A Mobile Guide for geographically displaying estates listed for sale.

Rune Romundstad

Master of Science in Computer Science
Submission date: June 2007
Supervisor: John Krogstie, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

The project will develop a prototype for displaying estates listed for sale in a map on mobile devices. The state-of-the-art within location-oriented services, positioning and map techniques will be surveyed, and the best suited maps will be deployed in the prototype.

The goal of this project is to see if such an application is useful to potential users. The result will be concluded through an evaluation which among other factors involves feedback from an estate agency.

Assignment given: 18. January 2007
Supervisor: John Krogstie, IDI

Abstract

Instead of displaying a simple list with the houses and apartments which are listed for sale, an alternative is to display them geographically in a map to get an overview of where they are located. To make it simple the map also has an icon of where you are located, so you know where you are according to the houses.

A prototype of such a system with the most important functionality has been implemented. The system gives the user the option to choose different other facilities he or she wants to show in the map together with the apartments, such as schools and parking spaces etc. The system gives also short information of the different apartments when clicking on them, and the possibility to open the prospect of that specific apartment.

The prototype has been evaluated by some “experts” from one of the largest estate agencies, together with a user- and a technical evaluation. The feedback from the “experts” has been very positive, and they are convinced that such a system will be very helpful for potential users. The response regarding the user-friendliness has also been good, except from some comments on too many confirmations needed when running the application. During the technical evaluation we got confirmed that a J2ME application not is directly portable between different devices without slightly adjustments.

Preface

This report presents the final thesis for the Master of Science in Technology at Norwegian University of Science and Technology, the spring 2007.

We would like to thank supervisor John Krogstie for his support and help during the process. We would also like to thank our fellow students for support during the work, together with Camilla Tepfers and Gro Frydenlund at DnBNOR for taking the time to give us valuable feedback during the evaluation.

Trondheim 19th of June 2007

____

Rune Romundstad

Contents

List of Figures

List of Tables

Chapter 1: Introduction

This first chapter will introduce the problem and explain our motivation and goals. A scenario for the use of the solution will be presented, and in the end we will shortly introduce the chapters following in the report.

1.1 Motivation and goals

Wireless technologies are becoming more and more popular around the world. Consumers appreciate the “wireless lifestyle”. This is a lifestyle that exists with the purpose of making the daily tasks easier to carry out. We are surrounded by all sorts of technologies and carry around mobile devices which can be used to perform more and more of our needs.

The background for this report is to make use of wireless network technologies in the city center of Trondheim or other cities, together with estate agencies and the process of selling apartments. In this context it is interesting to look into location oriented services and positioning technologies to see what type of services can be useful in this domain.

Location-oriented services have a huge potential because it can give the user the specific information he or she wants at that precise location and eliminate other irrelevant information in that specific setting.

After having read the customer paper [1] from a major estate agency, where they predict that in near future mobile phones will be the most important channel of communication between estate agent, buyer and seller, we are convinced we have chosen a domain with huge potential.

With this background we have picked one potential area and have looked into geographically displaying houses and apartments listed for sale.

This type of project when developing technology-based solutions to relevant business problems is in the direction of what is called “design-science research”. We have therefore tried to relate our process to the guidelines described in [33]. Not all of them are applicable to our process, but some of them; problem relevance and design evaluation we have included in the chapters of this report.

The main goal for this project is to confirm our hypothesis, that such a service will be useful to the people searching for houses or apartments. We will examine this by developing a prototype and performing an evaluation where we among other things contact a major company in the estate business to get useful feedback.

1.2 Scenario for location oriented services

An average family is out in the estate market, looking for a new apartment or house. The family consists of two parents and two children. The children are respectively two and six years old.

When they are looking at a specific apartment, they might be interested in getting an overview of other apartments nearby meeting their requirements.

It may also be useful to get an indication of where the nearest school and/or kindergarten are located, since they have two children. Information regarding other types of facilities nearby may also be of interest, for example if there are any parking areas in the neighborhood.

In this type of scenario it will be important that the information of interest is easily accessible.

1.3 Problem text

"A mobile guide for geographically displaying estates listed for sale"

1.4 The different parts of the project rapport

This subsection will give an overview of the different chapters following in this project report, and a short introduction to what each of them covers.

1.4.1 Theory and background

The chapter represents the state-of-the-art technologies we have surveyed to find a best possible solution to the problem.

1.4.2 Estate services

The chapter gives an introduction to the different estate services on the Internet and mobile devices. It also describes an existing system and how our solution is thought as a part of it. This chapter is relevant to the problem relevance guideline of the design science research.

1.4.3 Design and implementation of solution

This chapter will present the design of our prototype and state the reasons for the choices made. The functionality will be presented on a high perspective, because it is no need to go thoroughly into the technical details. We refer to the appendix for full Java documentation. This chapter is also related to the problem relevance guideline where we describe the developed solution.

1.4.4 Evaluation of prototype

This part will contain a technical and a user-related evaluation, together with an expert evaluation of high significance. The results from the different evaluations will be documented in this chapter. This chapter represents the guideline for design evaluation of the design science research.

1.4.5 Conclusion

The report finishes with a conclusion that sums up the work. It also have a subsection called future work, where we give ideas for improvements based on feedback and own experiences during the project.

Chapter 2:

Theory and background

This chapter gives a brief introduction of the main topics and the more specific technology used in this project.

2.1 Location oriented services

Within a wireless network there can be developed several services offered to potential users. Some can be more related to the user's position inside a network than others. A user at a specific position can be interested in other types of information, than users at other locations.

This is the intention of location oriented services [2]. When a user is at one location he or she can get detailed information about that specific place, not any irrelevant information about other places.

At the Carnegie Mellon University in Pittsburgh, USA, they have developed a semantic web environment for context-aware mobile services, called myCampus [3]. This system offers location oriented services to the users. When you are walking around on campus you can get an outline on your mobile device of where the nearest canteens are located, based on your current position.

To provide a benefit to the users, the service requires information about the current position of the user. Possible methods for positioning a user will be covered in a later section of this chapter, Positioning.

One of the largest companies in Norway within passenger transport, Oslo Taxi, has invested in a traffic routing system to increase the efficiency of taxi ordering. By calling the number to the nearest taxi rank, you will automatically be transferred to the nearest available taxi driver. This system uses GPS communication to keep track of the location of the cars. The system increases the driver's security in case of an accident or assault and also reduces unnecessary idling. [4] [5]

A small example of a location oriented service in Trondheim is the city journey project. [6] When a person walking around in Trondheim, stops for example at Trondheim square, and wants to see how that place looked like for many years ago it is of no interest getting pictures of all other places around and need to scroll and look for the right one. Just the right pictures of the actual location should be provided.

Location oriented services can be divided in to three types according to [7]:

- Pull-based applications. User must actively send out information to the server about his/hers position to receive information about the location and neighborhood.
- Push-based applications. Service providers detect a user in the proximity and sends out (push) services such as advertisements. In these types of application it is a central issue of how to protect the user's privacy. Location information can be sensitive, so it is important vendors get and push information, from and to only optional users.
- Combination of the two types above. Users can, for example get information about location for a cinema, and make a reservation on their mobile devices. Then he or she can send an alert from the cinema to others within a certain distance and the other will be able to accept or reject the invitation.

Location oriented services offers appealing business values, but it's not always easy to implement. Any location oriented service is based on a Geographic Information System (GIS) to find the users location and offer the specific service.

The optimal of any location oriented service is to run on a GIS server [8], because those servers are dedicated to work with maps. They are capable of map rendering and have built in complex geographic algorithms.

There are both similarities and differences between location oriented service technology and standard GISs [8]. Much of the underlying functionality, for example mapping and geocoding that is used to deliver location oriented services, originate from GIS. What differences the location oriented systems is that it is deployed on the foundation of IT and wireless technology. GIS is generally limited to desktop or client-server solutions, whereas location oriented systems includes many players, for example hardware and software vendors, infrastructure providers and mobile device vendors.

The Norwegian yellow pages have implemented a WAP solution where the user easy can search for persons, shops or services nearby his/hers current location with the use of a mobile phone. The position is calculated by the telephone companies with GSM positioning out of the distance to the mobile antennas. [9] Also see the section for positioning later in this chapter for more information on GSM postioning.

One of the critical topics regarding location oriented services is the privacy of the user. [2] It is important to not flood the user with information without his or hers approval.

Some of the key requirements a location oriented service should meet are high performance, reliability, scalability and interoperability with other systems. [8]

In the rest of this chapter we will present topics relevant to be able to offer location oriented services. Positioning, wireless networks and map services together with more implementation specific technology.

2.2 Positioning

Positioning a user is one of the most important tasks in a location oriented service. Without reference to where the user is located, it is impossible to offer useful services.

Positioning a user within a wireless network may be a very complex task, with heavy calculations regarding the distance to the nearest access points. Fortunately there exist solutions for such calculations. Later in this section the system used in wireless Trondheim, Cisco wireless location appliance will be explained together with other methods, GSM- or GPS-positioning.

When operating with map references and coordinates, it is mainly two different types. The format used in the standard GPS systems, WGS-84 (World Geodetic System) [10] gives the location in degrees, longitude and latitude. While the other type, Euref89 (European Reference Frame 1989) [11] also gives the position in longitude and latitude, but in meters. For instance the longitude is expressed with the number of meters north of the equator. The coordinates in WGS-84 deviate less than half a meter from the coordinates in Euref89. [10]

In this project the Euref89, or UTM[EUREF89] (Universal Transversal Merkator) which is the correct designation, has been used because it is the easiest to work with when the objective is to calculate positions in a map.

2.2.1 GPS

GPS (Global Positioning System) [12] is a conception used for satellite-based navigation system, or NAVSTAR which is the original name for the network of at least 24 satellites circulating the earth. The satellites were placed in the orbit by the U.S. Department of Defense [13] and were originally intended for military purposes. GPS became available for civilian use in the 1980s, and is a system free to use. The satellites are powered by solar energy with backup batteries that keep them running when the earth shadow for the sunrays.

2.2.1.1 The GPS location process

The GPS-satellites [13] circles the earth two times within 24 hours and sends out radio signals to the earth. We have GPS receivers getting this information and uses 3-point crosschecking to calculate a position. This gives a two-dimensional position with longitude and latitude. The GPS receiver must be locked on at least 3 different satellites to get a two-dimensional position, with four or more it can also calculate the users' altitude to give a three-dimensional position.

For only a couple of years ago, GPS receivers were mostly a tool mainly used by people on expeditions or for navigating through unknown territories such as mountain plateaus.

Today more and more mobile devices come with a built in GPS or with support for an external through a cable or Bluetooth connection. Many newer cars also have built-in navigation systems, or you can buy one yourself and put it in your car.

With the continuously growing market and development of devices with GPS, it opens for constantly new types of services.

2.2.2 GSM

There are some different ways in which a position can be derived from measurement of signals applied to the GSM system. [34] Each of them defines a locus where the positioned mobile phone must lie. The position of the mobile phone is defined by the intersection of the geometric position from multiple measurements. If too few measurements are available the geometric positions will intersect at more than one point and resulting in an ambiguous position estimate.

The most important measurements to calculate a position are:

- Propagation time. Based on measuring the time a signal use to travel between a mobile phone and a base station.
- Time difference or arrival (TDOA). A mobile phone “listens” to different base stations and measures the time difference between each arrival. Each TDOA measurement defines a hyperbolic geometric location in which the mobile phone must lie and an intersection of the different hyperbolic locations defines the position.
- Angle of arrival. This method measures the angle of the arrived signal from a base station or vice versa. A single measurement gives a straight line geometric location from a base station to the mobile phone. Another measurement yields a second straight line and the intersection of the two lines give a position.
- Carrier phase. A large number of ambiguities arise in the positioning solution. It can measure the phase of the received signal but it cannot directly measure the number of wavelengths between the transmitter and receiver. It also needs to maintain a lock on the carrier signal. Failure to do so will result in cycle slips and error in position. This method works better for GPS than GSM.

It is also possible to mix different measurement types. In [34] the authors give an example with radars, which combines the angel of arrival method with propagation time measurement. By mixing different measurements you can get advantages, for example with a circular-angle system you can get a position out of just one base station.

2.2.3 Cisco wireless location appliance

The Cisco wireless location appliance [14] provides a solution to simultaneously track the devices within a wireless network. It uses advanced RF (RadionFrequency) fingerprinting technology to track the 802.11 compatible devices. The appliance also provides location-based information and records the information to be used for location trending.

The location appliance uses Cisco wireless controller and access points to track the physical location of the wireless devices down to just a few meters. Once network maps and access points are added to the appliance, it can generate heatmaps to display the location of devices graphically (see figure 1 below). The location information is also available to third-party applications through SOAP/XML API.

