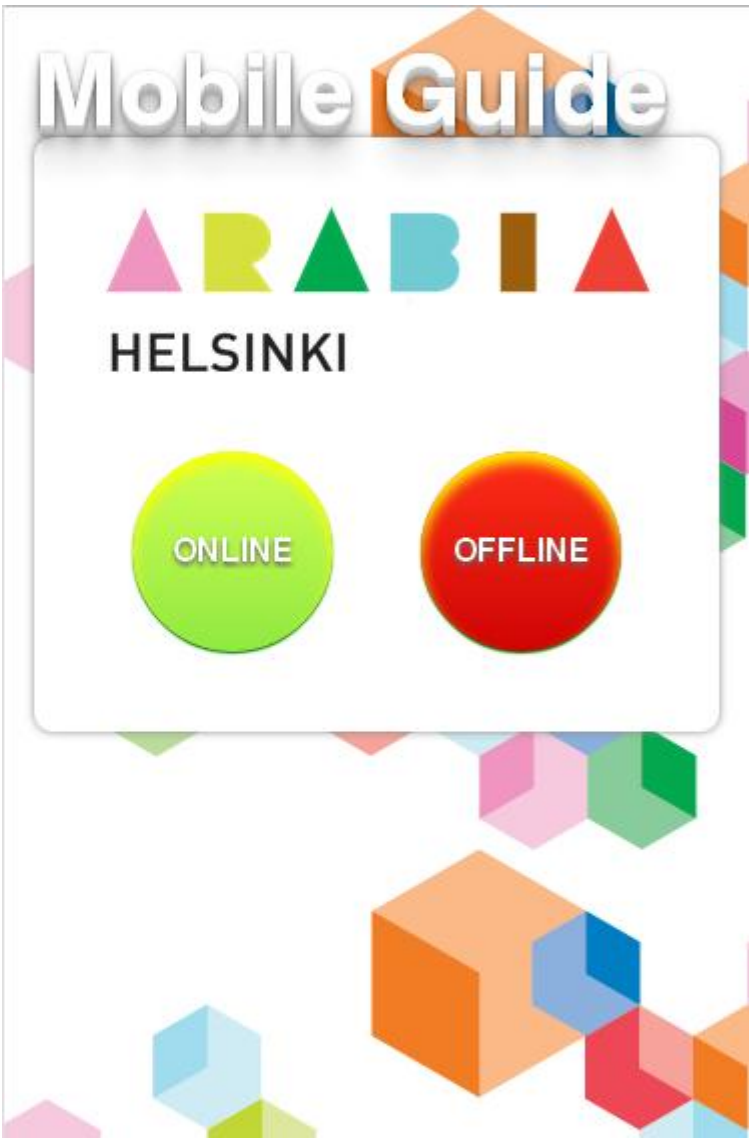


Figure 11 A view of the front-end operation mode select

5.3.1 Dynamic manifest file

Mobile Guide itself is a highly dynamic application, completely customizable by the map owners. This means that the manifest file also had to be just as dynamic as the application itself. The whole process is described in Figure 12, with code taken out of context.

```
// index.php:
<?php
<html manifest="<?=base_url()?>manifest/view/<?=$mapID?>/<?=$langID?>">
?>

// manifest file:
header('Content-Type: text/cache-manifest'); // !important
echo "CACHE MANIFEST\n"; // !important
echo "CACHE:\n"; // !important
echo base_url()."assets/js/jquery/jquery.js\n";
echo base_url()."assets/js/jquerymobile/jquery.mobile-1.0rc2.min.js\n";
echo base_url()."assets/js/jquerymobile/images/ajax-loader.png\n";
echo base_url()."assets/js/leaflet3/leaflet.css\n";
$mapID = $this->uri->segment('3'); // third parameter in url
$langID = $this->uri->segment('4'); // fourth parameter in url
$this -> map_info -> initialize($mapID);
$this -> map_info -> mapBounding();
$this -> map_info -> mapTiles($this->map_info->zoom);
$this -> map_info -> poiHtml($langID);
foreach($this->map_info->poiDataArray as $poiObject){
           foreach($poiObject->image as $mediaFile){
                      echo base_url().$mediaFile->source."\n";
           }
}
foreach($this->map_info->tileArray as $tile){
           $url = $this->map_info->tileUrl;
           $placeholders = array('{z}', '{x}', '{y}');
           $replacewith = array($tile->z, $tile->x, $tile->y);
           echo str_replace($placeholders, $replacewith, $url)."\n";
}
```

Figure 12 Dynamic manifest file using PHP

The only requirement for the page to be cached is to tell it where its manifest file lies. Because the manifest file needs to be able to generate a list of files based on the selected map, the parameters $mapID and $langID are passed along with the path.

Because a regular manifest file is not capable of interpreting PHP code, a PHP file is required (extension .php). The other problem here is that a browser does not recognize a PHP file as a manifest file, and will not be able to parse it. That is why we manually set the content type of the PHP file to "text/cache-manifest" using PHP:s header() function.

As mentioned in the chapter "Offline web applications" chapter the "CACHE MANIFEST" line is always required to be the first line in the document for the caching process to work. The "CACHE:" line in turn tells the browser, that each line beneath it contains a file that should be cached. Alternatives to CACHE are NETWORK and FALLBACK, which are not required in this application.

Now that the manifest file is set up to handle PHP code, the special map classes are available. All required map data such as images, text, configurations and map tiles are fetched and echoed into the manifest file. Figure 13 shows a small portion of the files found in the manifest for a Mobile Guide Map.

```
CACHE MANIFEST
CACHE:
#1320
http://kokonniemi.fi/guide/assets/js/jquerymobile/jquery.mobile-1.0rc2.min.css
http://kokonniemi.fi/guide/assets/js/jquery/jquery.js
http://kokonniemi.fi/guide/assets/js/jquerymobile/jquery.mobile-1.0rc2.min.js
http://kokonniemi.fi/guide/assets/js/jquerymobile/images/ajax-loader.png
http://kokonniemi.fi/guide/assets/js/leaflet3/leaflet.css
http://kokonniemi.fi/guide/assets/js/leaflet3/leaflet.js
http://kokonniemi.fi/guide/assets/js/leaflet3/images/marker.png
http://kokonniemi.fi/guide/assets/js/leaflet3/images/marker-shadow.png
http://kokonniemi.fi/guide/assets/images/layout/user.png
http://kokonniemi.fi/guide/assets/images/layout/user-bw.png
http://kokonniemi.fi/guide/assets/images/layout/home.png
http://kokonniemi.fi/guide/assets/images/layout/home-bw.png
http://kokonniemi.fi/guide/assets/images/layout/poi-icon.png
http://kokonniemi.fi/guide/assets/js/tinysort/jquery.tinysort.min.js
http://kokonniemi.fi/guide/assets/js/photoswipe/simple-inheritance.min.js
http://kokonniemi.fi/guide/assets/js/photoswipe/code-photoswipe-jQuery-1.0.19.min.js
http://kokonniemi.fi/guide/assets/js/photoswipe/photoswipe.css
http://kokonniemi.fi/guide/assets/js/photoswipe/photoswipe-icons.png
http://kokonniemi.fi/guide/assets/js/photoswipe/photoswipe-loader.gif
http://kokonniemi.fi/guide/assets/images/layout/arrow-left-icon.png
http://kokonniemi.fi/guide/assets/images/layout/arrow-up-icon.png
http://kokonniemi.fi/guide/assets/images/layout/bullet-2-icon.png
http://kokonniemi.fi/guide/assets/images/layout/list.png
http://kokonniemi.fi/guide/assets/images/layout/map.png
http://kokonniemi.fi/guide/index.php/view/map/13/20/1
http://kokonniemi.fi/guide/index.php/manifest/view
http://kokonniemi.fi/guide/upload_pic/resize_1317224456.png
http://kokonniemi.fi/guide/upload_pic/resize_1317224333.png
http://a.tile.cloudmade.com/84b255854c18417b8641e17eb887c074/997/256/16/37312/18955.png
http://a.tile.cloudmade.com/84b255854c18417b8641e17eb887c074/997/256/16/37312/18956.png
http://a.tile.cloudmade.com/84b255854c18417b8641e17eb887c074/997/256/16/37312/18957.png
http://a.tile.cloudmade.com/84b255854c18417b8641e17eb887c074/997/256/16/37312/18958.png
……
```

Figure 13 HTML5 Manifest file example

The Mobile Guide application also implements some user friendly functionality by showing the progress of the caching progress. The HTML5 offline web application API contains events for each stage of the caching progress, by listening to these events and examining the data they return, it is possible to count the number of files that have been cached. The event cacheProgress(e) is triggered every time a file listed in the manifest is downloaded. There is however no documented way to quickly count the amount of files that are left. Mobile Guide therefore contains a quick solution for this.

```
<script>
$.ajax({
			type: "get",
			url: "<?=base_url()?>manifest/view/<?=$mapID?>/<?=$langID?>",
			dataType: "text",
			cache: false,
			success: function(content ) {
			// Strip out the non-cache sections.
			// NOTE: The line break here is only to prevent wrapping
			content = content.replace(new RegExp("(NETWORK|FALLBACK):" +
"((?!(NETWORK|FALLBACK|CACHE):)[\\w\\W]*)", "gi"), "");
			// Strip out all comments.
			content = content.replace(new RegExp("#[^\\r\\n]*(\\r\\n?|\\n)", "g"), "");
			// Strip out the cache manifest header and trailing slashes.
			content = content.replace(new RegExp("CACHE MANIFEST\\s*|\\s*$", "g"), "");
			// Strip out extra line breaks and replace with a hash sign that we can break on.
			content = content.replace(new RegExp("[\\r\\n]+", "g"), "#");
			// Get the total number of files by counting the amount of # symbols
			var totalFiles = content.split("#").length;
			cachable = totalFiles;
}
</script>
```

Figure 14 Javascript code for calculating total amount of files within the manifest

As seen in Figure 14, everything not referring to a file is removed from the manifest file. After this, every line left contains a file to be downloaded, so the amount of lines is equal to the total amount of files that are to be downloaded. The system then divides the amount of downloaded files with the total amount and displays the percentage to the user within an animated horizontal bar (Figure 15).

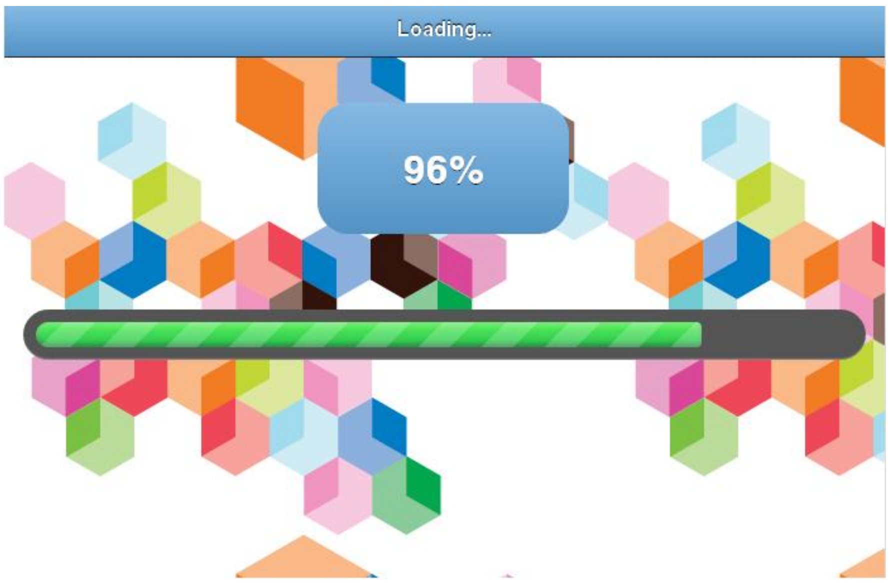


Figure 15 Screenshot of the download process

5.4 Mobile optimization

One big problem in traditional web design is creating websites which look good and function in all resolutions. Developers usually end up designing their pages for the smaller desktop resolutions. To make a website mobile-optimized, it needs to be built from the ground up with scalability in mind. This means all Interface elements such as buttons and forms need to scale and reposition depending on the current resolution. Scaling can be achieved with CSS and JavaScript, but requires a lot of tweaking and work to make it cross-browser compliant. There are many mobile frameworks based on CSS and JavaScript which aid the developer in optimizing their website for all resolutions and browsers. jQTouch, jQueryMobile and Sencha Touch are a few popular examples of such frameworks. jQueryMobile was the framework of choice for this application, mainly because of its interoperability with the leading JavaScript framework jQuery.