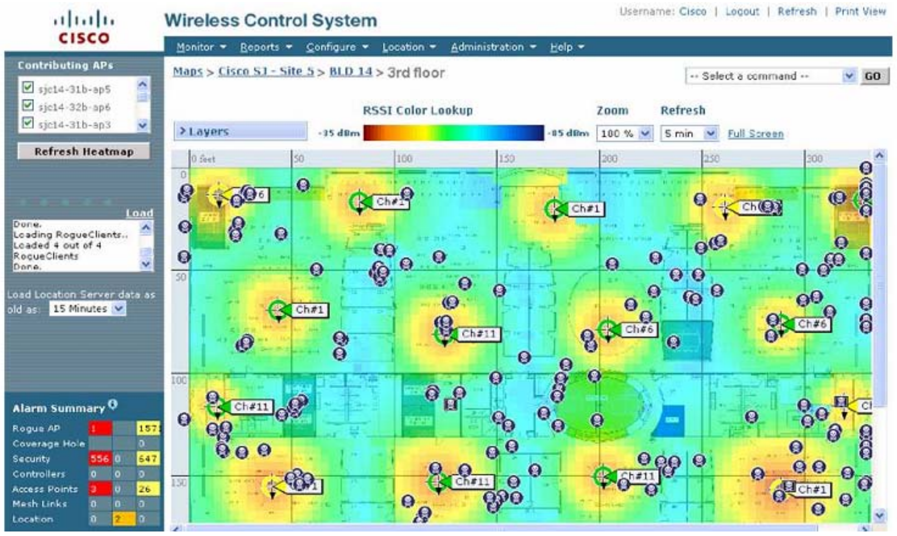


Figure 1: Example of targeted devices within a wireless network [14]

The administrators of the system can also get detailed information about each of the devices/users in the area for containment of wireless intruders or troubleshooting. For example IP address, username, MAC address together with historical physical location and point in time information of where users have been.

Alongside tracking 802.11 clients, it also offers the possibility to track non-Wi-Fi mobile devices equipped with active RFID (Radio Frequency Identification) tags.

2.3 Wireless Technologies

Wireless technologies are constantly increasing in number and size.

Most of the wireless networks today are based on the 802.11 specifications. [15] A person with a WI-FI enabled system such as a computer or mobile device can connect to these networks when he/she is in proximity of an access point. The numbers of devices which support WI-FI are also increasing rapidly.

Many stores or other public services have wireless networks for people to connect and go online or sending mail etc. And a constant increasing number of people also have wireless networks in their homes.

A newer technology, who in the future can replace the standard 802.11 networks, is WiMAX (World Interoperability for Microwave Access). Because it has a better range, it can offer services to a much wider area.

WiMAX [16] is based on the IEEE 802.16 standard and offers true broadband speed over wireless network. There are two main types of WiMAX today, fixed and mobile. Fixed is point-to-multipoint enabling broadband access to homes and business, while mobile WiMAX offers full mobility of cellular networks at true broadband speed.

Mobile WiMAX is based on the OFDMA (Orthogonal Frequency Division Multiple Access) technology which inherent advantages to provide higher performance than today’s wide area wireless technologies.

A big disadvantage with WiMAX is that there is no possibility to locate a device in the network. With just one access point covering a large area you don’t get the reference points you need to locate a user.

2.4 Wireless Trondheim

Wireless Trondheim [17], [18], [19] started as a research project at NTNU (The Norwegian University of Science and Technology), in 2005. One of the main goals is to have a laboratory for research and development within wireless technologies and services. This is a unique way to test out new types of services to provide to people, the potential users can test it themselves all over the city as long as they are in a covered area.

Examples of services which can be provided are: a tourist guide, city walk guide (see picture of what the location looked like for many years ago) [6], different types of city or shopping guide. This together with the more standard services, where people can get online wherever they are in the city, will expand the services offered to the people in Trondheim.

Today (by 2nd of May 2007) mostly of the city center in Trondheim together with a part of the road up to NTNU is covered by the wireless network.

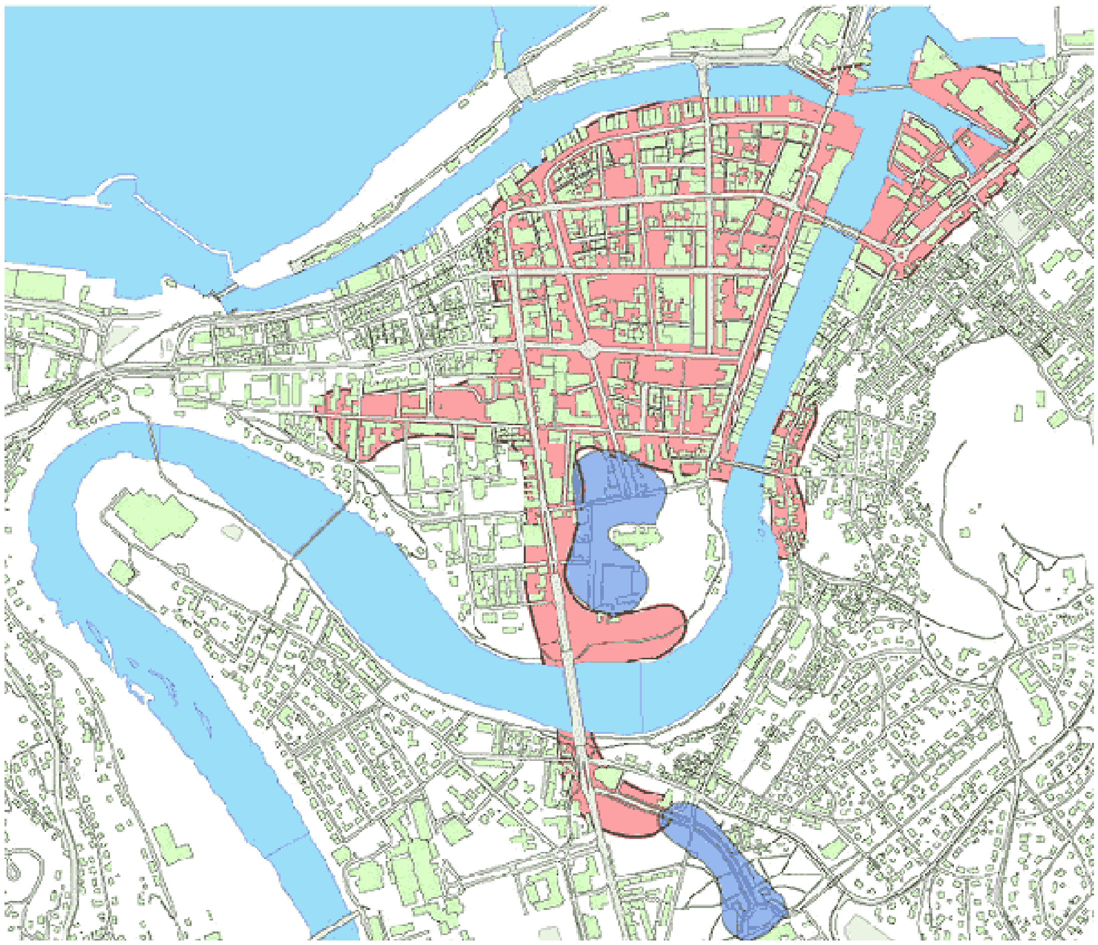


Figure 2: Coverage map of wireless Trondheim. [20]

Above (Figure 2) is a map of the city center with the coverage marked with red. The blue areas are the first step of the planned expansion.

2.4.1 Wireless network technologies used in wireless Trondheim

The network used in wireless Trondheim is based on the WI-FI standard, 802.11g, with a lot of access points[1] placed strategically around the city to provide a widest possible coverage.

There exists plans for wireless Trondheim in the future to be based on the standard WIMAX and the network will then be able to cover a larger area of the city, but nothing is yet decided. The benefit of changing to this type of technology is the broader

[1] By 1st of March 2007, Wireless Trondheim consists of 118 access points.

coverage, but there will be more difficult to get a geographical position of the users, and therefore more difficult to provide location oriented services to the people.

2.5 Map services

There exist a number of map services on the web and we will in this section look in to some of them. See appendix C for an example-map of Trondheim city centre from the three following map services for an indication on the detail-level. At the end of this section we will in addition present a map server technology.

2.5.1 Google Maps

Google Maps [21] is a free map service from Google. It also has an open API [22] so you can include a map in your own webpage. It is for free use as long as you don't get paid for the service you provide with it. All you need is to sign up for an API key, and include this in the source code.

Google Maps can also be used for travel planning, where you get a detailed description of how you get from one location to another.

2.5.2 Yahoo! Maps

Yahoo! Maps [23] is the map service from Yahoo! network, which like Google also offer an API to the developers. [24] Compared to the Google maps, the Yahoo! Maps have less detailed maps with fewer street names, on a nearly equal zoom level.

2.5.3 Microsoft Virtual World

Microsoft has a map service called Microsoft Virtual World [25], which is the same map they also use in their live search service [26]. It is a high detailed map, which also provides a SDK for the developers.

2.5.4 Map server

There exists some environment for developing own map servers. One is MapServer [27] which is an open source environment. It renders maps, images and vector data for the Internet. MapServer was developed by the University of Minnesota in cooperation with the Department of Natural Resources and NASA, and is still hosted by a NASA sponsored project.

MapServer includes both support for cross-platform and for the most popular scripting and development environments (PHP, Perl, Java and C#).

One of the major challenges when working with a map server is to find good and public available map data.

2.6 Development language

This section will introduce the most common language for developing applications for mobile devices, J2ME. [28]

2.6.1 J2ME

J2ME (Java 2 Micro Edition) is the java development language for mobile devices. It provides a flexible environment for running applications on different embedded devices, such as mobile phones, PDAs or printers. J2ME is the most flexible development platform for embedded devices, because most of the devices developed today, have support for Java. Most developers use Java as their language when developing for embedded devices and it is also the most popular platform for creating games for mobile phones, mainly because the games easily can be emulated on a PC during development, before uploaded to a phone.

J2ME devices implement a profile and the most common are MIDP (Mobile Information Device Profile) [29] or Personal Profile. MIDP is directed at mobile phones, while personal profile is directed at embedded devices such as PDAs and devices connected to a television.

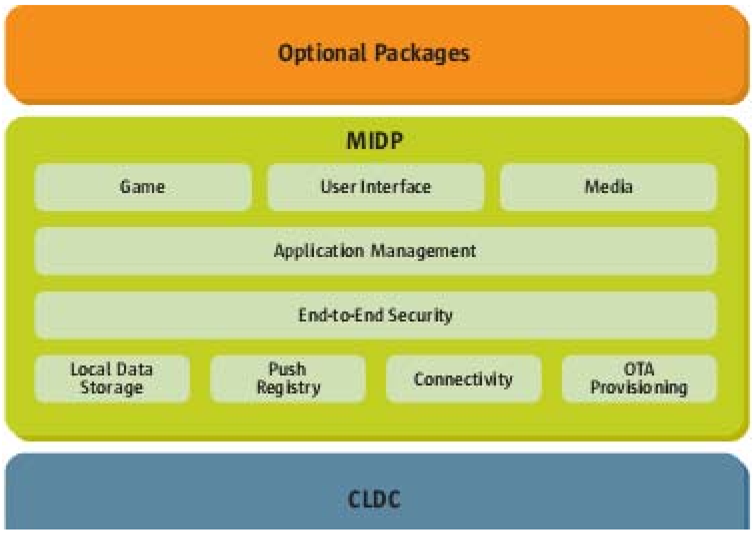


Figure 3: MIDP subsystems and services. [29]

The profiles lay on top of a configuration. Either CLDC (Connected Limited Device Configuration) or CDC (Connected Device Configuration).

2.6.1.1 Connected Limited Device Configuration

CLDC contains only a reduced subset of the Java class library and defines a base set of programming interfaces for resource-constrained mobile devices. Together with a profile, such as MIDP, it provides a solid platform for developing applications to run on devices with limited memory and process power.

Mobile Information Device Profile

MIDP is designed for mobile phones and almost all new phones come with an implementation of MIDP. The applications written for this profile are called MIDlets. There exist a number of applications on several platforms for writing MIDlets, for example Sun Java Wireless Toolkit.

MIDlet application together with the CLDC is the best suited for this project, since the application will be developed on a mobile phone.

The MIDlet applications are built up on three ground states. [37]

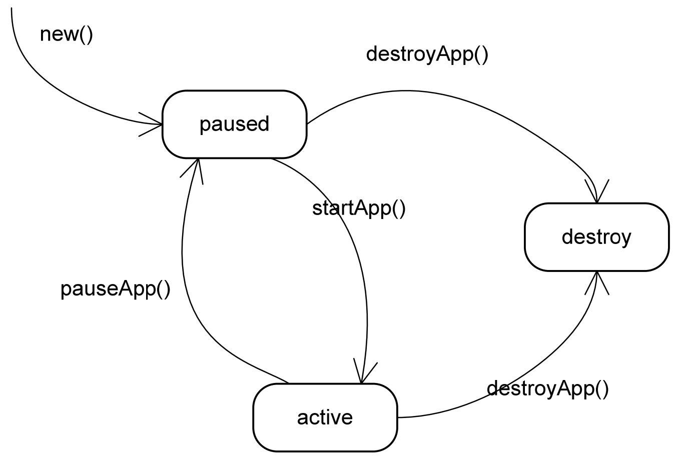


Figure 4: The different states of a MIDlet application. [37]

The 3rd generation of the MIDP, version 3.0, is under development. It will include new functionality all over and improve the interoperability across devices.

2.6.1.2 Connected Device Configuration

CDC is almost a full subset of the Java SE library, except of the classes which are GUI related. The CDC provides a much richer library than the CLDC, and is mostly used to develop applications for embedded devices.

Chapter 3: Estate services

Everybody will at least once in a lifetime be involved with an estate agency in the process of buying or selling houses or apartments. All estate agencies offer different types of services to the customers through distinct communication channels.

This chapter gives an introduction to the relevance of the project. We explain where our solution fits into the already available services.

The most common medium for advertising estates are Internet or newspapers. Every weekend one of the major newspapers in the Oslo region has a giant addition with nearly one hundred pages with listings of homes for sale.

Today the estate agencies also needs to think “new” to more efficiently reach out to the people [1], therefore continuously more service are implemented to also work on mobile devices, but then with slightly reduced functionality. This because the mobile devices have limited support for many of the technologies used when the services are offered through Internet. But the development in the mobile industry is growing rapidly, and we don’t think it will take long until mobile devices have full support for the most used technologies.

3.1 Internet services

There exists different estate services on the Internet today and every estate agency has their own web pages where they have listings of houses for sale. One of the most known and visited web pages for advertisement of estates in Norway is Finn.no, with more than 2, 1 million unique users every month. [35]

A new type of estate service is Igglo [36], which is developed by Igglo Media As. This service is temporary in a fully functional beta version and is only available for Oslo. The service has a map where all buildings with apartments are marked with a house-icon.

When you put your mouse pointer over the icons, an image of the building will show in a small pop-up window.

The new functionality which we haven't seen anywhere else is the possibility to earmark apartments anywhere in the city which you are interested in. This enables the owner of an apartment to see if any are interested in his/hers apartment before they consider to sell. This brings the estate services a step further so the buyers can show their interest also in apartments not for sale, and not just those already listed for sale. The houses which are earmarked have a red icon, the others are black. The map services offered by Igglo are based on Google maps. [21]

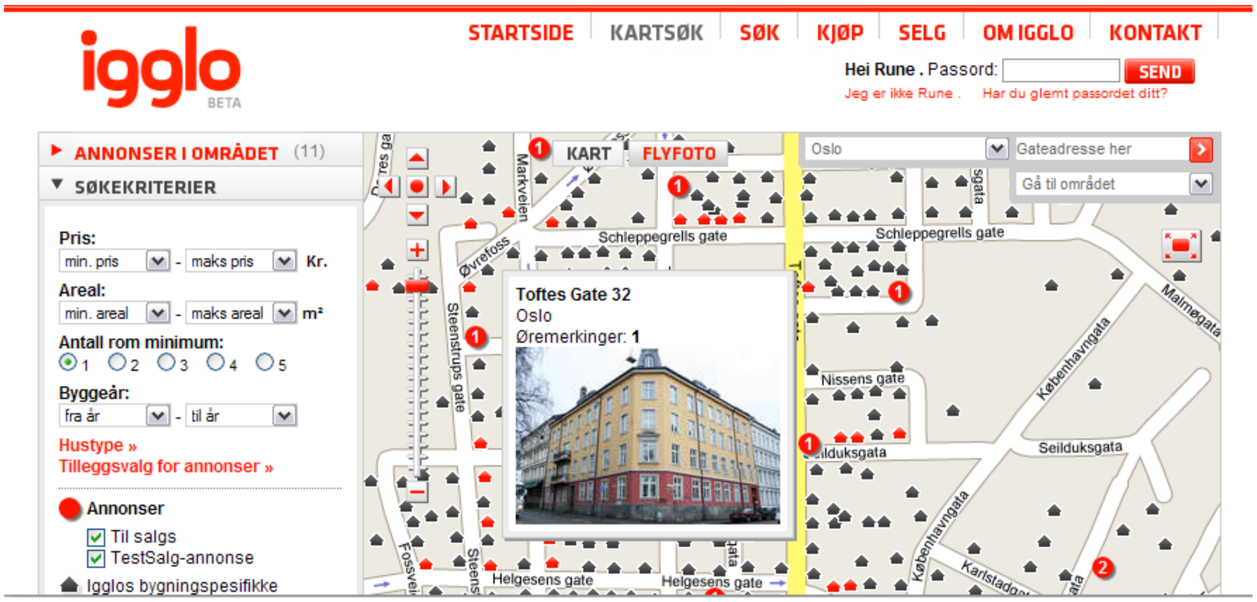


Figure 5: Example of Igglo's house search. [36]

We will further in this chapter introduce some of the existing mobile estate service available and describe shortly how they work and what they offer.

3.2 Mobile estate services

Some of the agencies in the estate market offer services for mobile devices. We will in this section take a closer look at two of them. First we will introduce Finn.no's mobile service and then we will describe the existing mobile portal offered by DnBNOR. In the end of this chapter we will describe how our solution will fit into the existing solution DnBNOR offer.

The mobile portal from Finn.no is a simple application to download and install. By sending a text-message with the word “finn mobil” to a certain number, you get a link to download from. Then it is just to open the application every time you want to use the service. The application has a simple interface and is very user-friendly.

The portal offered by DnBNOR is more an Internet portal where you get a link to a Internet location. This offers key services to the users such as searching for houses or vacation homes sorted by province. They also offer some information about housing loan and you can get in contact with an estate agent. But in our context it is the services for searching for houses which is important, so we have just looked into this function.

When searching for houses in the portal you will first be enquired of which province you are interested in, first region and then municipality. When selecting municipality you also choose the wanted prize interval, one or several choices can be ticked off. Then you will get the different houses within the choices made, listed.

Neither of these systems includes any form of map service or information of other facilities nearby. Neither has the existing system functionality to get track of where the user are positioned, the user must self know where he is, and finding interesting apartments in the neighborhood can therefore take too much time.

We think a map service with a positioning system would be very useful in such a context.

3.3 Our Solution

Our solution is thought as part of the already existing mobile portal from DnBNOR, described above. As an addition to the already existing functions, we find it useful for the users to also have a map service which gives a reference of where the different apartments for sale are compared to where the user is.

Our solution will fit into the already existing portal, as described above, and will be available after the users have chosen the type of house and price interval. This will be an alternative to just getting the different houses listed up.

The solution will give the users a map on his/hers mobile device to see which apartments nearby are listed for sale and open for he or she to take a look. This will be very helpful so you don’t need to remember all potential apartments and their addresses. It also offers the possibility to click on the apartments and get short information with key facts, together with a command for opening the prospect of the given apartment.

Our solution also gives the possibility to display different facilities, such as parking spaces, schools or kindergarten located nearby, in the map. Also other facilities could be implemented, but these were chosen just to give an example of how it would work.

We will develop a prototype with the thought solution. The prototype will only have the main functionality as described above - this is enough to look into what we want, and get response of the potential of such a service.

Chapter 4:

Design and Implementation

This chapter describes the design and the implementation of the client and server. First we will describe the whole system with architecture, limitations, main functionality and how the communication between client and server happens.

Next we will describe the design and implementation of the client before we in the end of this chapter write about the server.

4.1 System architecture

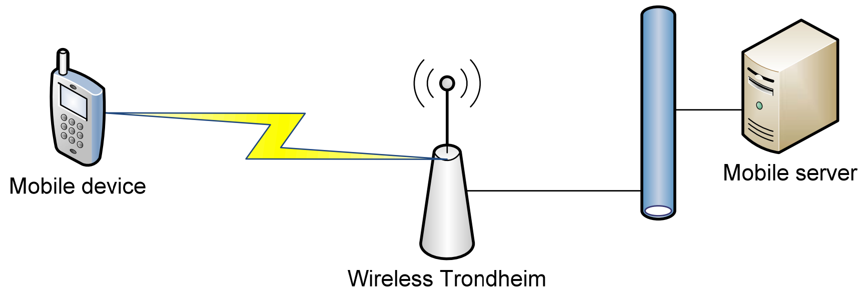


Figure 6: The architecture of the whole system

The project focuses on the system involving a client and a server as shown above in figure 6, communicating through a wireless network.

4.2 Limitations

We will not implement functionality for a map server since development of such a server we find out of scope for this project. Instead the whole map, cut into pieces will be stored locally on the mobile device.

Because there do not exist any well functional and easy-to-use maps within the wireless Trondheim environment, the maps used have been ripped from the Google Maps service. After having looked into different map-services, we chose to use the map from Google Maps for our application. This because it was the map we found best and easiest-to-follow, with the most detailed information regarding street names.

Some other Internet based map services also exist, such as Yahoo and Microsoft, but Google was the service we had used the most before, and is also the most widespread of them. Google Maps offers also an API for developers, for example to include a type of map service to their website. For web application it would work great to use this API and we have tested it during this project to see how it works and if it was possible to use it on the server in our application. But since we are implementing a service for mobile phones, it would face some difficulties showing the maps on the mobile devices because map services are using AJAX (Asynchronous JavaScript and XML) something which is not commonly supported on mobile devices.

This prototype will not be connected to any positioning server or GPS to get the current position of the user. The used position will be set by the server.

During the development process we had a mobile device available for continuously testing. This device had a touch-screen, so the application will be limited to work a hundred percent on suchlike devices.

4.3 Main functionality

The main functionality of the system is to provide the user with relevant map information. The maps presented to the user will have his/hers position marked in it together with the location of the nearest apartments and other user-specified facilities.

At startup the user will be asked to choose which types of facilities he or she wants to be displayed in the map. The list to choose from consists of the choices: Parking, Schools and Kindergartens. Based on the choices in this menu, the client will display the chosen facilities in the map together with the user's position and the apartments.

In the map it is possible to click on one of the apartments with a pen and key-info about the specific apartment shows below the map.

The user will also have the possibility to open the prospect for an apartment of interest in the Internet browser on the mobile device. The link to the apartment will be set when the user clicks on one of the apartments on the screen.

4.4 Communication

The client sends a request to the server to tell it wants the location data. In reply it gets a string from the server with a position and a reference to the right piece of the map.

The system uses the http protocol to communicate between client and server. The http protocol was chosen because it is the easiest form of communication between servers and mobile devices, regarding data transfer.

4.5 Test data

The data used to test this application is manually generated. All positions have not been check to be actual apartments, schools or parking areas. A future possibility for this type of system is to have a link to an estate agency's database, for automatically retrieval of relevant information.

Each position shown in the map is represented with "X" and "Y" coordinates which refer to respectively latitude and longitude in the Euref89 format.

4.6 Design and implementation of client and server

This part introduces the design and gives account for the implementation of the system. First we present a sequence diagram for the whole system, and then we will go though the client and server in more detail.

4.6.1 Sequence diagram for the whole system

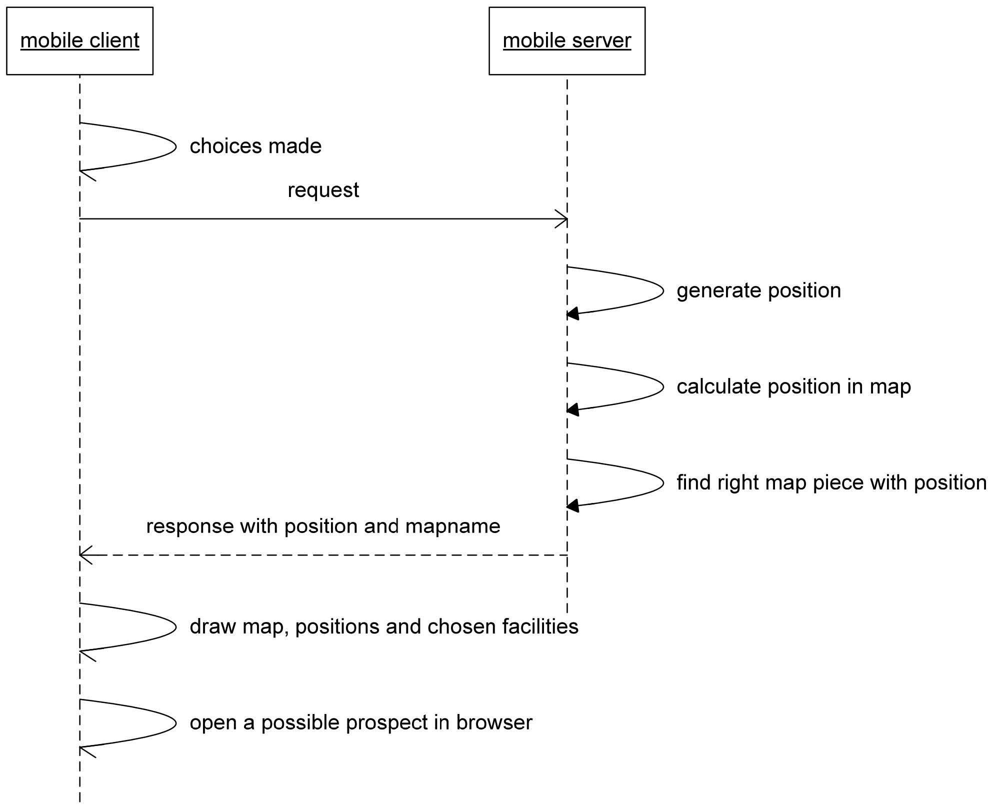

mobile client
mobile server
choices made
request
generate position
calculate position in map
find right map piece with position
response with position and mapname
draw map, positions and chosen facilities
open a possible prospect in browser

Figure 7: System-level sequence diagram.