Thanks to jQueryMobile, the application can be run in any resolution and preserve its look and feel, as an addition to this jQueryMobile also enables sleek transitions when changing pages. This functionality is utilized on every page of the application.

As mentioned earlier, jQueryMobile also comes with default themes which tell the browser in which sizes and colors the elements should be rendered. The Mobile Guide front-end uses parts of the default theme framework for sizing and positioning of elements. But the color schemes and images are completely customizable by the map administrator and therefore replace much of the default theme.

5.5 Map view

The map view is a graphical representation over the area and its POIs. All POIs are represented on the map, and once clicked-on a small popup with the position name and a picture of the position is displayed. By clicking the popup the user is redirected to the POI page, containing all text data and images provided for that specific position.

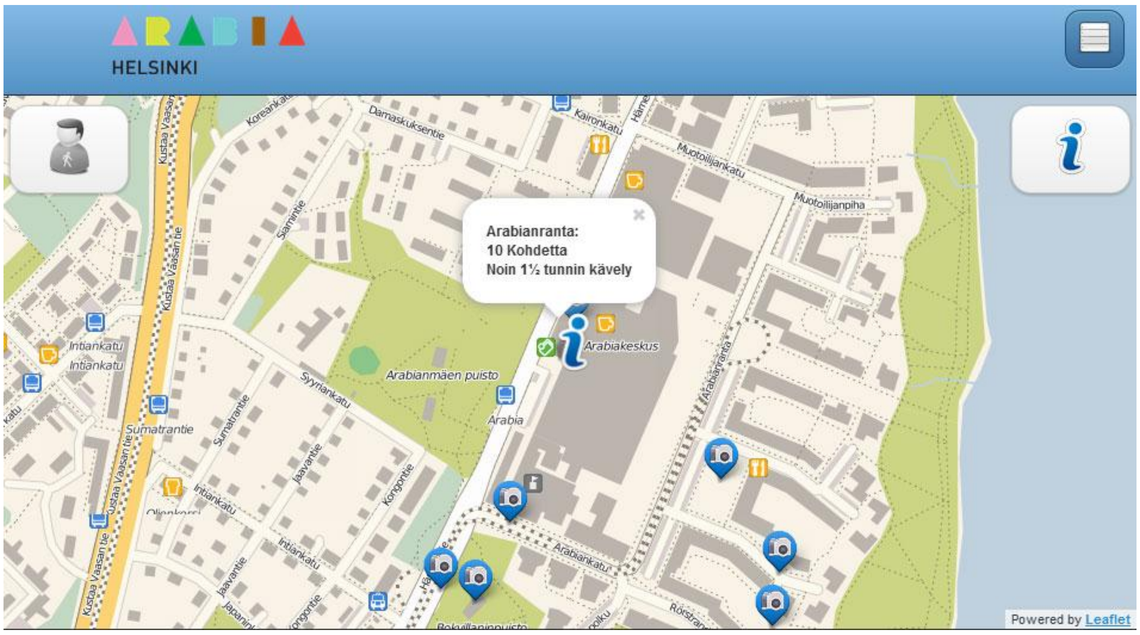


Figure 16 The map view

The map view is navigated by touch input, so the user can touch and drag the screen to pan the map in all directions. The zoom level is locked so that the amount of tiles to download is held to a minimum. Since it is not possible to zoom, there are two buttons which help the user navigate around the map, see Figure 16. In the top-left corner lies the “My location” button, which when clicked pans the view to the users position and activates the user tracking. When activated, the user tracking automatically pans the map according to the user’s position, always keeping the user centered in the view.

In the top-right corner sits an info icon, which is linked to a special position of interest, the info point. The info point is a location which represents a central point of the area. This could for example be a tourist office or simply a location in the middle of all attractions. Once the user clicks the info icon in the top-right corner the map pans to the info point. This helps the user locate the attractions on the map.

When selecting the API for displaying the map to the user two criteria's had to be met. The map API had to be touch capable, so that the user could pan the map by swiping the screen. Another requirement was that it had to be possible to download the API and display locally stored map tiles. Google Maps was first evaluated, while having excellent touch support it was not possible to use offline. Technically it is possible to cache Google Maps tiles and scripts through some tweaking. This was tested and proved to work well. However, Google has strict terms of service regarding bulk download of their content:

"(c) No Mass Downloads or Bulk Feeds of Content. You must not use the Service in a manner that gives you or any other person access to mass downloads or bulk feeds of any Content, including but not limited to numerical latitude or longitude coordinates, imagery, visible map data, or places data (including business listings). For example, you are not permitted to offer a batch geocoding service that uses Content contained in the Maps API(s)." - http://code.google.com/intl/fi-FI/apis/maps/terms.html 10.1.3

This meant that the Google Maps API was not an alternative, as offline support was one of the most important parts of the Application. Other open source alternatives such as OpenLayers, CloudMade, TouchMapLite and Leaflet were evaluated. Leaflet was chosen due to small script file sizes, good looking interface elements and its great mobile support.

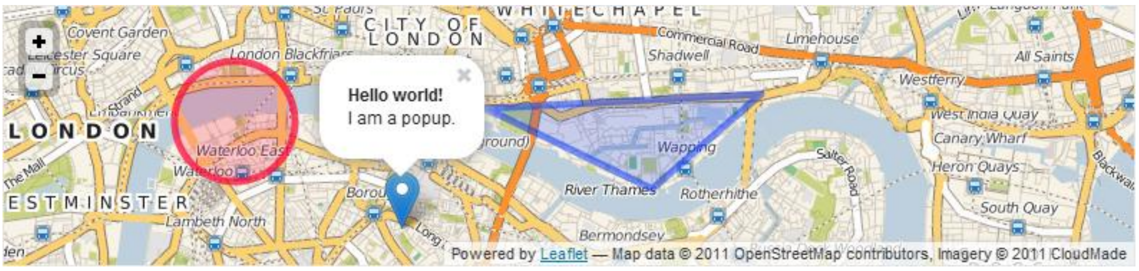

Hello world!
I am a popup.
Powered by Leaflet — Map data © 2011 OpenStreetMap contributors, Imagery © 2011 CloudMade

Figure 17 Leaflet demo map

5.5.1 Leaflet usage and functionality

Getting started with Leaflet is in itself quite easy, all that is required is to include the CSS and JavaScript source files in the document, add a container div and assigning it a unique identifier. Then initialize the map using a few lines of code, where the developer defines the desired tile source and a few optional options. Figure 18 contains all the code required to display a map view similar to the one found in Figure 17.

```
<!DOCTYPE html>
<head>
<script src="http://leaflet.cloudmade.com/dist/leaflet.js"></script>
<link rel="stylesheet" href="http://leaflet.cloudmade.com/dist/leaflet.css" />
<!--[if lte IE 8]><link rel="stylesheet" href="leaflet/leaflet.ie.css" /><![endif]-->
<script type="text/javascript">
var map = new L.Map('map');
var cloudmadeUrl = 'http://{s}.tile.cloudmade.com/YOUR-API-KEY/997/256/{z}/{x}/{y}.png',
    cloudmadeAttrib = 'Map data &copy; 2011 OpenStreetMap contributors, Imagery &copy; 2011 CloudMade',
    cloudmade = new L.TileLayer(cloudmadeUrl, {maxZoom: 18, attribution: cloudmadeAttrib});
var london = new L.LatLng(51.505, -0.09); // geographical point (longitude and latitude)
map.setView(london, 13).addLayer(cloudmade);
</script>
</head>
<body>
<div id="map" style="height: 200px"></div>
</body>
</html>
```

Figure 18 A quick example of a Leaflet map in use (Leaflet, 2011)

5.5.2 Caching map tiles

Map tiles are small blocks of images containing the graphical part of the map. There are many free sources for map tiles, some come with limitations and some are completely free to use. A great service providing map tiles is Cloudmade. After registering developers are allowed access to existing maps hosted on Cloudmade, but can also customize and create their own styles. Leaflet is a map script developed for displaying mainly Cloudmade maps. Once the developer has chosen which map style the application should use, it is defined in Leaflet using the map URL.

"http://{s}.tile.cloudmade.com/DEVELOPERS-API-KEY/MAP-STYLE-ID/256/{z}/{x}/{y}.png"

The variables with { } tags are constants in the Leaflet API and should not be changed, {s} stands for server, this lets the map API fetch tiles from multiple servers at once to speed up the download process. {z} stands for zoom, this variable is replaced by the current zoom level of the map, {x} & {y} naturally represent the coordinates. 256 in the above example defines the size of each tile in pixels, 256 pixels is the standard.

This functionality is quite straightforward, the API checks where the user is positioned and which zoom level is active, and fetches all the tiles required to fill the screen with graphics from the server. It is however not as simple when there is no connection to the server. For the tiles to become available offline the system first has to calculate which tiles are required (since downloading all tiles is not a possibility). This is done using the algorithm in Figure 19.

```
function mapBounding() {
if(isset($this->mapID)){
$poiCoordinates = $this -> ci -> db -> query('select * from mg_pois where map_id = ' . $this->mapID);
foreach($poiCoordinates->result() as $row) {
		array_push($this->mapPois,$row->poi_id);
		$poiLat = $row -> latitude;
		$poiLon = $row -> longitude;
		$this -> maxLat = ($poiLat > $this -> maxLat || $this -> maxLat == null ? $poiLat : $this -> maxLat);
		$this -> minLat = ($poiLat < $this -> minLat || $this -> minLat == null ? $poiLat : $this -> minLat);
		$this -> maxLon = ($poiLon > $this -> maxLon || $this -> maxLon == null ? $poiLon : $this -> maxLon);
		$this -> minLon = ($poiLon < $this -> minLon || $this -> minLon == null ? $poiLon : $this -> minLon);
}
}
}
function mapTiles($zoom){
if(isset($this->mapID)){
$NWxtile = floor((($this->minLon + 180) / 360) * pow(2, $zoom));
$NWytile = floor((1 - log(tan(deg2rad($this->maxLat)) + 1 / cos(deg2rad($this->maxLat))) / pi()) /2 * pow(2, $zoom));
$SExtile = floor((($this->maxLon + 180) / 360) * pow(2, $zoom));
$SEytile = floor((1 - log(tan(deg2rad($this->minLat)) + 1 / cos(deg2rad($this->minLat))) / pi()) /2 * pow(2, $zoom));
		for ($x = $NWxtile; $x <= $SExtile; $x++) {
		for($y = $NWytile; $y <= $SEytile; $y++){
				$tile = new tile;
				$tile->y = $y;
				$tile->x = $x;
				$tile->z = $zoom;
				 array_push($this->tileArray, $tile);
		}
		}
}
}
```

Figure 19 Algoritm for calculating required map tiles

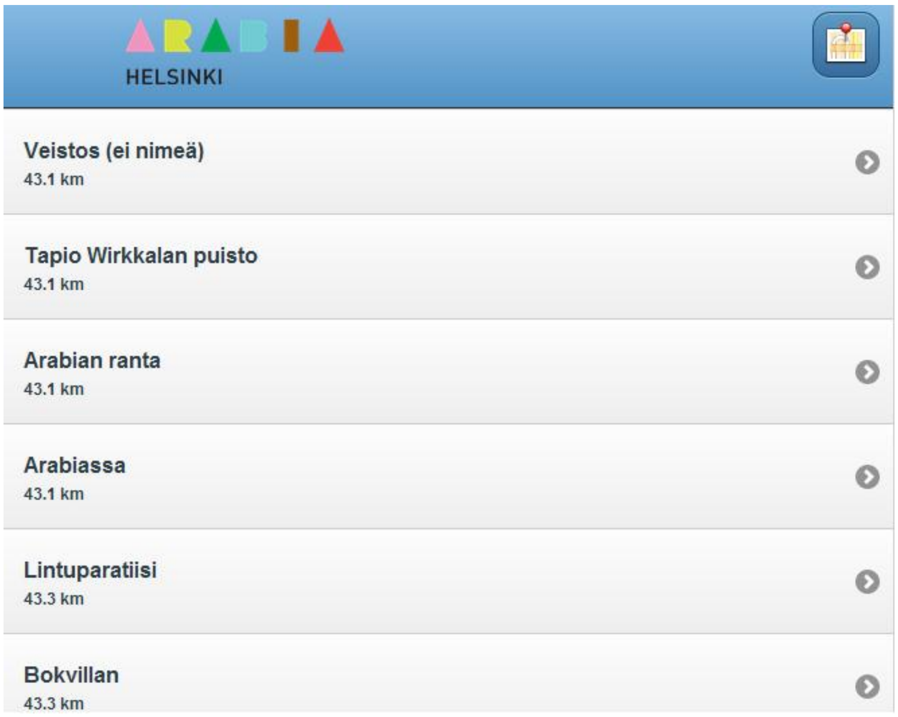
Simply put, the mapBounding function gathers all coordinates defined in the map, and determines the south most, north most, east most and west most points within these co-ordinates. After this the application has a rectangular area as a bounding box, which means that all coordinates within this box must get a tile.