The sequence diagram above (Figure 7) shows how the system works when a user starts the client on his/hers mobile device.

4.6.2 Design of client

The client will be developed in Java with the Java 2 Micro Edition (J2ME) API. J2ME is the Java API for developing applications for devices with limited memory resources and processor power, for example smart phones. Java was chosen rather than for example the .NET Compact Framework, because of its platform independency. There are many different vendors on the market, but mostly of the mobile devices have Java implemented.

4.6.2.1 Components

In this subsection we give a short introduction of the components of the client.

Figure 8 gives a view of the classes in the client application. The classes and functions will be presented briefly in this part, but we will refer to Appendix D for the full Java documentation.

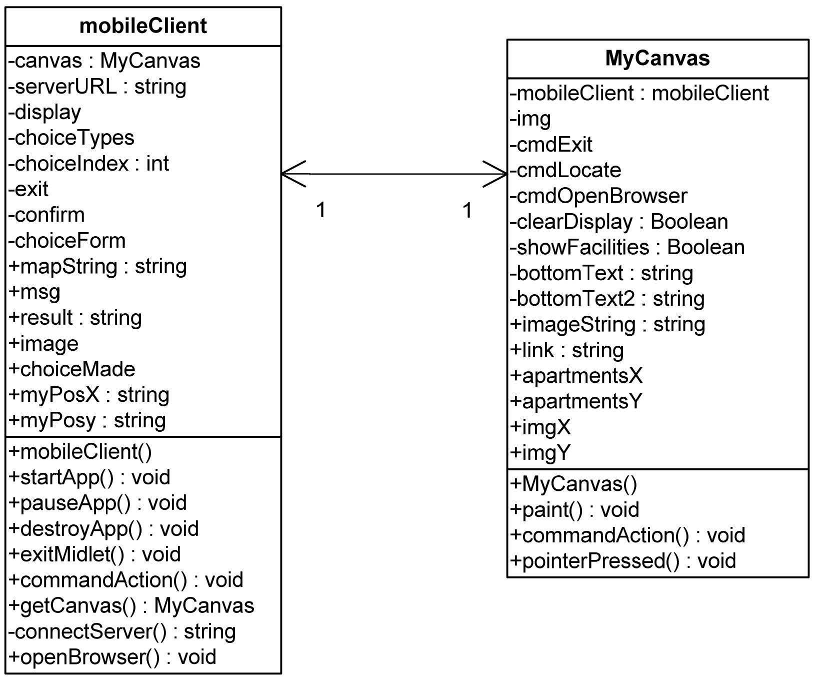


Figure 8: Class diagram of client.

The client application consists of two classes. The `mobileClient` is the main class, which makes the program running and includes the requested functions for a mobile application. And the `MyCanvas` class, which takes care of the graphical elements, such as drawing the image, displaying the shapes in the image and listening to user-made events such as when the pointer is pressed on the screen.

4.6.3 Implementation of client

An application on a device with limited connection has three main states; paused, active and destroyed. When a MIDlet application is started, it goes automatically into the paused state. The function that takes it out of this state and makes it active is `startApp`. In this function the main functionality is initialized.

The main class of the client includes the three required functions for a MIDlet application, together with some other main functions which will be explained briefly in this section.

The constructor sets up the choice-form from where the users choose what types of facilities they want to be displayed in the map, and the command buttons used to proceed to the next screen or exit the application.

When running the application the `startApp` function makes the connection with the server (closer described in the next section) by calling the `connectServer` function, and sets the display to show the choice-form. In return from the connection to the server it receives a string. The string contains three parts, an x position, y position and the filename of the right piece of the map. These values are separated with a comma and are split into three different variables for further use by using the `indexOf` and substring methods on the received string.

```
int index = result.indexOf(',');

int lastIndex = result.lastIndexOf(',');

myPosX = result.substring( 0 , index );

myPosY = result.substring( index + 1, lastIndex );

mapString = result.substring( lastIndex + 1 );
```

Figure 9: Code for splitting the string into three variables

4.6.3.1 Communication with server

The function `connectServer` sets up and communicate with the server. It takes two in-parameters, `reqMessage` and `url`. The `reqMessage` is a string with the message/data the client sends to the server and the `url` is the Internet address to where the server is located.

The connection is set up by opening an Http connection to the server and setting the request method of the connection. In this case the request method is set to POST. The request is sent to the server byte by byte.

Then the client opens an input stream to get the response from the server. The response is read into a string buffer which is converted to the returned string `responseFromServer`.

4.6.3.2 Command functions

Central in an application for mobile devices is a command listener which observe if the user press any of the command buttons. When the class implements a command listener, it requires the class to have a `commandAction` function. It is responsible for executing the right functions when a command is pressed.

In this case `commandAction` stores the choices made by the user at the choice menu in an array for later use. If the user should instead press the "exit" command, it calls the function `exitMidlet`, which destroys the midlet.

4.6.3.3 Open a prospect in the Internet browser

The `mobileClient` also implements a function to open an URL address on the mobile device. The function `openBrowser` takes a URL address as parameter and makes a `platformRequest` with that address. The `platformRequest` method takes the argument and passes it to the platform's application management software (AMS). The AMS examines the argument and loads the appropriate service; in this case it will open the Internet browser.

4.6.3.4 MyCanvas class

The other class, `MyCanvas`, is handling all the graphical functions. The main function in this class is the paint-function which does the drawing to the screen.

The constructor mainly creates the command buttons used in the graphical screens, and adds a command listener to them.

4.6.3.5 Draw map

In a class which extends canvas it is required to have a function named paint, taking a graphical object as parameter.

The paint function first creates an image from the `imageString`. When the paint function is called the first time, the `imageString` is equal to the overview map of the Trondheim city centre and the map is drawn.

Since the constructor added a command listener to the commands, this class also has a function which listens if any command buttons are pressed. When the user pushes the 'locate' button from the overview map, the `imageString` will be set to the value returned by the server. Also when 'locate' is pressed, the Boolean variable `showFacilities` is set to true. This implies the user's position will be shown in the map, together the apartments and other user-chosen facilities.

The user's position is shown as a black star in the map and the apartments as black triangles. The chosen facilities are also shown as triangles, but in other colors. Parking spaces are blue, schools green and kindergartens are marked in red.

When working with a paint class, it can be called over and over again by making the call `repaint()`. This makes it easy to update the screen whenever you want, just make sure to clear the screen between every update or else the new drawing will be on top of the previous. To handle this in a simple way, we have made a Boolean variable called `clearDisplay`. When this is true, a white rectangle covering the whole screen is drawn. Therefore `clearDisplay` is set to true before every `repaint()` call.

All the positions in the map are drawn with a size based on a scale. This to make sure all the triangles is of the same size. The value of the scale is the number of pixels from the exact position to the edges of the triangle.

When drawing an exact position in the map we also have to take the map size in proportion to the screen size, into consideration. This have been solved by having an X and Y value of where in the screen the upper left corner of the map is positioned. Since the map is centered we take the difference of the image width and screen width divided by two, and we get the X value. And the same is done with the height to get the Y value.

4.6.3.6 Pointer events

A mobile application has also built-in functionality which listens for user made events, such as when keys or pointers are pressed.

We have implemented a function of the pointer events, `pointerPressed`. This function gives an X and Y value of where the user pressed in the screen. We have implemented it because if the user presses at one of the apartments, short information about that specific apartment will be displayed below the map. This gives the user more info before he or she eventually opens the prospect of an apartment. When pressing at one of the apartments, the URL which is sent to the `openBrowser` function described earlier will change to the Internet address of that specific apartment's prospect.

4.6.4 *Design of server*

The server will also be developed in Java. Since the client is developed in J2ME it is easiest to just dealing with one language. Since the http protocol is used for the communication, the server could have been developed in other languages.

The server application will be running on an http server to be easy accessible to the client.

4.6.4.1 Components

The server consists only of one class, `mServer`. This class includes the functions needed to serve the client with the necessary map information.

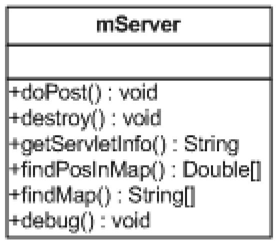


Figure 10: Class diagram of server

The class diagram above (Figure 10) shows the functionality on the server.

4.6.5 *Implementation of server*

This section will describe the functions on the server, and give the reasons for the choices made during the process.

4.6.5.1 Servlet

The server has been implemented as an http servlet running on an Apache Tomcat web server. Apache was chose because of its prominent profile in the market, and it is also used in the official Reference Implementation of the Java servlet. [30]

4.6.5.2 Mobile server

The `mServer` class includes the Apache servlet API to be able to provide the servlet functionality. It is required that an http servlet have a `doPost` or `doGet` function implemented dependent on the call from the client. In this case we have implemented the `doPost` function.

The `doPost` function handles the request from the client through an `HttpServletRequest` object, and responds through an `HttpServletResponse`, which makes it easy to deal with the communication between the server and client. The server reads the input stream from the request object and converts it to a string. In response the server creates a `ServletOutputStream` from the response object, and writes the response string directly to the output stream. The client is then able to read from the stream.

The server also includes two map functions for locating the user in the map and finding the right piece of the large map where the user is positioned. The map functions are described in the following sections.

4.6.5.3 Map functions on server

Since there do not exist any dedicated map server within the wireless Trondheim network, we decided to make our own simple type of server. This server takes basis of a large map of the city center, which has been cut up into 12 parts of 200 x 200 pixels, more viewable on mobile devices.

The map we have chosen to use in this project is ripped from the Google maps service because we haven’t succeeded in finding any other good maps to use. Google maps gives in our opinion the best representation of the city, with name on the main streets.

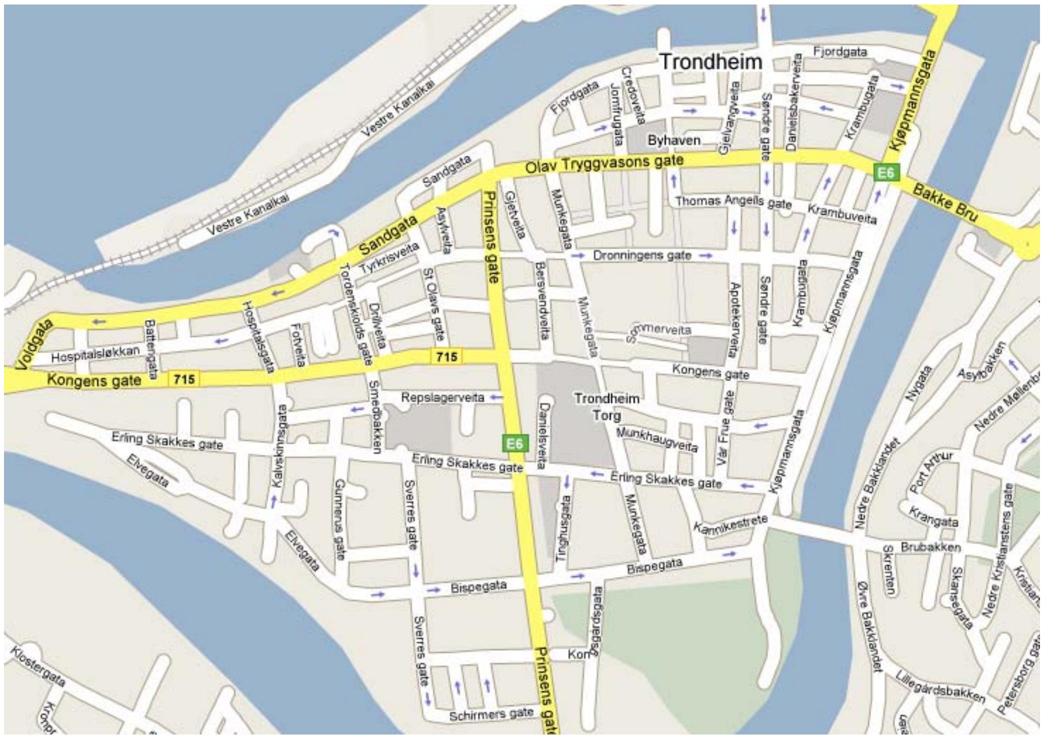


Figure 11: The map of the city center used in the project.