Tiles are defined by splitting up the world in a grid, which means that one tile X,Y contains multiple longitude, latitude points. So for each corner coordinate (Northwest, Northeast, Southeast and Southwest), a corresponding X,Y grid coordinate has to be calculated. After this the application has four corners in a grid system all represented by X,Y coordinates, so it only has to loop through each row and column in the grid system to determine the required tiles, X1,Y1 to Xn,Y1 and so on. All of these coordinates, along with the predefined zoom level are stored in an array, when generating the manifest file (See Dynamic Manifest file chapter) the array is parsed and each element is used to create the full path, /ZOOM/X/Y:

`http://a.tile.cloudmade.com/84b255854c18417b8641e17eb887c074/997/256/16/37314/18957.png`

5.6 List view

The list view is the view loaded up first in the application. It contains all positions of interest, ordered by their distance to the user's position. By looking at the list, users can quickly determine which position of interest is closest to them.



Veistos (ei nimeä)
43.1 km

Tapio Wirkkalan puisto
43.1 km

Arabian ranta
43.1 km

Arabiassa
43.1 km

Lintuparatiisi
43.3 km

Bokvillan
43.3 km

Figure 20 The Mobile Guide list view for the Arabianranta map

From the list view (Figure 20) the user can either jump directly back to the map view by clicking the map icon located in the top right corner, or alternatively have a closer look at any of the positions by selecting them from the list. This opens the POI window, covered in the next chapter.

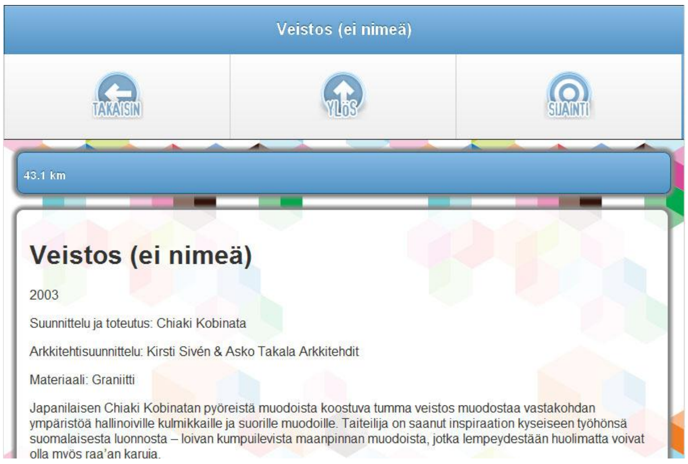
In order for the list to be precise, it has to be updated every time the user moves, since the distances change for every move the user makes. Each list item has data-attributes containing their geographic location, and their distance to the user. Each time the user moves, a function loops through each list item, compares their coordinates to the users and resets their distance value. Once the function has looped through all elements, the list is ordered using jQuery Tinysort (Figure 21).

```
$('.poiListItem').each(function(){
   $(this).attr("data-distance", distanceHaversine(user_lat,user_lon,poi_lat,poi_lon));
}
$('ul.poiList>li').tsort({attr:'data-distance'});
```

Figure 21 A simplified example of the code used to order the list view

5.7 POI window

As seen in Figure 22, the POI window contains all text and image data defined for that position. This window contains three navigational buttons which are fixed to the top of the page, and remain visible even though the page is scrolled. The back button located to the left, takes the user back to the list or map view, depending on which was last used. The center button scrolls the page back up to the top, which can be useful especially when there is a lot of text. The Rightmost button opens up the map view and zooms in on the POI. By combining this with the list view, one can quickly locate a specific POI on the map.



Veistos (ei nimeä)

2003

Suunnittelu ja toteutus: Chiaki Kobinata

Arkkitehtisuunnittelu: Kirsti Sivén & Asko Takala Arkkitehdit

Materiaali: Graniitti

Japanilaisen Chiaki Kobinatan pyöreistä muodoista koostuva tumma veistos muodostaa vastakohdan ympäristöä hallinoiville kulmikkaille ja suorille muodoille. Taiteilija on saanut inspiraation kyseiseen työhönsä suomalaisesta luonnosta – loivan kumpuilevista maanpinnan muodoista, jotka lempeydestään huolimatta voivat olla myös raa'an karuja.

Figure 22 Example of a POI window

Besides text data, POIs can also contain multiple pictures. If there are pictures defined for a POI, the user is displayed an image link, which opens a mobile optimized gallery of the images defined for the position, see Figure 23 for an example. The gallery listens to touch interactions, so the user can swipe through the images like in the native galleries found in touch based operating systems.

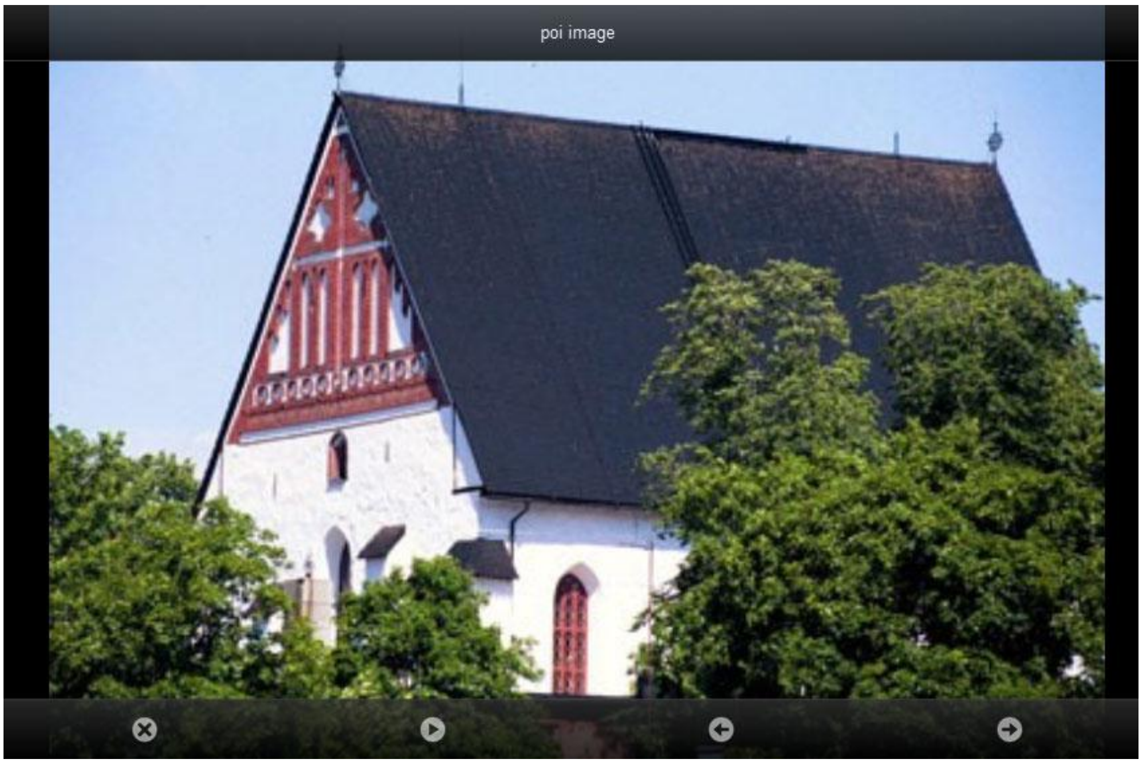


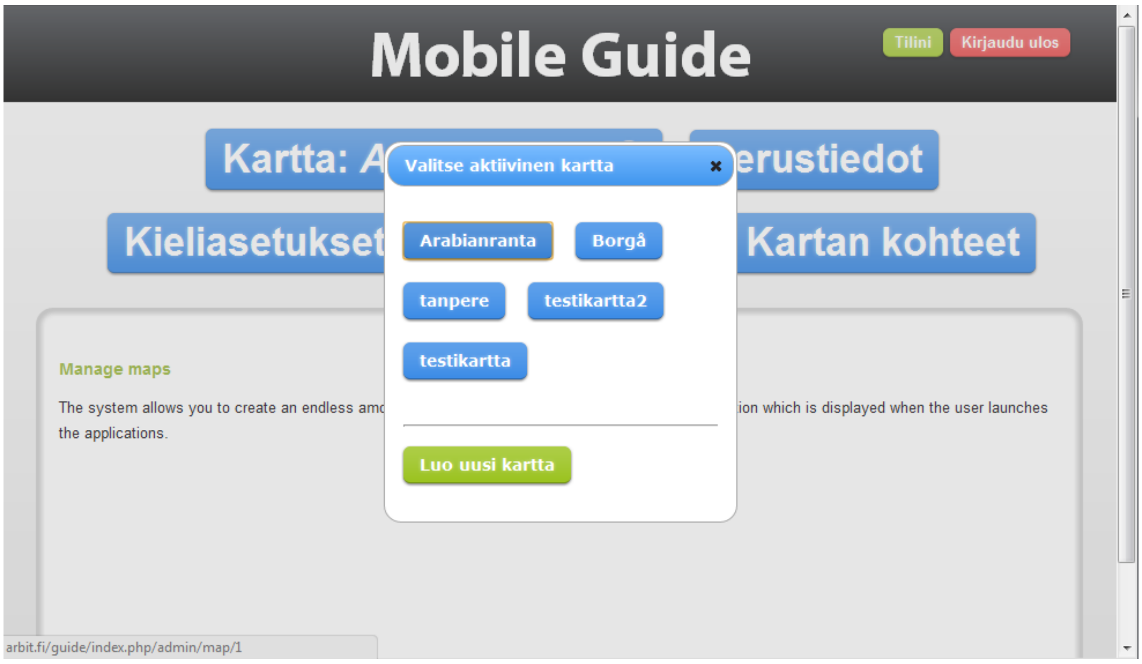
Figure 23 The POI gallery

6 THE ADMIN INTERFACE

The admin interface consists of multiple pages for managing maps and POIs. The system is built to function as a service which means that anyone could register and start creating maps and POIs of their own. Figure 24 contains a screenshot taken from the admin interface with multiple maps already defined.

Because the target user group of the admin interface would be companies and tourist departments there needs to be a way to personalize and brand the maps according to the organizations branding and needs. This is why the admin interface also contains functionality for customizing the color schemes and logos for each map.

In order to broaden the user-base the admin interface also lets the administrator(s) define available map languages, which means companies in any country could start offering their own mobile guide for their guests, in any language. Once a language is defined for the map, every POI can be translated in the new language. In case the translation is not given for a specific position, the frontend text will be displayed in the default language, which is also defined by the administrator and can be changed at any time.