To cut the map into pieces of 200 x 200 pixels, we used an open source program called GeoreferenceTool. [31] The figure below (Figure 12) shows an example of a 200 x 200 map generated by the program. This program was also used to positioning the map. By marking 3 points in the map which you know the exact position of in EUREF93 coordinates, the program gives you a reference point in the top left corner of the map, which can be used to calculate any position in the map.

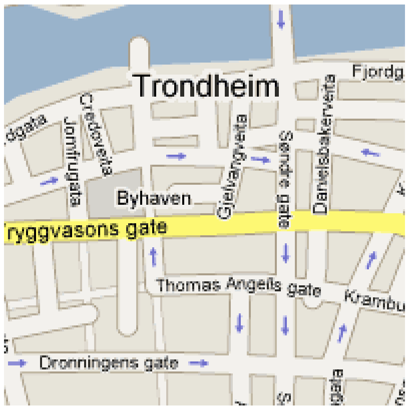


Figure 12: Example of a 200 x 200 map piece.

When the server gets a request from the client, it will calculate where in the map the client is positioned, find the right part of the map and make a response to the client where it is located. A deeper description of the location process you will find in the following subsections.

4.6.5.4 Location process for finding position in large map

When receiving the request from the client, the server sets up a fictive position in EUREF93 coordinates of where the client is located. The server will then run the `findPosInMap( posX, posY )` function, where `posX` and `posY` are the EUREF93 position, to calculate the clients position in the large map. The x and y values returned from the function are the distance from the top left corner of the map, in pixels.

To calculate the position, the function is based on the reference point in the top left corner of the map. This reference point is also in EUREF93 coordinates, and the function then calculates the difference from the position to the corner. Since EUREF93 gives coordinates in meters, we also get the distance in meters. When we used the GeoReferenceTool, it also calculated a scale for the map, the ratio meters per pixel – how many actually meters equals one pixel in the map.

4.6.5.5 Finding right map piece

After the server have found the position in the large map, it sends the result to the `findMap( posX, posY )` function. In the input to this function, the `posX` and `posY` are the pixel coordinates returned by the `findPosInMap` function. The function takes the two parameters and check if they are in certain intervals. An example is shown in the figure below.

```
if( ( pixelX < 200 ) ) {
      if( ( pixelY < 200 ) ) {
            imageFile ="/trondheim_midtbyen_0_0.png";
      }else if( ( pixelY > 200 && pixelY < 400 ) ) {
            imageFile ="/trondheim_midtbyen_0_200.png";
            pixelY = pixelY - 200;
```

Figure 13: Sample of the code for finding the right piece of map.

The function checks in this case if the X position is between 0 and 200. If it is, it checks the Y value. When it finds the right interval, the function sets the `imageFile` string to the corresponding filename.

The `findMap` function also alters the pixel positions, from referencing to a position in the large map it now reference to the position in the specific piece of map.

The server gets a string array in return from the `findMap` function. This array includes the new X and Y pixel position together with the `imageFile` string with the filename of the relevant map piece.

4.6.5.6 Response

After receiving the two positions and the `imagefile` string from the `findMap` functions, the server concatenates them into one string separated by a comma, returned to the client.

Chapter 5: Evaluation of prototype

We will introductorily in this chapter explain why evaluation is a vital part of the project, followed by what types of evaluation we have performed.

Different evaluations are important to systematic go through the developed system. The evaluation process consists of a technical evaluation, a user-evaluation and an expert evaluation.

In the technical evaluation we will run the application on different mobile devices to see how the functionality works and if there are any difference between the devices. This will also reveal potential differences in the Java implementation on the devices.

The user evaluation will be performed by one or several persons outside the development process, to get an objectively view and useful feedback from other people. The testers will give their opinion on what they think of the system, and suggestions for future improvements.

We will also send a document with description and screenshots of the application to some experts working at one of the major companies within the estate business, to get some important feedback. Together with the document we will send some concrete questions for them to answer. We would like to refer to Appendix B for the demo document with screenshots and the questions.

5.1 Technical evaluation

This part contains the technical evaluation of the application, which is a test of the functionality on different mobile devices.

5.1.1 Devices

The functionality will be tested on two mobile devices from two different vendors, HTC and Nokia. In this subsection we will briefly describe the two devices.

5.1.1.1 HTC TyTN

The first device we have tested on is the TyTN model from HTC. This device was chosen because it among other factors has a representative screen size for smart phones. It has also a touch screen and a slide-out keyboard. All phones from HTC are based on the Microsoft Windows Mobile platform.

Figure 14: HTC TyTN

5.1.1.2 Nokia N93

The other device used in the technical evaluation is Nokia N93. This device has a smaller screen and slightly more mobile phone look, but no touch screen. The N93 model is based on the Nokia S60 3rd edition developer platform and Symbian OS.

Figure 15: Nokia N93

5.1.1.3 Comparison of devices

The key facts of the two devices are collected in the following table (Table 1), to give a brief and quick overview of them.

Model name	**HTC TyTN**	**Nokia N93**
Platform	Microsoft Windows Mobile 5.0	Symbian OS, S60 software
Memory	64 MB SDRAM	Up to 50 MB dynamic
LCD	2.8" TFT-LCD 240x320 65 536 colors	2.4" QVGA display 240x320 262 144 colors
Connectivity	Bluetooth 2.0 Wi-Fi 802.11 b/g	Bluetooth 2.0 Wi-Fi 802.11 b/g
Java	CLDC 1.1 MIDP 2.0	CLDC 1.1 MIDP 2.0

Table 1: Overview of the devices

The largest differences between the two devices are the screen-size and the software. The devices run on two different platforms, something which can be an interesting element in the evaluation.

5.1.2 Functional requirements

We have defined some definite requirements that the application should satisfy. For a full list and a more detailed description of the functional requirements, see Appendix A. The requirements will be tested on both devices to see how the application works.

We will go through the requirements one by one to easily compare the application's performance and make a short comment to every requirement. In the end we will make a short conclusion where we sum up the outcome of the testing.

5.1.2.1 Installation

Functional Requirement 1: Installation	
Date:	21. May 2007
Evaluation / Result:	
HTC TyTN:	Successful. No problems during installation of the application
Nokia N93:	Successful. No problems occurred.

Table 2: Functional requirement - Installation

The application was installed successfully on both devices with no problems occurring.

5.1.2.2 Start-up

Functional Requirement 2: Start-up	
Date:	21. May 2007
Evaluation / Result:	
HTC TyTN:	The application starts up and the choice form shows perfect.
Nokia N93:	The application starts up and shows the form without any problems.

Table 3: Functional requirement - Start-up

The application started without any problems on both devices and showed the choice form in a clear and user-friendly way. Clearly the devices have different interpretation of fonts in mobile applications, because the font type was different from each other.

5.1.2.3 Choices made

Functional Requirement 3: Choices made	
Date:	21. May 2007
Evaluation / Result:	
HTC TyTN:	It is easy to tick off the wanted choices in the form, either by using the navigation button or the pen right on the touch-screen.
Nokia N93:	Easy to make the choices by using the navigation button.

Table 4: Functional requirement - Choices made

It is easy to make the wanted choices on both devices. The device from HTC also gives the possibility to use a pen on the touch-screen to make the choices.

5.1.2.4 Connection

Functional Requirement 4: Connection	
Date:	21. May 2007
Evaluation / Result:	
HTC TyTN:	The device gives a warning to the user that the phone wants access to the network, which can cause spending of conversation time if not connected to a wireless network. Connects to server without any problem.
Nokia N93:	On the Nokia device, a warning pops up with the question if you want the application to use a network to send or receive data. If you push 'yes' a list with possible ways of access. Either through wireless network or GPRS. No problems with connecting to the server.

Table 5: Functional requirement - Connection

Both devices connect to the server and receive the location data without any problems. Only difference is the two devices way of connecting to Internet. The device from HTC connects automatically through the wireless network if connected to one, while the Nokia gives you a choice even if you already are connected to a network. On the Nokia phone too many confirmations have to be done.

5.1.2.5 Overview

Functional Requirement 5: Overview	
Date:	21. May 2007
Evaluation / Result:	
HTC TyTN:	Shows the map and the text in clear way.
Nokia N93:	Shows the map, but the screen is too small to show the text below the map.

Table 6: Functional requirement - Overview

Both devices display the map nicely, but the Nokia's screen is too small to also show the text. The text is written behind the map which makes it impossible to see what it says. This screen is not so crucial to the application, and the most important is that the 'locate' command is available to the user which it is on both devices.

5.1.2.6 Locate

Functional Requirement 6: Locate	
Date:	21. May 2007
Evaluation / Result:	
HTC TyTN:	The map is displayed properly with all apartments and wanted facilities marked in it.
Nokia N93:	The map is displayed with all the relevant positions marked in it. The explanation of the different colors is covered by the map and is not possible to read.

Table 7: Functional requirement - Locate

The map is displayed properly on both devices, with all apartments and chosen facilities marked. On the HTC device the explanation is readable below the map, while on the Nokia it is covered by the map and is not readable. This makes it difficult to use the service on the Nokia device since you are unable to read what the different symbols in the map are.

5.1.2.7 Info

Functional Requirement 7: Info	
Date:	21. May 2007
Evaluation / Result:	
HTC TyTN:	It is possible to use the pen on the touch-screen and point on the apartments. When pointing at an apartment, two lines of small info about that specific apartment show below the map.
Nokia N93:	Not possible.

Table 8: Functional requirement – Info

The info function works well on the HTC, the related information about the apartment shows on two lines below the map. This function is not available on the Nokia device, because it has no touch-screen.

5.1.2.8 Browser

Functional Requirement 8: Browser	
Date:	21. May 2007
Evaluation / Result:	
HTC TyTN:	After pressing on an apartment's location in the map, and press the prospect command, the selected apartment's prospect opens in the Internet browser on the device.
Nokia N93:	Not available.

Table 9: Functional requirement – Browser

The browser function works without any problems on the HTC. The function is not available on the Nokia, because the link to the prospect is set when the user presses on a given apartment's location in the map. And since it is not possible to push on the screen on the Nokia device, the browser function has no link to send to the Internet browser.

5.1.2.9 Comparison

After having run the application on the two devices we can conclude that it doesn't work well on both. The HTC device has no problem carrying out all the functions, while the Nokia suffers from not having a touch screen, and therefore problems with the two last requirements.

The explanation is that the HTC device was available for continuously testing during mostly of the implementation process and the application is therefore well suited for it. While the Nokia have been brought in afterwards to have a second device for testing the platform independency. Since the Nokia is unable to perform the key functionality as 'info' or 'browser', it is not suited for this application without any adaptations are made.

5.2 Expert evaluation

This section includes an evaluation from the "experts" in the estate market. After finishing the implementation of the prototype we have sent a document with detailed description and screenshots to the estate agency to get their feedback on the system.

They appeared very interested in the initial phase of the project, but they showed no interest to come out with any ideas or point of view which could have helped us a lot in the startup of the work. Therefore all functionalities are based on our own thoughts of what will be helpful to include of services.

Feedback from the real actors in the market is a key factor for the evaluation of the project, because they are the persons working in the market and have continuous contact with the people looking for apartments. They have much knowledge of what people want, if it would be useful and they are also the ones who potentially will offer such a service to the customers in the future.

As said in the introduction, after having read the estate paper they sent to their customers on the 27th of March 2007 [1], and their positive conduct before the start of the project we really think we are on the track of a useful service. In the customer paper they repeatedly say that mobile technology is the future and their marketing director expresses: "in a short time the mobile will be the main channel of communication between estate agent, buyer and seller". [1]

5.2.1 Questions

To secure a certain quality and usefulness of the response from the agency, we sent a short document with some concrete evaluation questions together with the screenshots (Appendix B).

The questions we sent where:

- How useful/helpful will such a service be to persons looking for new apartments?
- How can the application be helpful?
- Seem the application user-friendly?
- Do you have suggestions for new functionality or possible modifications?

We also made a last comment in the document where we encouraged them to come with all general feedback they could think of.

5.2.2 Response

We got feedback from both of the employees of the estate agency we involved. We will in this subsection go through the questions we sent and the corresponding feedback we got.

The first question of how useful/helpful such a service would be they both replied that it would be very useful and interesting. They didn't give any elaborative explanation, just a short and concise answer.

On the next question, "How the application can be helpful", they detailed the response more. They said that map is a vital part of the information given to the users of where the houses are located, especially when you can use this information to also know where you are located in accordance to the houses when you are on your way to take a look at one of them.

They also liked the function of short information about the different apartments shown below the map when the user clicks on one of them together with the link to the specific apartment's prospectus, if the user finds the short information interesting. They say that this would make it easier to find other relevant apartments nearby open for viewing for people who like the neighborhood they currently are in.

The possibility to get the location of other types of facilities nearby displayed in the map, they also responded very positively on.

On the question of user-friendliness they replied that it seemed good, with a few exceptions. They gave the suggestion to remove the overview map since it really had no function, something we agree on. The overview map only leads to the user have to click one extra time. They also suggested it could be optional to show the apartments in the map as well, if you for example just wanted to get the nearest parking spaces.

We didn't get any direct feedback on the last question where we asked for suggestions to new functions or modifications - they only referred to the existing Internet services we have introduced in chapter 3, Finn.no and Igglo. But we have a few suggestions for such an application ourselves which we will come back to under the “future work” section in the Conclusion chapter.

We are generally satisfied with the response we got from the estate agency. The response was very positive and they seemed very interested in such a service.

5.3 User evaluation

The application has been tested by at least one person outside the project. One of the purposes with the application is that the user interface should be really easy to follow. The user should not have to do much to get the information he or she wants.

The users have been served some direct questions about the application to get concrete feedback, and to secure response on the topics we want. We haven’t had many people available so the user evaluation is unfortunately not a hundred percent objective.

We sent the same questions to the potential users as to the experts, but here we set the importance to the user-friendliness in this case. We used the same questions to also get some general response from the users and to get some suggestions for future improvements.

The response of this evaluation is that the functionality itself is user-friendly, but you have to confirm too many times the use of midlet and if you are sure you want the application to go online. It was also tested on two different devices and shows that a J2ME application is not directly portable between devices without small modifications.

We also got some suggestions for future improvements which we will list up here, and write further about in the future work section in the last chapter. The suggestions for further improvement were:

- More differentiated choices in the choice menu, for example between primary school, middle school and college.
- Also link to information about schools, kindergartens etc.
- Only display schools in the relevant school district dependent of house address.
- Possibility for manual positioning (i.e. to write in an address and go to it).
- GPS positioning.

Chapter 6: Conclusion

Continuously more and more services are developed for mobile devices to achieve higher availability. People want to use the services wherever they are and the service providers therefore must think mobile.

This thesis show how a mobile guide for geographically display estates listed for sale can be developed. The prototype has been evaluated in a user and expert survey together with a more technical way where the functionality have been tested on two different devices.

The Evaluation clearly indicated that such an application will be very useful for people searching for new apartments or houses. The estate agency we brought in came with exclusively positive feedback to the screenshots and description we sent them.

The feedback on the user-friendliness where also relatively positive, with a few exceptions with too many confirmations of connection to Internet and the screen with overview map which have no special function and can be deleted.

Results from the technical evaluation showed that even if the functionality worked relatively well on both devices, a J2ME application is not directly portable between devices without slightly modifications.

Based on the results from the evaluation of this project we think that such an application will be very useful for people looking for new apartments and houses.

If we where to do this type of work another time, we would at an early stage clearly choose what type of map service and positioning we would use. We used a little too much time on examining different technologies just to find them not useable or too complex.

6.1 Future work

This prototype is a first tryout of such an application for mobile devices and has given many new ideas for development of a fully functional system for the future. We will in this part present the ideas for future work we have at this point. This is ideas we have come up with ourselves and received through the evaluation of our prototype.

6.1.1 Profile

Users should be able to create a profile for later use, so that they don't have to make the same choices every time they use the system. Such a profile should be implemented in the already existing system by the estate agency. Such a profile can include type of housing, price range, size etc.

6.1.2 Positioning

The system should be able to retrieve the user's position automatically through GPS or GSM. This is a more real alternative than the use of wireless network. Users should also be given the possibility to manually give their current position, so they can type in their location and see where they should go.

6.1.3 Facilities

Other types of facilities can be shown in the map, for example nearest bus stop, bank, medical office or things like that. It can also be more differentiated choices regarding schools, primary school or middle school etc. Maybe the application also can be connected to the school regions.

6.1.4 Information

It can also be relevant to get information when clicking on one of the facilities as well. Also a link to the specific facilities web-page can be useful, since every facility today seem to have their web-page. It could be an option in the future to for example see directly how many notified vacancies a kindergarten or a medical office have.

References:

[1] DnB NOR. *"EiendomsMarkedet – Boligavis"*, 27. March 2007

[2] L. Barkuus & A. Dey, *"Location-Based Services for Mobile Telephony: a Study of Users' Privacy Concerns"*, Intel Research Berkeley , 2003

[3] Carnegie Mellon University. *myCampus*, http://www.cs.cmu.edu/~sadeh/mycampus.htm#intro, last visited: 29.May 2007

[4] Oslo Taxi. *Taxi direkte*, http://www.oslotaxi.no/templates/Page____112.aspx, last visited: 23.May 2007

[5] BNR 2talsystemer. *Oslo Taxi Trafikk (OTT)*, http://www.bnr.dk/, last visited: 23.May 2007

[6] M. Bjørge & L.M. Kristensen. *"Trådløs Byvandring"* summer project NTNU, 2006

[7] Michael J. Yuan. *"Enterprise J2ME – developing mobile java applications"*, Prentice Hall, 2004

[8] H. Karimi & A. Hammad. *"Telegeoinformatics – Location-Based Computing and Services"*, Crc Press, 2004

[9] Gule Sider. *Wap*, http://www.gulesider.no/mobil/wap.html, last visited: 23.May 2007

[10] Statens Kartverk. *Kart og geodata*, http://www.statkart.no/IPS/?module=Files;action=File.getFile;ID=21960, downloaded: 20.May 2007

[11] Statens Kartverk. *Innføring av Euref89*, http://www.statkart.no/IPS/IPS?module=Articles;action=ArticleFolder.publicOpenFolder;ID=479, last visited: 20.May 2007

[12] Garmin Ltd. *What is GPS?*, http://www8.garmin.com/aboutGPS/, last visited: 11. May 2007

[13] NASA. *Global Positioning System (GPS)*, http://leonardo.jpl.nasa.gov/msl/Programs/gps.html, last visited: 10. May 2007

[14] Cisco Systems Inc. *Cisco Wireless Location Appliance*, http://www.cisco.com/application/pdf/en/us/guest/products/ps6386/c1650/cdccont_0900aecd80293728.pdf , downloaded: 20.April 2007

[15] IEEE, *IEEE 802.11 The working group setting standards for Wireless LAN* , http://www.ieee802.org/11/, last visited: 12.May 2007

[16] Intel Corporation, *WiMAX Broadband Wireless Technology Access*, http://www.intel.com/netcomms/technologies/wimax/index.htm, last visited: 20. April 2007

[17] Trådløse Trondheim. *Wireless Trondheim*, http://www.wirelesstrondheim.no/ 13. May 2007

[18] John Krogstie. *Information Systems Engineering in Ubiquitous Environments: Wireless Trondheim as a city-wide international testbed for wireless services*, presentation, 2007

[19] Thomas Jelle. *Wireless Trondheim – A modern incubator for creativity, research and product development*, Presentation in Toronto, 2007

[20] Wireless Trondheim. *Coverage map*, http://www.wirelesstrondheim.no/gfx/dekningskart.gif 13.May 2007

[21] Google. *Google Maps*, http://maps.google.com/, last visited: 10. May 2007

[22] Google. Google Maps API, http://www.google.com/apis/maps/, last visited: 10 May 2007

[23] Yahoo! *Yahoo! Maps*, http://maps.yahoo.com/, last visited: 10 May 2007

[24] Yahoo! *The Yahoo! Maps Developer APIs*, http://developer.yahoo.com/maps/, last visited: 10. May 2007

[25] Microsoft Corporation. *Virtual Earth Interactive SDK*, http://dev.live.com/virtualearth/sdk/, last visited: 10. May 2007

[26] Microsoft Corporation. *Live Search*, http://maps.live.com, last visited: 10. May 2007

[27] University of Minnesota. *MapServer*, http://mapserver.gis.umn.edu/, last visited: 30. May 2007

[28] Sun Microsystems. *Sun Developer Network: Java ME at a Glance*, http://java.sun.com/javame/index.jsp, last visited: 30. May 2007

[29] Sun Microsystems. *Mobile Information Device Profile*, http://java.sun.com/products/midp/midp-ds.pdf downloaded: 13.March 2007

[30] The Apache Software Foundation. *Apache Tomcat*, http://tomcat.apache.org/index.html, last visited: 10. March 2007

[31] Christopher Michaelis. *Image Georeferencing Tool*, http://www.happysquirrel.com/index.php?feature=georef, last visited: 11.May 2007

[32] Ben Shneiderman. *"Designing the User Interface: Strategies for Effective Human-Computer Interaction"*, second Edition, Addison Wesley, 1992

[33] Hevner R., Ram S., March S. and Park. J. *"Design Science in Information Systems Research"*, MIS Quarterly Vol. 28 No. 1 pp. 75-105, March 2004.

[34] Drane C., Macnaughtan M. and Scott C. "Positioning GSM Telephones", IEEE Communications Magazine, April 1998.

[35] Finn.no. *Om Finn.no*, http://www.finn.no/finn/gojsp/daily/info/about.jsp , last visited: 11. June 2007

[36] Igglo Media AS. *Igglo*, http://www.igglo.no/, last visited: 11. June 2007

[37] Riggs R., Taivalssari A. and Vandenbrink M. *"Wireless Java 2 ME Platform Programming"*, Pearson Education, 2002

Appendix A: Functional requirements

This part introduces the different functional requirements in the table below.

Requirement	**Name**	**Description**
FR1	Installation	The device installs the application without any problems occurring.
FR2	Start-up	The application starts and the choice form shows.
FR3	Choices made	It should be possible to tick off the wanted choices.
FR4	Connection	An http connection to the server established.
FR5	Overview	After confirmed the ticked off choices, the Overview map is shown.
FR6	Locate	After user pressed the 'Locate' command, the right map with the user's position marked as a black star is shown. Also apartments marked as a black triangle and the wanted facilities from the choice-menu are shown.
FR7	Info	It's possible to press on the different apartments on the touch screen to get short information about that specific apartment.
FR8	Browser	The prospect of the last pressed apartment opens in the device browser, after pressing the 'Prospect' command.

Table 10: Functional requirementsments

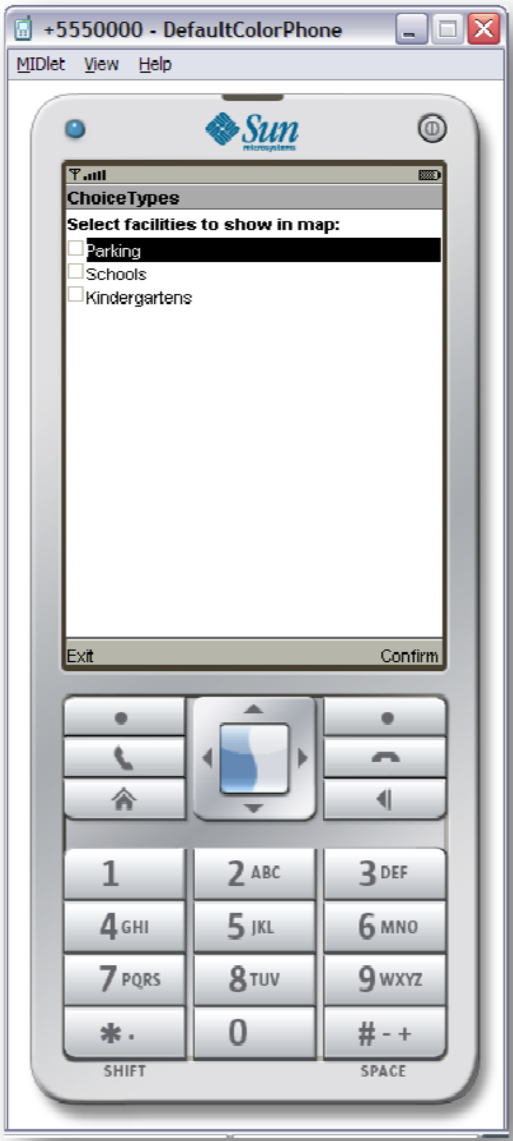
Appendix B:

Demo of Mobile guide

Guide for people searching apartments which matches their needs, running on a mobile device.