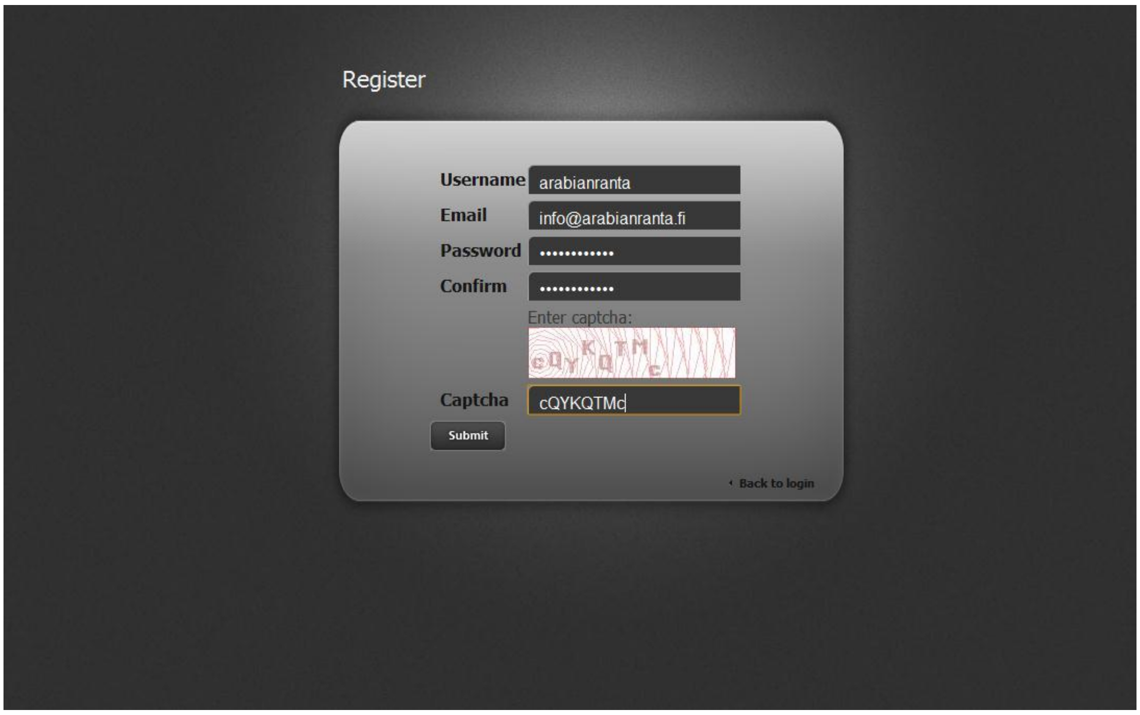


The system allows you to create an endless amo

ion which is displayed when the user launches the applications.

Figure 24 A screenshot of the admin interface

6.1 Registration and authentication



Username: arabianranta
Email: info@arabianranta.fi
Password: ••••••••••••
Confirm: ••••••••••••
Captcha: cQYKQTMc

Figure 25 Login and Registration page for Mobile Guide

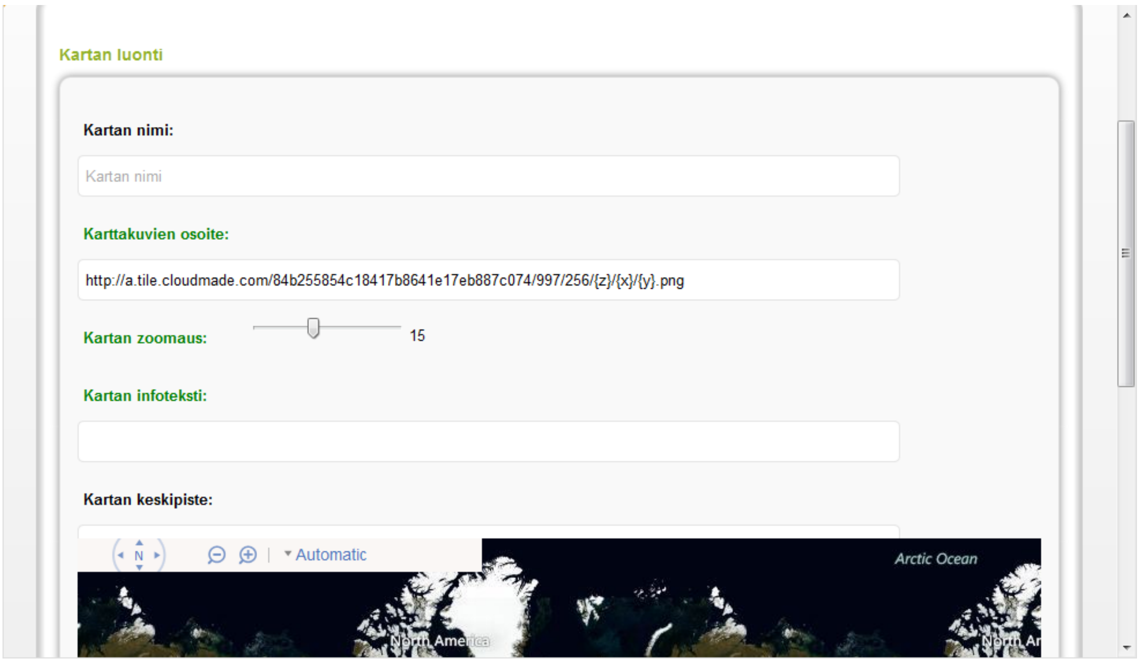
In order to start creating and modifying a map, users are required to register to the service through the registration form (Figure 25). The registration and authentication are handled by a Codeigniter library called Tank Auth. Upon registering, the users are sent an email to confirm their registration. After this they are able to login to the service and start creating maps.

When handling passwords Tank Auth uses the Portable PHP password hashing framework for added security, which basically means that passwords are never transferred to or from the database without first being hashed by OpenBSD-style Blowfish-based bcrypt, which is considered more safe then md5, sha1 and other hashing functions.

Tank Auth handles the login and registration of users, but does not directly control user privileges. For this a special library called “Can_access” was built. Each time an administrator performs an operation on a map, POI or even an image, Can_Access checks if this user has the right to make changes to that particular item.

6.2 Map creation and selection

Once the user has successfully registered and logged in, the main view is shown. From this view the user can change and create an endless amount of maps. Once the user decides to add a new map, the form shown in Figure 26 is opened, where all the general information over the map is given.



Kartan luonti

Kartan nimi:

Kartan nimi

Karttakuvien osoite:

http://a.tile.cloudmade.com/84b255854c18417b8641e17eb887c074/997/256/{z}/{x}/{y}.png

Kartan zoomaus: 15

Kartan infoteksti:

Kartan keskipiste:

Figure 26 Screenshot of the map creation form

All of the options given in the map creation process are customizable at a later stage, but are initially required in order to proceed with the creation of POIs. The first line, the map name is used both in the backend and the frontend, letting the user know which map is currently being edited or viewed.

The second line tells Mobile Guide, which map tiles should be used. There are some free tile service providers around the Internet that can be used. As an example, MapQuest provides Open Aerial tiles through the following url:
http://oatile1.mqcdn.com/naip/15/5240/12661.jpg
To use MapQuests Aerial tiles, the last three url segments have to be changed to zoom {z}, X coordinate {x} and Y coordinate {y}. So the tile url would in this case become:
http://oatile1.mqcdn.com/naip/{z}/{x}/{y}.jpg

The default tile server used is generated through a service called Cloudmade, where users can, after a free registration, style their maps exactly as they please. Cloudmade provides instructions on how to do this on their website *cloudmade.com*.

The third line in the map creation determines the zoom level for the map, 18 being the maximal value which is the closest possible zoom. This number is not recommended as the number of required tiles would increase to thousands. This in turn would greatly affect the download time when using the application in offline mode.

Finally, the map center, referred to as the info point needs to be set. This is done be dragging the marker to its place using the map widget.

6.3 Map languages

Map languages are used for presenting data in multiple languages, which is important if there is an international user base. After the map has been created, the map languages have to be defined through the map language definition form (Figure 27), since it is not possible to create POIs unless the map contains at least one language.

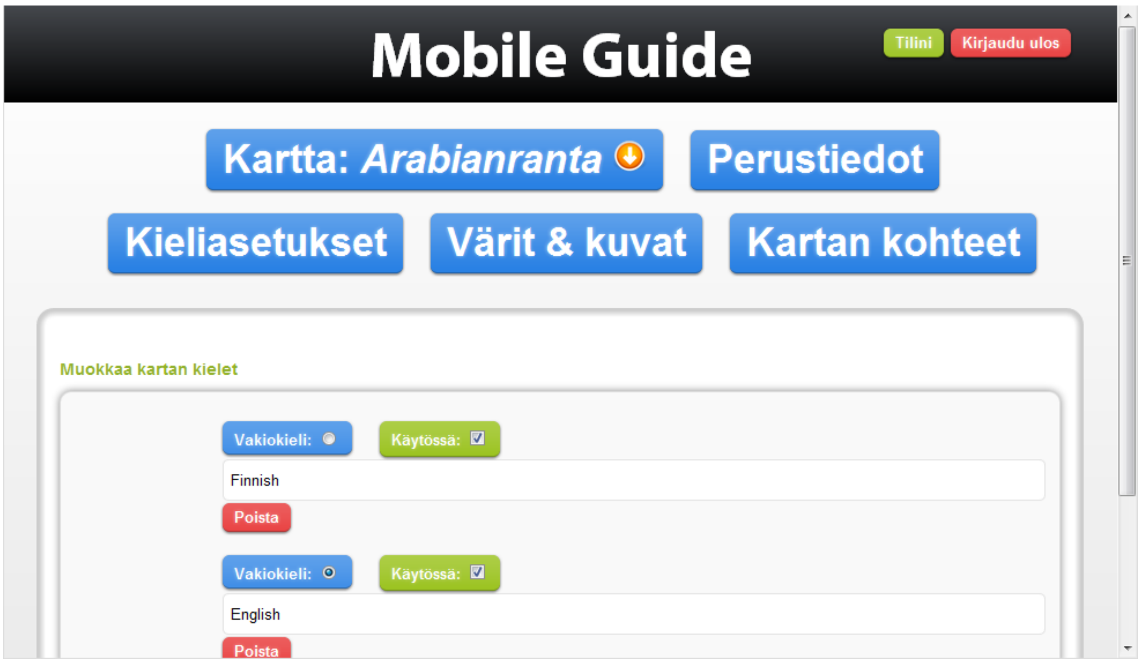


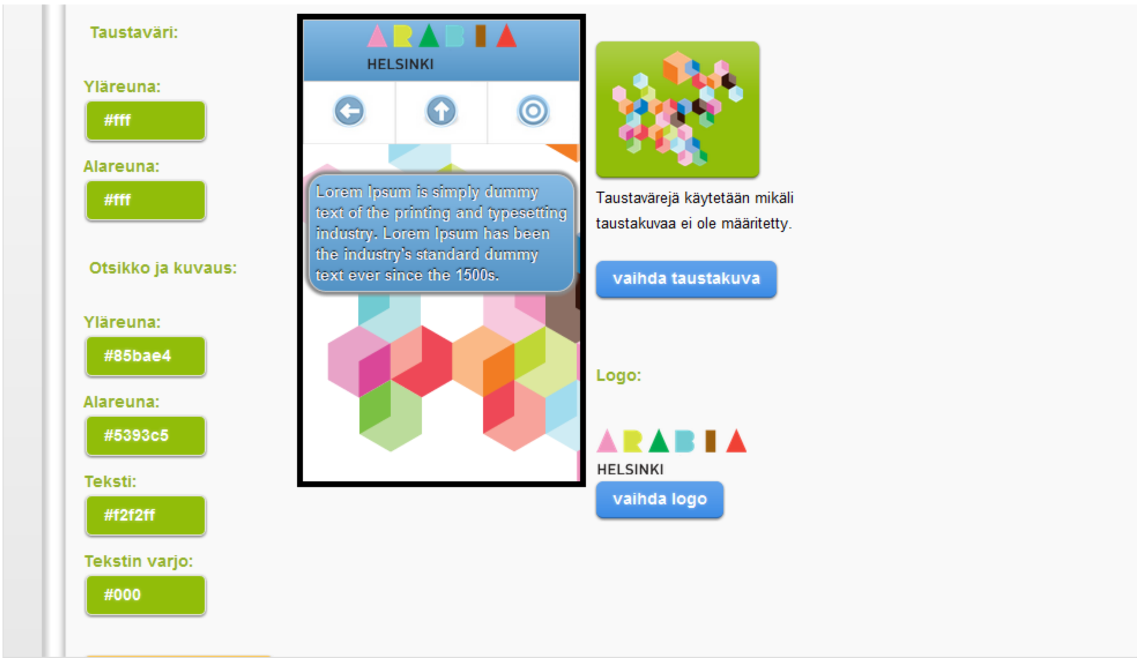
Figure 27 Screenshot of the map language definition form

The administrator can add, rename and remove languages. It is however important to note that once a language is removed, the translations connected to that language will no longer be available in the front-end.

One language is always defined as the master language of the map. The master language is used to substitue missing translations, so if the user choses Finnish as the display language, and a specific POI is not translated into Finnish, the master language English would be used to present the data. In case a translation is not provided in the master language nor in the currently selected language, the POI will be completely hidden from the end user.

6.4 Map styling

The map styling page enables map administrators to customize the frontends color scheme, logos and background images to their liking (Figure 28). This is a great way to brand the map and improve the overall experience.



Taustaväri:

Yläreuna: #fff

Alareuna: #fff

Otsikko ja kuvaus:

Yläreuna: #85bae4

Alareuna: #5393c5

Teksti: #f2f2ff

Tekstin varjo: #000

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s.

Taustavärejä käytetään mikäli taustakuvaa ei ole määritetty.

Logo:

Figure 28 The Mobile Guide style editor

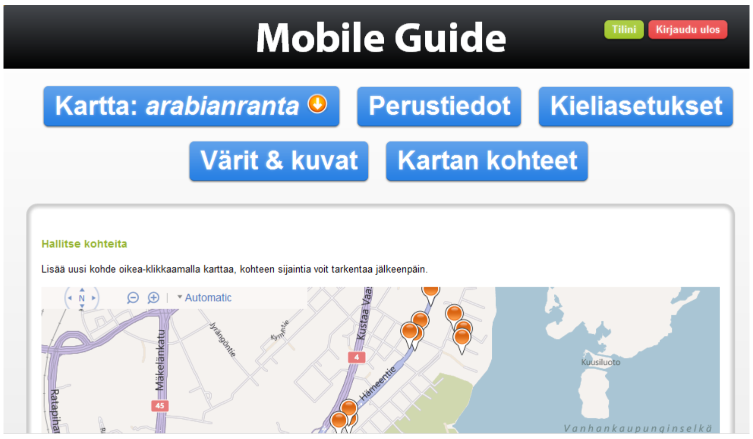
All colors are chosen using a RGB color picker. The header, info box and background can be assigned a background color or gradient color. The background can alternatively be applied an image instead of colors. The header image can be set by uploading a logo. If it's not set the map name will be displayed in text format.

To ease the process of styling the map, a preview widget is included. The widget is updated in real time by resetting its CSS every time the administrator changes a color or uploads an image. The code responsible for updating the widget CSS is presented in Figure 29.

```
$('#backgroundPicker').ColorPicker({
  onSubmit: function(hsb, hex, rgb, el) {
    $(el).val(hex);
    $(el).ColorPickerHide();
  },
  onBeforeShow: function () {
    $(this).ColorPickerSetColor(this.value);
  },
  onChange: function (hsb, hex, rgb) {
   $('#colorpickerField1').val("#"+hex);
   bgstart = hex;
   repaintBackground();
  }
})

function repaintBackground(){
    $.support.backgroundLinearGradient = (function() {
        var test = $('#previewWindow');
        $('.bgimg').addClass("red");
        test.css('background-image', 'none');
        test.css('background-image', 'linear-gradient(top, #'+bgstart+', #'+bgend+')');
        if(test.css('background-image') != 'none') { return "linear-gradient"; }
        test.css('background-image', '-moz-linear-gradient(top, #'+bgstart+', #'+bgend+')');
        if(test.css('background-image') != 'none') { return "-moz-linear-gradient"; }
        test.css('background-image', '-webkit-linear-gradient(top, #'+bgstart+', #'+bgend+')');
        if(test.css('background-image') != 'none') { return "-webkit-linear-gradient"; }
        test.css('background-image', '-o-linear-gradient(top, #'+bgstart+', #'+bgend+')');
        if(test.css('background-image') != 'none') { return "-o-linear-gradient"; }
        test.css('background-image', '-ms-linear-gradient(top, #'+bgstart+', #'+bgend+')');
        if(test.css('background-image') != 'none') { return "-ms-linear-gradient"; }
        return false;
    })();

}
```

Figure 29 Color update in real time

6.5 POI

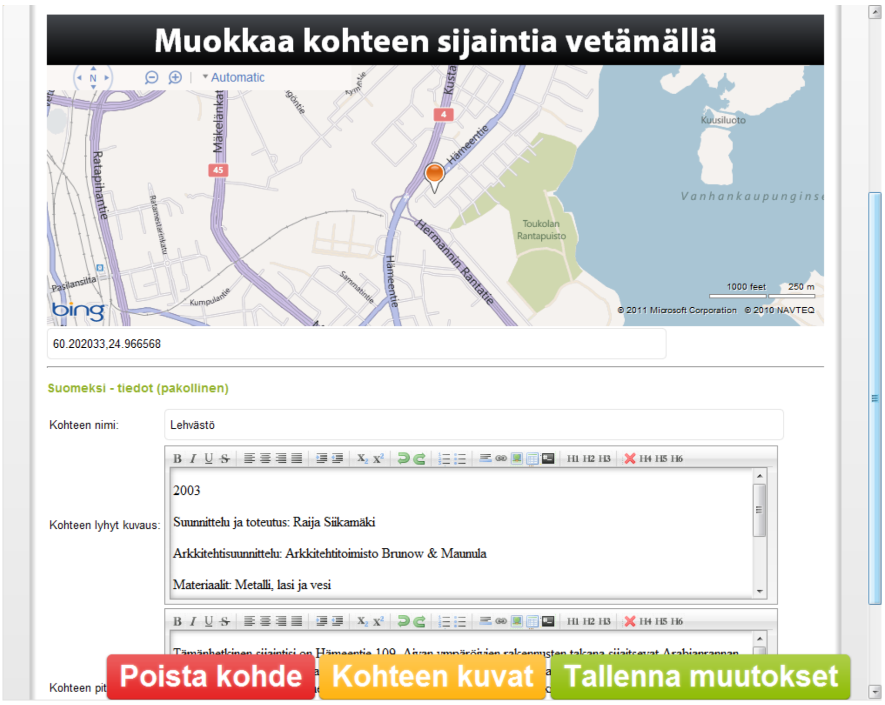


Hallitse kohteita

Lisää uusi kohde oikea-klikkaamalla karttaa, kohteen sijaintia voit tarkentaa jälkeenpäin.

Figure 30 The POI editors main view, all defined POIS are accessible from this map

The most important functionality in the admin interface is the ability to create and modify the map’s POIs. The POI editor consists of a main map (Figure 30), where all the map POIs are painted. To create a new POI, the administrator right clicks the desired position on the map, and from the dropdown menu select to add a new POI. After this the POI definition page is opened (Figure 31).



60.202033,24.966568

Suomeksi - tiedot (pakollinen)

Kohteen nimi:	Lehvästö
Kohteen lyhyt kuvaus:	2003 Suunnittelu ja toteutus: Raija Siikamäki Arkkitehtisuunnittelu: Arkkitehtitoimisto Brunow & Maunula Materiaalit: Metalli, lasi ja vesi
Kohteen pit	Tämänhetkinen sijaintisi on Hämeentie 109. Aivan ympäröivien rakennusten takana sijaitsevat Arabianrannan

Figure 31 The POI definition page, all text and image data is entered through this page

Each POI can be defined a name, a location, a short description and a more in-depth description. The location can be changed at any time by dragging the marker on the map widget. The administrator can also open the image handler (Figure 32) from the POI definition page. From the image handler the administrator has the possibility to manage and add multiple images to a POI, keeping in mind that each image increases the download time for the end-user. As a precaution, the server side script automatically resizes all uploaded images to a maximum width of 500 pixels in order to limit the total size of the map.

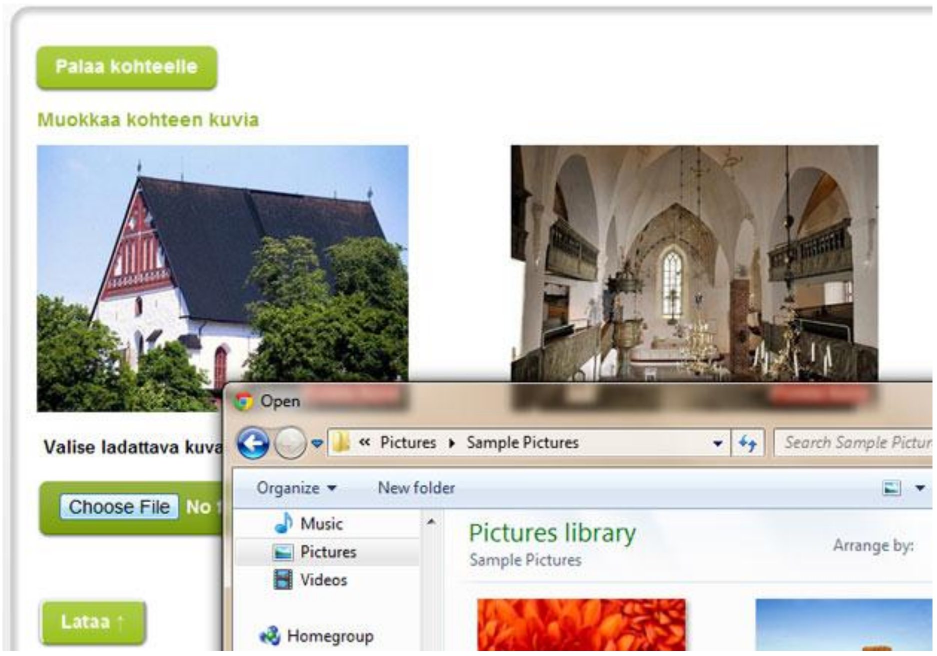


Figure 32 The image handler, here the administrator can add and remove POI images

7 DISCUSSION

7.1 Application testing

The actual content for ADC was created by another student as a separate thesis (Krista Fransman, 2011), who used the admin interface for storing the data. Many flaws were found and fixed due to extensive testing. In order to ensure that the frontend application was working as expected a real life test was conducted. Students of Arcada were recruited for the test. They formed groups, and walked around Arabianranta with Mobile Guide running in one of the group member’s phone. As it was not possible to test the application on all possible phones during the development stage, the test revealed some unexpected flaws in a few older phones.

The technical comments were mostly related to the missing zoom capability, which was left out on purpose to decrease the download time. Another desired functionality was routing, which was disabled before the test for performance issues, but is possible to implement from a technical point of view. Some devices like the iPhone 3 had problems with the Leaflet map widget. This has been reported to the maintainers and developers of Leaflet and will hopefully be fixed in the near future.

Besides writing down the technical experience the students were also asked to comment on the content, since this is going to be used along with the application itself to guide tourists during Helsinki WDC 2012.

7.2 Further development

Even though the application is at a stage where it can and will be deployed in real environments, there are countless of possibilities to improve the functionality. On the backend, a map size meter should be implemented. This however is not an easy task as counting the total size of the map tiles is quite complex. The frontend should undergo extensive testing on multiple old and new devices to ensure maximal compatibility.

As destination routing and zoom capability was the functionality that most test users were craving for, these could be worth implementing. Actual turn-by-turn routing would not work in offline mode, but painting a straight line between the user and his/her destination is possible. The zoom functionality could be implemented by providing the end-user with two different zoom levels.

7.3 Conclusion

The development of Mobile Guide went through many phases. In the beginning the goal was to develop a tour guide using a programming framework called Qt. This was however quickly dropped as Nokia decided to stop developing Symbian, Maemo and Meego phones, the only mobile operating systems that supported Qt.

After the decision of not developing in Qt was made, many other platforms, such as Android and IOS were considered. Phonegap, an HTML5 platform which allows developers to create native applications in HTML5 was evaluated and tested. Phonegap was a great alternative, but it quickly became clear that it didn’t bring any added value to the application, as all the required functionality found in Phonegap was available in HTML5 itself. HTML5 is a safe choice as most mobile browsers already support it, and all upcoming browser updates will definitely be HTML5 related.

I now have a great understanding of HTML5 and its possibilities, which are close to limitless. HTML5 is going to be the base for future projects as it is great to work with and is not platform dependent.

REFERENCES

Codeigniter 2011, CodeIgniter User Guide Version 2.1.0

[www] Retrieved 03.05.2011.