This application will give an overview through a map of the nearest apartments for sale and open for public view. The apartments will be marked on the map, together with the user's position and other user-chosen facilities such as parking spaces or schools.

Following is the screenshots from the application, to give an impression of how it works.



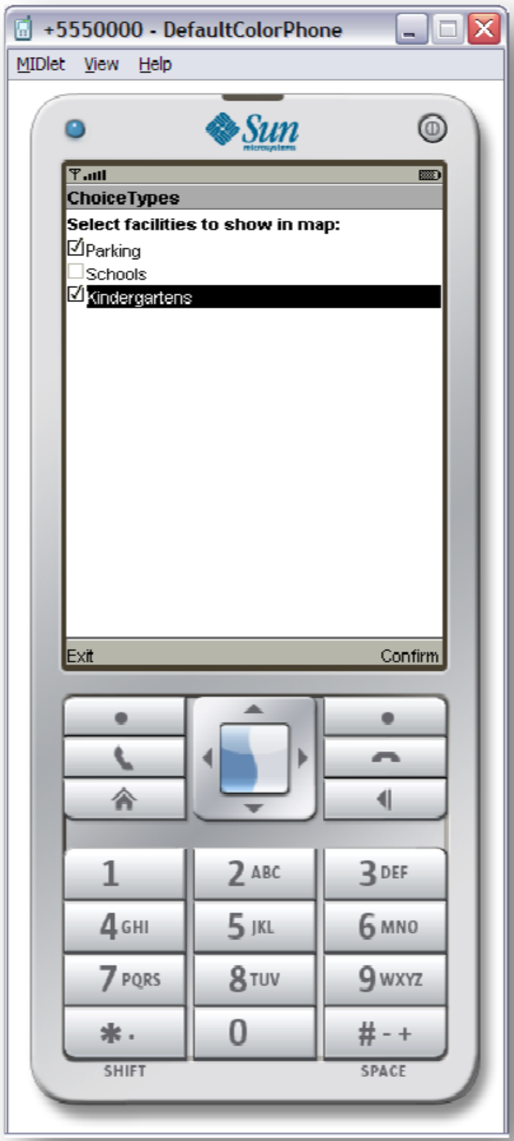
Select facilities to show in map:

- [ ] Parking
- [ ] Schools
- [ ] Kindergartens

Figure 16: Choice menu

The first screen of the application is the menu where the user chooses what types of other facilities (in addition to the apartments) the user wants to display in the map. To choose, tick off the wanted ones and click the button saying ‘confirm’ which will take you to the next screen.

The next screenshot is just an example of the menu where two of the choices are ticked off. This is also in accordance with the map in this demo.



ChoiceTypes

Select facilities to show in map:

☑ Parking

☐ Schools

☑ Kindergartens

Figure 17: Choice form with choices ticked off

After the user has confirmed the choices, the next screenshot is just an overview map. This screen has really no function, but is just meant as a reference for the user to get a quick overview of the city centre of Trondheim.

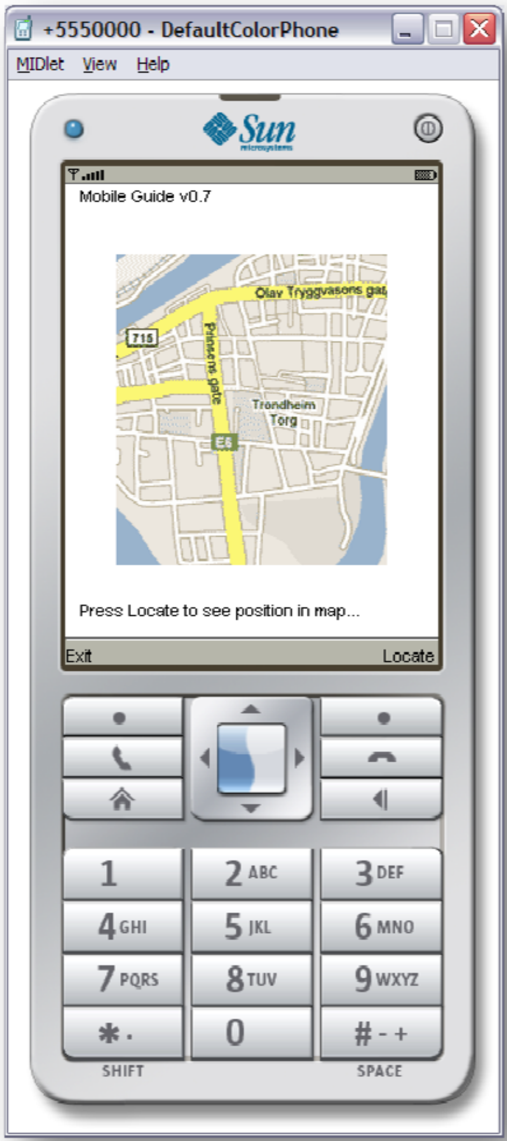
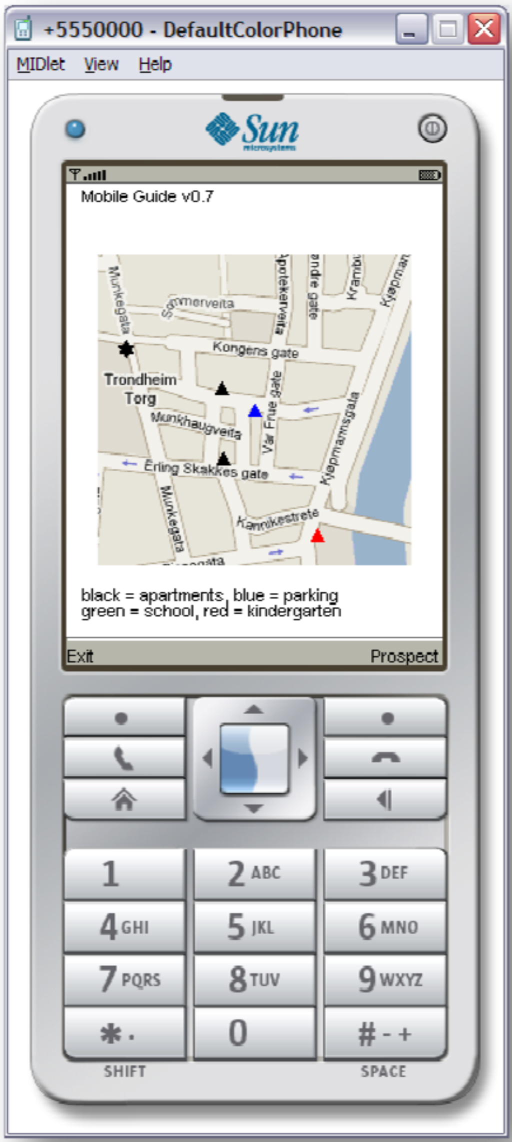


Figure 18: Overview map

The only thing the user have to do from the overview map, is to press the button named 'locate' and the application will display a more detailed map around the users current location, shown in the next figure.



Mobile Guide v0.7

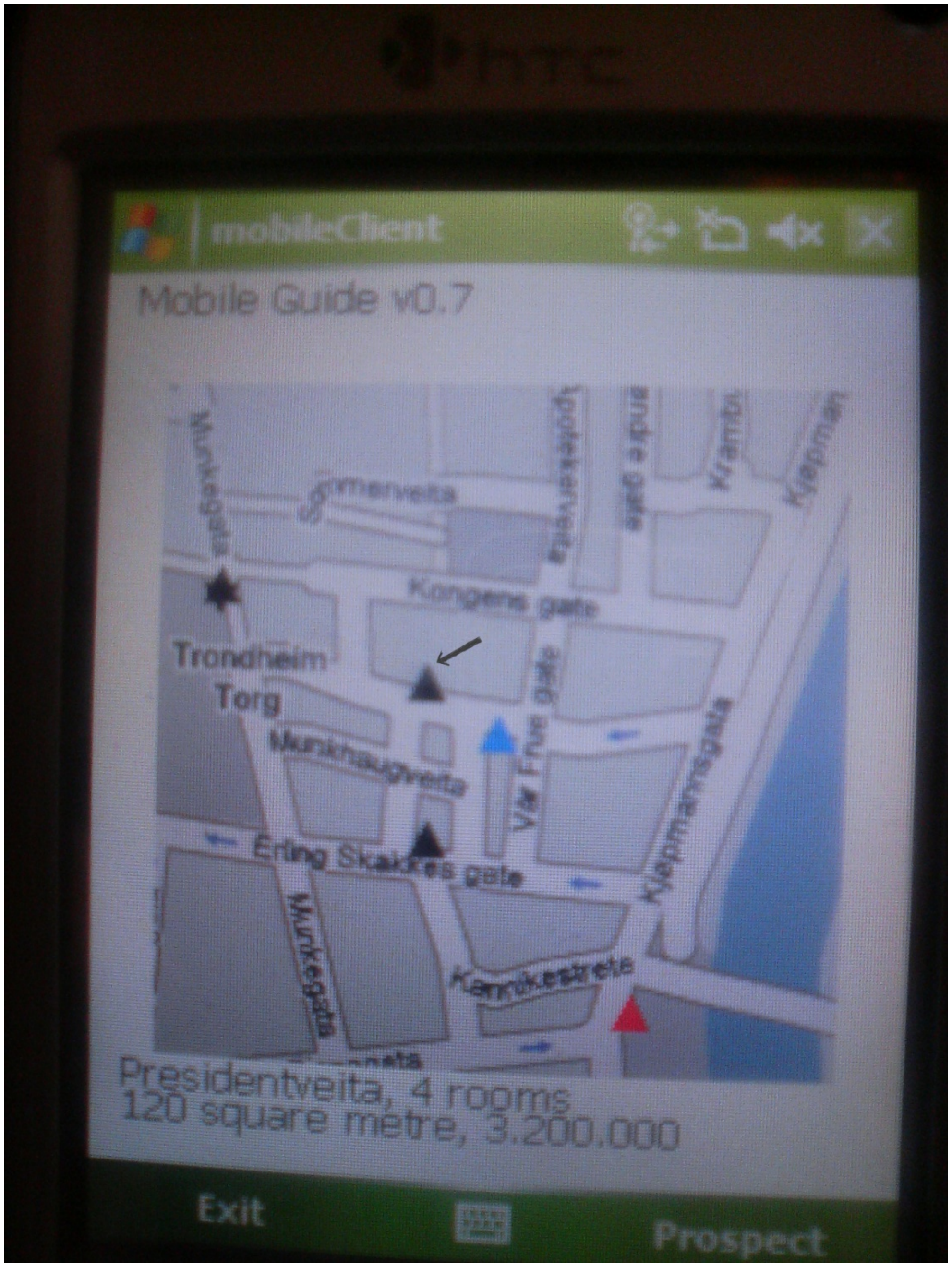
black = apartments, blue = parking
green = school, red = kindergarten

Figure 19: Detailed map with facilities

This map shows the users position marked with a black star. The other different facilities are explained below the map. The facilities shown in the map above are in accordance with the choices made in the menu earlier.

The two last screenshots had to be taken as a picture of the HTC TyTN smartphone, because the emulators on the PC don't support touch-screen functions.

The application gives the user the possibility to press on one of the apartments in the map, and a little info about that specific apartment will occur on the screen below the map.