Fount at: http://codeigniter.com/user_guide/

Maximiliano Firtman 2011. Mobile HTML5

[www] [Retrieved 08.12.2011]

Found at: http://mobilehtml5.org/

Mark Pilgrim 2011. DIVE INTO HTML5

[www] [Retrieved 08.12.2011]

Found at: http://diveintohtml5.info/

Leaflet 2011, Leaflet documentation

[www] [Retrieved 03.09.2011]

Found at: http://leaflet.cloudmade.com/reference.html

jQueryMobile 2011, jQueryMobile documentation Version 1.0

[www] [Retrieved 09.10.2011]

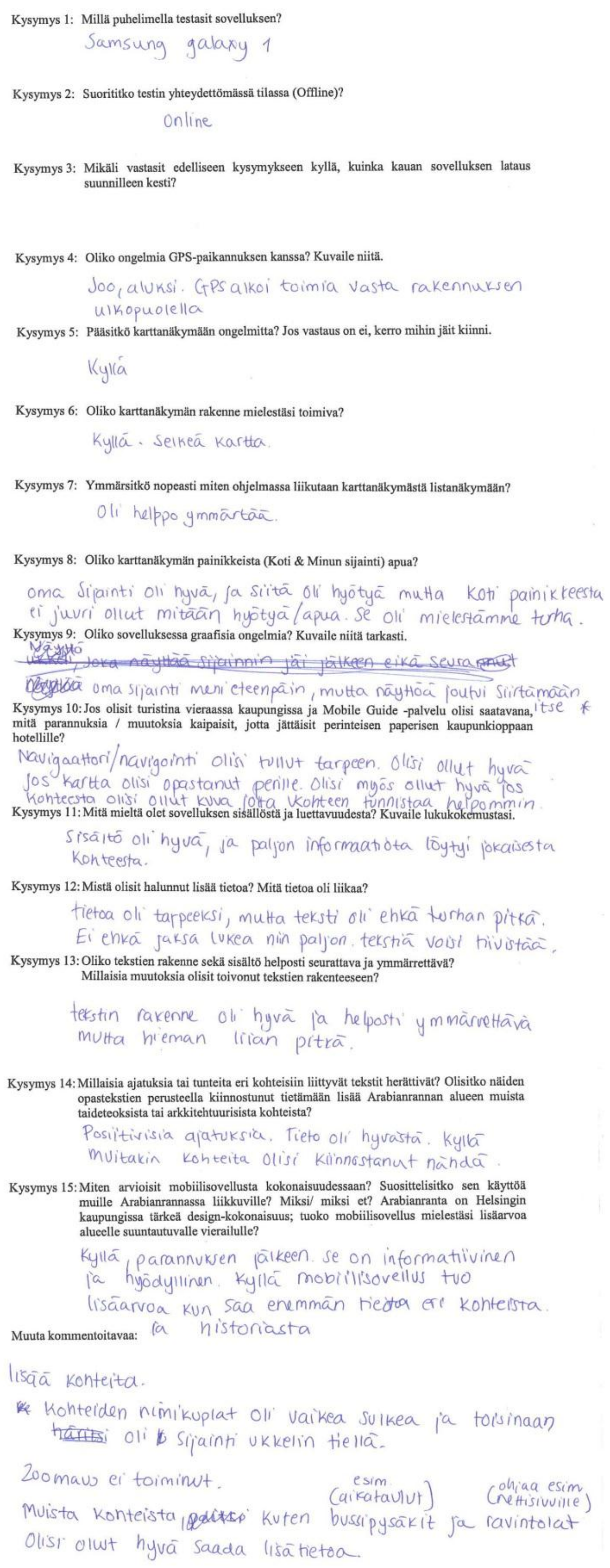
Found at: http://jquerymobile.com/demos/1.0/

Krista Fransman 2011, Matkaoppaana Arabianrannassa toimiva mobiilisovellus - tehtävänä sisällöntuotanto

BSC Thesis Manuscript

APPENDENCIES

- APPENDICE: Test feedback 1.



Kysymys 1: Millä puhelimella testasit sovelluksen?

Samsung galaxy 1

Kysymys 2: Suorititko testin yhteydettömässä tilassa (Offline)?

Online

Kysymys 3: Mikäli vastasit edelliseen kysymykseen kyllä, kuinka kauan sovelluksen lataus suunnilleen kesti?

Kysymys 4: Oliko ongelmia GPS-paikannuksen kanssa? Kuvaile niitä.

Joo, aluksi. GPS alkoi toimia vasta rakennuksen ulkopuolella

Kysymys 5: Pääsitkö karttanäkymään ongelmitta? Jos vastaus on ei, kerro mihin jäit kiinni.

Kyllä

Kysymys 6: Oliko karttanäkymän rakenne mielestäsi toimiva?

Kyllä. Selkeä kartta.

Kysymys 7: Ymmärsitkö nopeasti miten ohjelmassa liikutaan karttanäkymästä listanäkymään?

Oli helppo ymmärtää.

Kysymys 8: Oliko karttanäkymän painikkeista (Koti & Minun sijainti) apua?

oma sijainti oli hyvä, ja siitä oli hyötyä mutta koti painikkeesta ei juuri ollut mitään hyötyä/apua. Se oli mielestämme turha.

Kysymys 9: Oliko sovelluksessa graafisia ongelmia? Kuvaile niitä tarkasti.

Näyttö
~~ukkeli, joka näyttää sijainnin jäi jälkeen eikä seurannut~~
~~[illegible]~~ oma sijainti meni eteenpäin, mutta näyttöä joutui siirtämään itse *

Kysymys 10: Jos olisit turistina vieraassa kaupungissa ja Mobile Guide -palvelu olisi saatavana, mitä parannuksia / muutoksia kaipaisit, jotta jättäisit perinteisen paperisen kaupunkioppaan hotellille?

Navigaattori/navigointi olisi tullut tarpeen. Olisi ollut hyvä jos kartta olisi opastanut perille. Olisi myös ollut hyvä jos kohteesta olisi ollut kuva jotta kohteen tunnistaa helpommin.

Kysymys 11: Mitä mieltä olet sovelluksen sisällöstä ja luettavuudesta? Kuvaile lukukokemustasi.

Sisältö oli hyvä, ja paljon informaatiota löytyi jokaisesta kohteesta.

Kysymys 12: Mistä olisit halunnut lisää tietoa? Mitä tietoa oli liikaa?

tietoa oli tarpeeksi, mutta teksti oli ehkä turhan pitkä. Ei ehkä jaksa lukea niin paljon. tekstiä voisi tiivistää.

Kysymys 13: Oliko tekstien rakenne sekä sisältö helposti seurattava ja ymmärrettävä? Millaisia muutoksia olisit toivonut tekstien rakenteeseen?

tekstin rakenne oli hyvä ja helposti ymmärrettävä mutta hieman liian pitkä.

Kysymys 14: Millaisia ajatuksia tai tunteita eri kohteisiin liittyvät tekstit herättivät? Olisitko näiden opastekstien perusteella kiinnostunut tietämään lisää Arabianrannan alueen muista taideteoksista tai arkkitehtuurisista kohteista?

Positiivisia ajatuksia. Tieto oli hyvästä. Kyllä muitakin kohteita olisi kiinnostanut nähdä.

Kysymys 15: Miten arvioisit mobiilisovellusta kokonaisuudessaan? Suosittelisitko sen käyttöä muille Arabianrannassa liikkuville? Miksi/ miksi et? Arabianranta on Helsingin kaupungissa tärkeä design-kokonaisuus; tuoko mobiilisovellus mielestäsi lisäarvoa alueelle suuntautuvalle vierailulle?

Kyllä, parannuksen jälkeen. se on informatiivinen ja hyödyllinen. Kyllä mobiilisovellus tuo lisäarvoa kun saa enemmän tietoa eri kohteista. ja historiasta

Muuta kommentoitavaa:

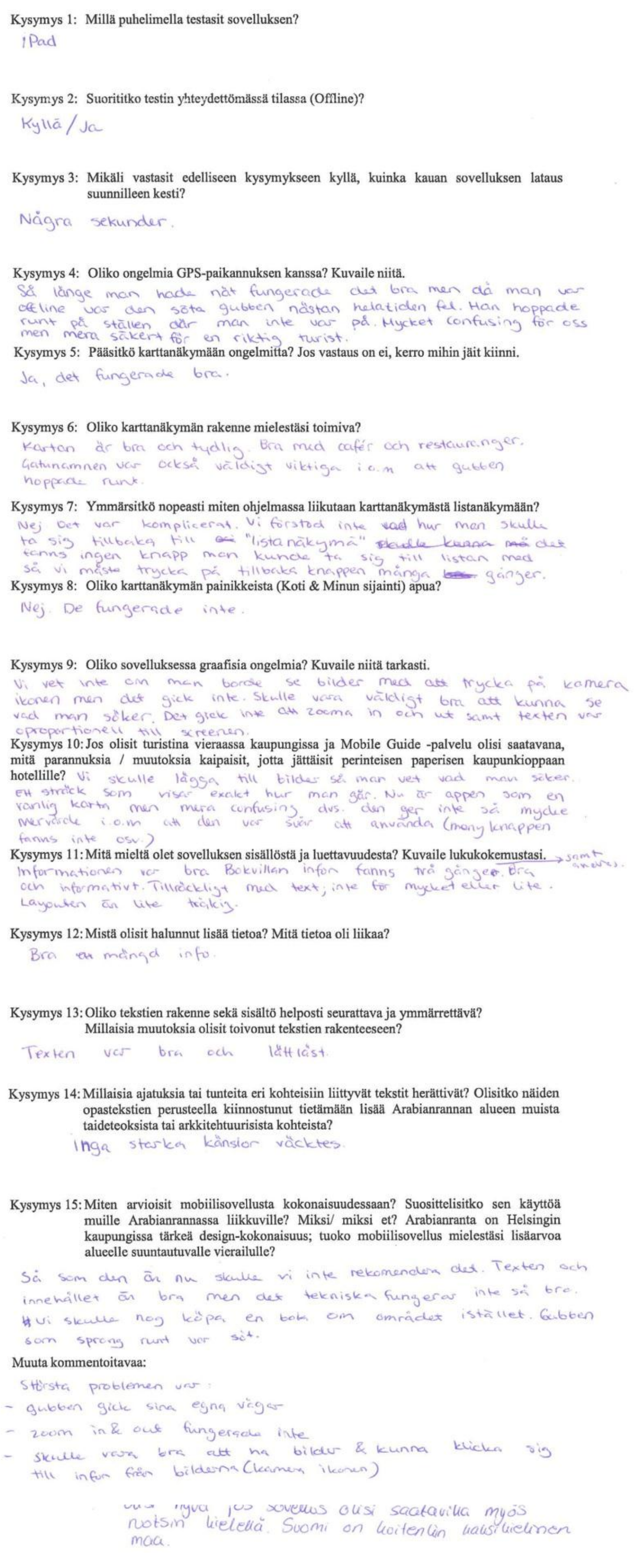
lisää kohteita.

* kohteiden nimikuplat oli vaikea sulkea ja toisinaan ~~[illegible]~~ oli sijainti ukkelin tiellä.

Zoomaus ei toiminut.

Muista kohteista ~~[illegible]~~ kuten (esim aikataulut) bussipysäkit ja ravintolat (ohjaa esim nettisivuille) Olisi ollut hyvä saada lisätietoa.

- APPENDICE: Test feedback 2.



Kysymys 1: Millä puhelimella testasit sovelluksen?

iPad

Kysymys 2: Suorititko testin yhteydettömässä tilassa (Offline)?

Kyllä / Ja

Kysymys 3: Mikäli vastasit edelliseen kysymykseen kyllä, kuinka kauan sovelluksen lataus suunnilleen kesti?

Några sekunder.

Kysymys 4: Oliko ongelmia GPS-paikannuksen kanssa? Kuvaile niitä.

Så länge man hade nät fungerade det bra men då man var offline var den söta gubben nästan helatiden fel. Han hoppade runt på ställen där man inte var på. Mycket confusing för oss men mera säkert för en riktig turist.

Kysymys 5: Pääsitkö karttanäkymään ongelmitta? Jos vastaus on ei, kerro mihin jäit kiinni.

Ja, det fungerade bra.

Kysymys 6: Oliko karttanäkymän rakenne mielestäsi toimiva?

Kartan är bra och tydlig. Bra med cafér och restauranger. Gatunamnen var också väldigt viktiga i.o.m att gubben hoppade runt.

Kysymys 7: Ymmärsitkö nopeasti miten ohjelmassa liikutaan karttanäkymästä listanäkymään?

Nej Det var komplicerat. Vi förstod inte ~~vad~~ hur man skulle ta sig tillbaka till ~~en~~ "listanäkymä" ~~skulle kunna må~~ det fanns ingen knapp man kunde ta sig till listan med så vi måste trycka på tillbaka knappen många ~~[illegible]~~ gånger.

Kysymys 8: Oliko karttanäkymän painikkeista (Koti & Minun sijainti) apua?

Nej. De fungerade inte.

Kysymys 9: Oliko sovelluksessa graafisia ongelmia? Kuvaile niitä tarkasti.

Vi vet inte om man borde se bilder med ~~att~~ trycka på kamera ikonen men det gick inte. Skulle vara väldigt bra att kunna se vad man söker. Det gick inte att zooma in och ut samt texten var oproportionell till screenen.

Kysymys 10: Jos olisit turistina vieraassa kaupungissa ja Mobile Guide -palvelu olisi saatavana, mitä parannuksia / muutoksia kaipaisit, jotta jättäisit perinteisen paperisen kaupunkioppaan hotellille? Vi skulle lägga till bilder så man vet vad man söker. Ett sträck som visar exakt hur man går. Nu är appen som en vanlig karta men mera confusing dvs. den ger inte så mycke mervärde i.o.m att den var svår att använda (meny knappen fanns inte osv.)

Kysymys 11: Mitä mieltä olet sovelluksen sisällöstä ja luettavuudesta? Kuvaile lukukokemustasi. → samt andres.

Informationen var bra Bokvillan infon fanns två gånger. Bra och informativt. Tillräckligt med text; inte för mycket eller lite. Layouten är lite tråkig.

Kysymys 12: Mistä olisit halunnut lisää tietoa? Mitä tietoa oli liikaa?

Bra ~~en~~ mängd info.

Kysymys 13: Oliko tekstien rakenne sekä sisältö helposti seurattava ja ymmärrettävä? Millaisia muutoksia olisit toivonut tekstien rakenteeseen?

Texten var bra och lättläst.

Kysymys 14: Millaisia ajatuksia tai tunteita eri kohteisiin liittyvät tekstit herättivät? Olisitko näiden opastekstien perusteella kiinnostunut tietämään lisää Arabianrannan alueen muista taideteoksista tai arkkitehtuurisista kohteista?

Inga starka känslor väcktes.

Kysymys 15: Miten arvioisit mobiilisovellusta kokonaisuudessaan? Suosittelisitko sen käyttöä muille Arabianrannassa liikkuville? Miksi/ miksi et? Arabianranta on Helsingin kaupungissa tärkeä design-kokonaisuus; tuoko mobiilisovellus mielestäsi lisäarvoa alueelle suuntautuvalle vierailulle?

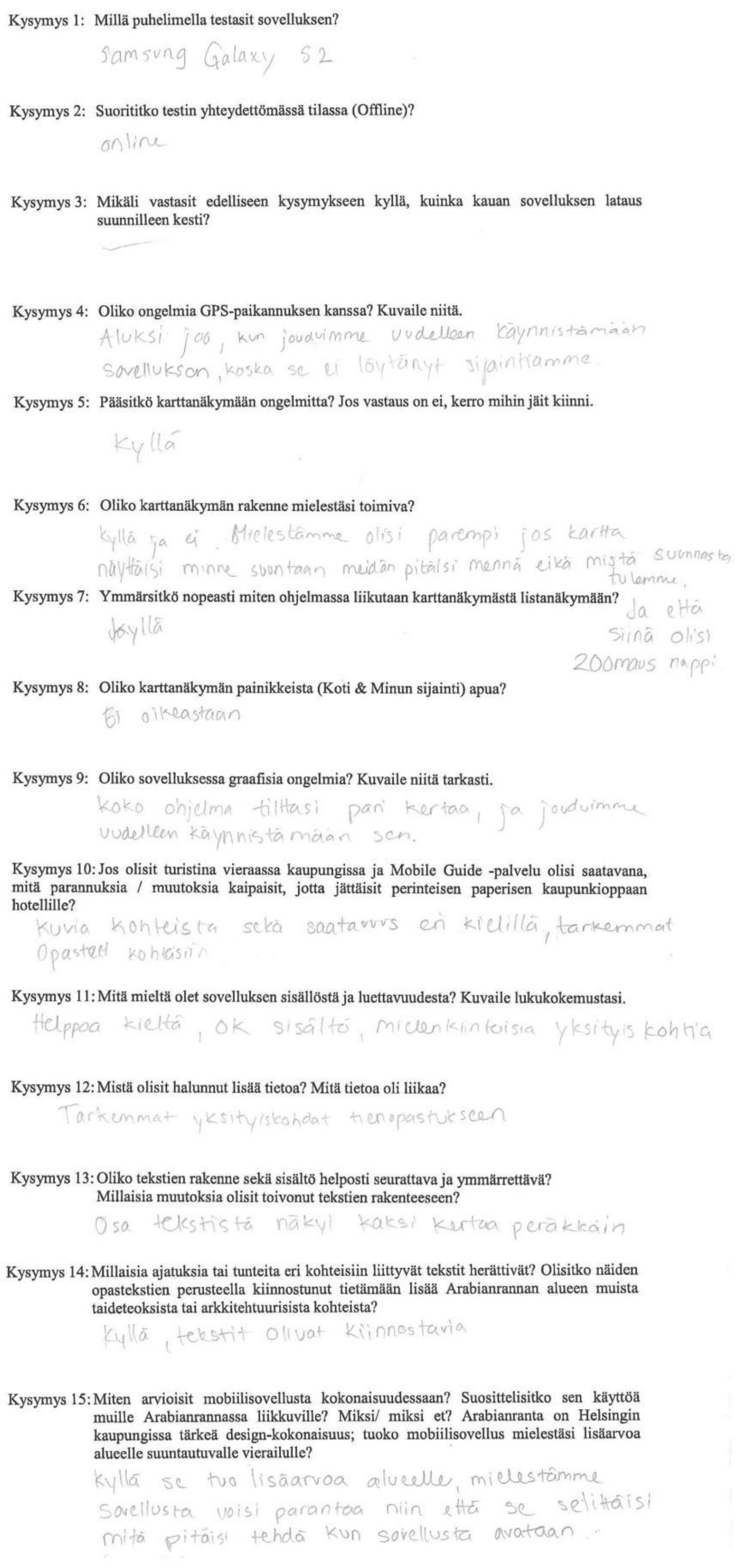
Så som den är nu skulle vi inte rekomendera det. Texten och innehållet är bra men det tekniska fungerar inte så bra. ~~#~~ Vi skulle nog köpa en bok om området istället. Gubben som sprang runt var söt.

Muuta kommentoitavaa:

Största problemen var:
- gubben gick sina egna vägar
- zoom in & out fungerade inte
- skulle vara bra att ha bilder & kunna klicka sig till infon från bilderna (kamera ikonen)

[illegible] hyvä jos sovellus olisi saatavilla myös ruotsin kielellä. Suomi on kuitenkin kaksikielinen maa.

- APPENDICE: Test feedback 3.



Kysymys 1: Millä puhelimella testasit sovelluksen?

Samsung Galaxy S2

Kysymys 2: Suorititko testin yhteydettömässä tilassa (Offline)?

online

Kysymys 3: Mikäli vastasit edelliseen kysymykseen kyllä, kuinka kauan sovelluksen lataus suunnilleen kesti?

—

Kysymys 4: Oliko ongelmia GPS-paikannuksen kanssa? Kuvaile niitä.

Aluksi joo, kun jouduimme uudelleen käynnistämään sovelluksen, koska se ei löytänyt sijaintiamme.

Kysymys 5: Pääsitkö karttanäkymään ongelmitta? Jos vastaus on ei, kerro mihin jäit kiinni.

Kyllä

Kysymys 6: Oliko karttanäkymän rakenne mielestäsi toimiva?

kyllä ja ei. Mielestämme olisi parempi jos kartta näyttäisi minne suuntaan meidän pitäisi mennä eikä mistä suunnasta tulemme, Ja että siinä olisi zoomaus nappi.

Kysymys 7: Ymmärsitkö nopeasti miten ohjelmassa liikutaan karttanäkymästä listanäkymään?

Kyllä

Kysymys 8: Oliko karttanäkymän painikkeista (Koti & Minun sijainti) apua?

Ei oikeastaan

Kysymys 9: Oliko sovelluksessa graafisia ongelmia? Kuvaile niitä tarkasti.

koko ohjelma tilttasi pari kertaa, ja jouduimme uudelleen käynnistämään sen.

Kysymys 10: Jos olisit turistina vieraassa kaupungissa ja Mobile Guide -palvelu olisi saatavana, mitä parannuksia / muutoksia kaipaisit, jotta jättäisit perinteisen paperisen kaupunkioppaan hotellille?

Kuvia kohteista sekä saatavuus eri kielillä, tarkemmat Opasteet kohteisiin

Kysymys 11: Mitä mieltä olet sovelluksen sisällöstä ja luettavuudesta? Kuvaile lukukokemustasi.

Helppoa kieltä, OK sisältö, mielenkiintoisia yksityiskohtia

Kysymys 12: Mistä olisit halunnut lisää tietoa? Mitä tietoa oli liikaa?

Tarkemmat yksityiskohdat tienopastukseen

Kysymys 13: Oliko tekstien rakenne sekä sisältö helposti seurattava ja ymmärrettävä? Millaisia muutoksia olisit toivonut tekstien rakenteeseen?

Osa tekstistä näkyi kaksi kertaa peräkkäin

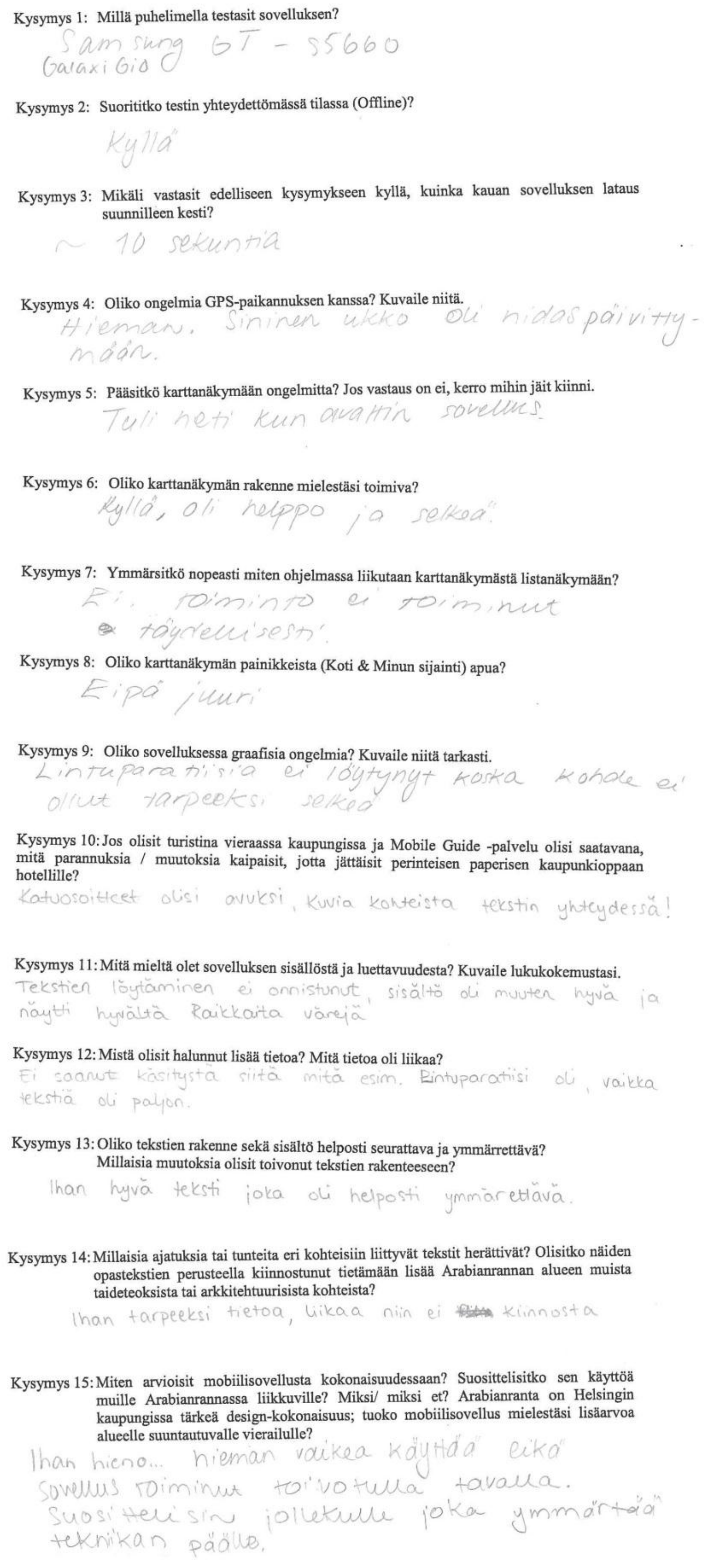
Kysymys 14: Millaisia ajatuksia tai tunteita eri kohteisiin liittyvät tekstit herättivät? Olisitko näiden opastekstien perusteella kiinnostunut tietämään lisää Arabianrannan alueen muista taideteoksista tai arkkitehtuurisista kohteista?

Kyllä, tekstit olivat kiinnostavia

Kysymys 15: Miten arvioisit mobiilisovellusta kokonaisuudessaan? Suosittelisitko sen käyttöä muille Arabianrannassa liikkuville? Miksi/ miksi et? Arabianranta on Helsingin kaupungissa tärkeä design-kokonaisuus; tuoko mobiilisovellus mielestäsi lisäarvoa alueelle suuntautuvalle vierailulle?

Kyllä se tuo lisäarvoa alueelle, mielestämme sovellusta voisi parantaa niin että se selittäisi mitä pitäisi tehdä kun sovellusta avataan.

- APPENDICE: Test feedback 4



Kysymys 1: Millä puhelimella testasit sovelluksen?

Samsung GT - S5660
Galaxi Gio

Kysymys 2: Suorititko testin yhteydettömässä tilassa (Offline)?

Kyllä

Kysymys 3: Mikäli vastasit edelliseen kysymykseen kyllä, kuinka kauan sovelluksen lataus suunnilleen kesti?

~ 10 sekuntia

Kysymys 4: Oliko ongelmia GPS-paikannuksen kanssa? Kuvaile niitä.

Hieman. Sininen ukko oli hidas päivittymään.

Kysymys 5: Pääsitkö karttanäkymään ongelmitta? Jos vastaus on ei, kerro mihin jäit kiinni.

Tuli heti kun avattin sovellus.

Kysymys 6: Oliko karttanäkymän rakenne mielestäsi toimiva?

Kyllä, oli helppo ja selkeä.

Kysymys 7: Ymmärsitkö nopeasti miten ohjelmassa liikutaan karttanäkymästä listanäkymään?

Ei. Toiminto ei toiminut täydellisesti.

Kysymys 8: Oliko karttanäkymän painikkeista (Koti & Minun sijainti) apua?

Eipä juuri

Kysymys 9: Oliko sovelluksessa graafisia ongelmia? Kuvaile niitä tarkasti.

Lintuparatiisia ei löytynyt koska kohde ei ollut tarpeeksi selkeä

Kysymys 10: Jos olisit turistina vieraassa kaupungissa ja Mobile Guide -palvelu olisi saatavana, mitä parannuksia / muutoksia kaipaisit, jotta jättäisit perinteisen paperisen kaupunkioppaan hotellille?

Katuosoitteet olisi avuksi, Kuvia kohteista tekstin yhteydessä!

Kysymys 11: Mitä mieltä olet sovelluksen sisällöstä ja luettavuudesta? Kuvaile lukukokemustasi.

Tekstien löytäminen ei onnistunut, sisältö oli muuten hyvä ja näytti hyvältä Raikkaita värejä.

Kysymys 12: Mistä olisit halunnut lisää tietoa? Mitä tietoa oli liikaa?

Ei saanut käsitystä siitä mitä esim. Lintuparatiisi oli, vaikka tekstiä oli paljon.

Kysymys 13: Oliko tekstien rakenne sekä sisältö helposti seurattava ja ymmärrettävä? Millaisia muutoksia olisit toivonut tekstien rakenteeseen?

Ihan hyvä teksti joka oli helposti ymmärettävä.

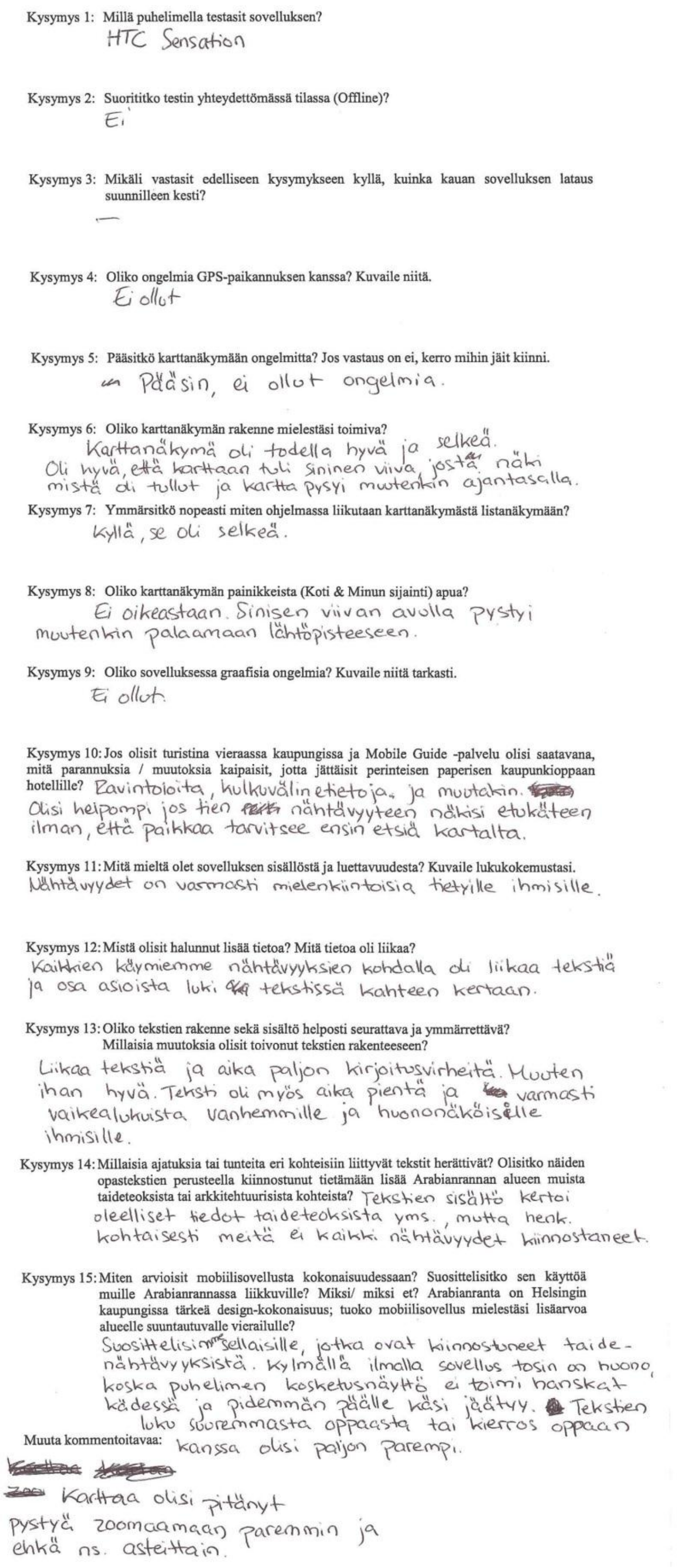
Kysymys 14: Millaisia ajatuksia tai tunteita eri kohteisiin liittyvät tekstit herättivät? Olisitko näiden opastekstien perusteella kiinnostunut tietämään lisää Arabianrannan alueen muista taideteoksista tai arkkitehtuurisista kohteista?

Ihan tarpeeksi tietoa, liikaa niin ei [illegible] kiinnosta

Kysymys 15: Miten arvioisit mobiilisovellusta kokonaisuudessaan? Suosittelisitko sen käyttöä muille Arabianrannassa liikkuville? Miksi/ miksi et? Arabianranta on Helsingin kaupungissa tärkeä design-kokonaisuus; tuoko mobiilisovellus mielestäsi lisäarvoa alueelle suuntautuvalle vierailulle?

Ihan hieno... hieman vaikea käyttää eikä sovellus toiminut toivotulla tavalla. Suosittelisin jollekulle joka ymmärtää tekniikan päälle.

- APPENDICE: Test feedback 5



Kysymys 1: Millä puhelimella testasit sovelluksen?

HTC Sensation

Kysymys 2: Suorititko testin yhteydettömässä tilassa (Offline)?

Ei

Kysymys 3: Mikäli vastasit edelliseen kysymykseen kyllä, kuinka kauan sovelluksen lataus suunnilleen kesti?

—

Kysymys 4: Oliko ongelmia GPS-paikannuksen kanssa? Kuvaile niitä.

Ei ollut

Kysymys 5: Pääsitkö karttanäkymään ongelmitta? Jos vastaus on ei, kerro mihin jäit kiinni.

Pääsin, ei ollut ongelmia.

Kysymys 6: Oliko karttanäkymän rakenne mielestäsi toimiva?

Karttanäkymä oli todella hyvä ja selkeä.
Oli hyvä, että karttaan tuli sininen viiva, josta näki
mistä oli tullut ja kartta pysyi muutenkin ajantasalla.

Kysymys 7: Ymmärsitkö nopeasti miten ohjelmassa liikutaan karttanäkymästä listanäkymään?

Kyllä, se oli selkeä.

Kysymys 8: Oliko karttanäkymän painikkeista (Koti & Minun sijainti) apua?

Ei oikeastaan. Sinisen viivan avulla pystyi
muutenkin palaamaan lähtöpisteeseen.

Kysymys 9: Oliko sovelluksessa graafisia ongelmia? Kuvaile niitä tarkasti.

Ei ollut.

Kysymys 10: Jos olisit turistina vieraassa kaupungissa ja Mobile Guide -palvelu olisi saatavana, mitä parannuksia / muutoksia kaipaisit, jotta jättäisit perinteisen paperisen kaupunkioppaan hotellille? Ravintoloita, kulkuvälinetietoja, ja muutakin.
Olisi helpompi jos tien nähtävyyteen näkisi etukäteen
ilman, että paikkaa tarvitsee ensin etsiä kartalta.

Kysymys 11: Mitä mieltä olet sovelluksen sisällöstä ja luettavuudesta? Kuvaile lukukokemustasi.

Nähtävyydet on varmasti mielenkiintoisia tietyille ihmisille.

Kysymys 12: Mistä olisit halunnut lisää tietoa? Mitä tietoa oli liikaa?

Kaikkien käymiemme nähtävyyksien kohdalla oli liikaa tekstiä
ja osa asioista luki tekstissä kahteen kertaan.

Kysymys 13: Oliko tekstien rakenne sekä sisältö helposti seurattava ja ymmärrettävä?
Millaisia muutoksia olisit toivonut tekstien rakenteeseen?

Liikaa tekstiä ja aika paljon kirjoitusvirheitä. Muuten
ihan hyvä. Teksti oli myös aika pientä ja varmasti
vaikealukuista vanhemmille ja huononäköisille
ihmisille.

Kysymys 14: Millaisia ajatuksia tai tunteita eri kohteisiin liittyvät tekstit herättivät? Olisitko näiden opastekstien perusteella kiinnostunut tietämään lisää Arabianrannan alueen muista taideteoksista tai arkkitehtuurisista kohteista? Tekstien sisältö kertoi
oleelliset tiedot taideteoksista yms., mutta henk.
kohtaisesti meitä ei kaikki nähtävyydet kiinnostaneet.

Kysymys 15: Miten arvioisit mobiilisovellusta kokonaisuudessaan? Suosittelisitko sen käyttöä muille Arabianrannassa liikkuville? Miksi/ miksi et? Arabianranta on Helsingin kaupungissa tärkeä design-kokonaisuus; tuoko mobiilisovellus mielestäsi lisäarvoa alueelle suuntautuvalle vierailulle?

Suosittelisimme sellaisille, jotka ovat kiinnostuneet taide-
nähtävyyksistä. Kylmällä ilmalla sovellus tosin on huono,
koska puhelimen kosketusnäyttö ei toimi hanskat
kädessä ja pidemmän päälle käsi jäätyy. Tekstien
luku suuremmasta oppaasta tai kierros oppaan
kanssa olisi paljon parempi.

Muuta kommentoitavaa:

~~Zoo~~ Karttaa olisi pitänyt
pystyä zoomaamaan paremmin ja
ehkä ns. asteittain.