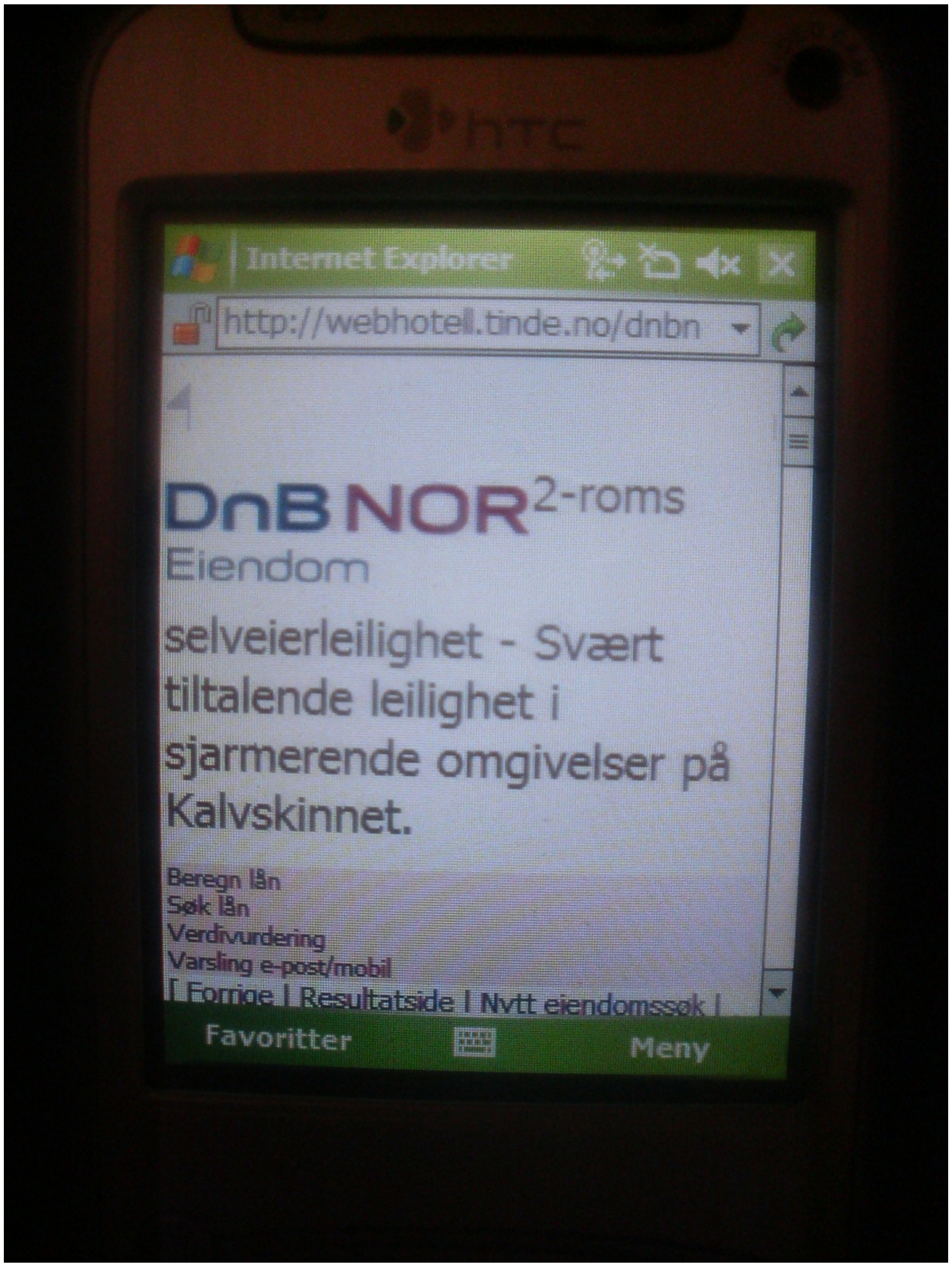


Presidentveita, 4 rooms
120 square metre, 3.200.000

Figure 20: Info of the selected apartment

Because we had to take a picture of the device with a camera, the quality of the image is not very good. But the most important to show is the info under the map which shows after pressing on one of the apartments on the screen. In the description under the map, the text is: “Presidentveita, 4 rooms, 120 square meters, 3.200.000 kr.”

In this screen it also is a ‘prospect’ command. By pressing this button the prospect of the selected apartment opens in the device’s Internet browser. An example is shown in the last figure, also this figure is a picture taken of the device, and the image is somewhat not clear. The links and prospect pages used are copied from DnBNORs web pages. Once again we have to remind that the prospect is just an example of how a link from the program works, and is not in correspondence with the apartment in the map.


Internet Explorer

http://webhotell.tinde.no/dnbn

DnB NOR Eiendom 2-roms selveierleilighet - Svært tiltalende leilighet i sjarmerende omgivelser på Kalvskinnet.

Beregn lån
Søk lån
Verdivurdering
Varsling e-post/mobil
[Forrige | Resultatside | Nytt eiendomssøk |

Figure 21: Prospect of selected apartment

Questions for evaluation of prototype

The purpose of the application is to show apartments nearby for sale and open for public view. In this prototype the user's position is set by the server. The apartments (marked with a black triangle) are clickable if the device has a touch-screen. The function "Prospect" is dependent on one of the apartments have been chosen, if not it has no link to open because the link is set when the user clicks on one of the apartments.

How useful/helpful will such a service be to persons looking for new apartments?

How can the application be helpful?

Seem the application user-friendly?

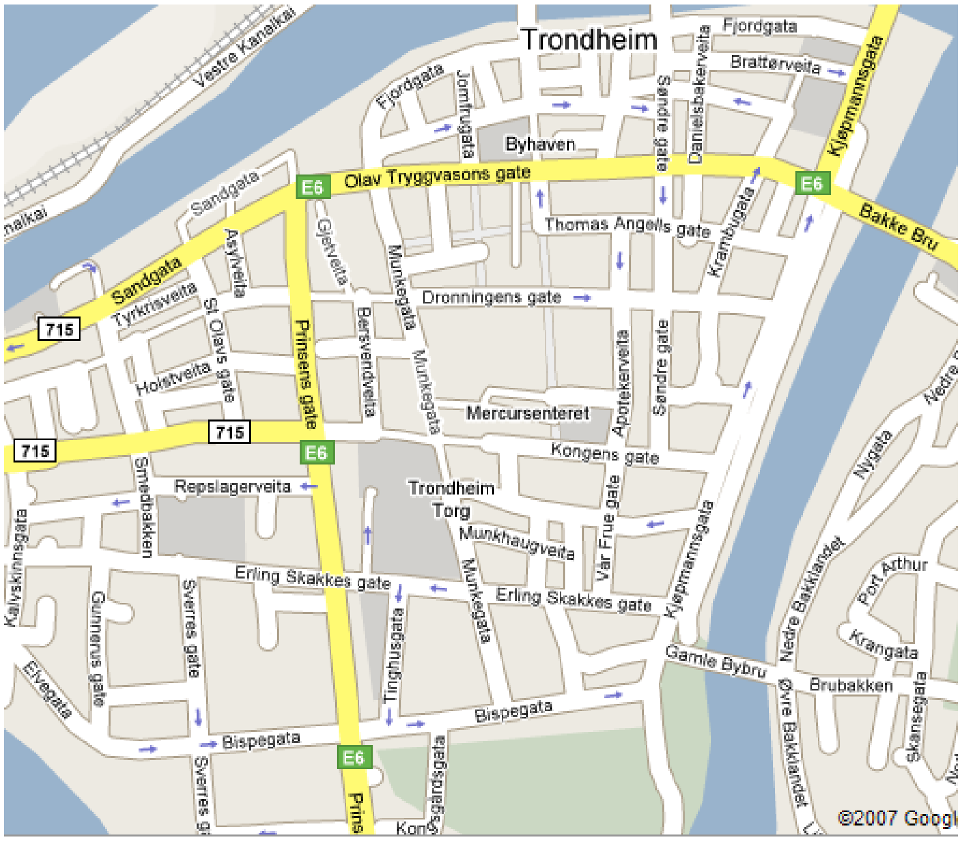
Do you have suggestions for new functionality or possible modifications?

Feel also free to come with all general feedback.

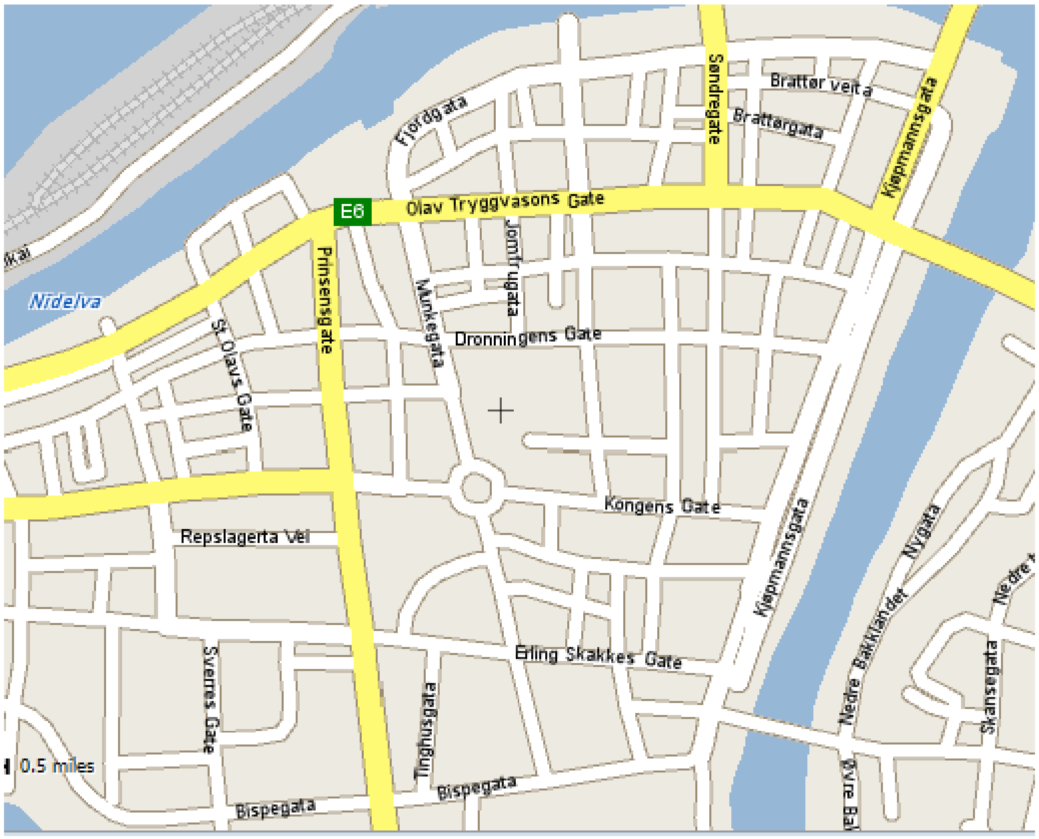
Appendix C: Maps

An example map of Trondheim city center with the map services mentioned in chapter 2, to give an overview of the detail level from the map providers. The maps have a near equally zoom level.

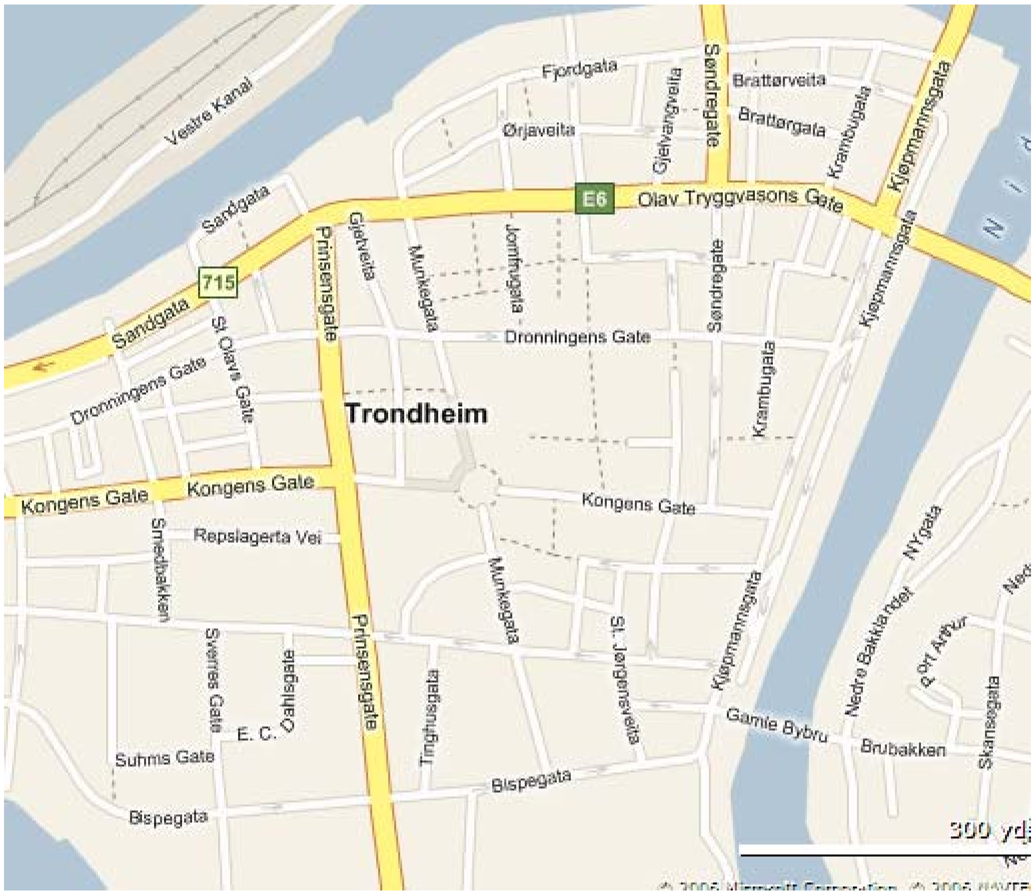
Google



Yahoo


Fjordgata
Søndregate
Brattør veita
Brattørgata
Kjøpmannsgata
E6
Olav Tryggvasons Gate
Jomfrugata
Munkegata
Prinsensgate
Nidelva
St. Olavs Gate
Dronningens Gate
Kongens Gate
Repslagerta Vei
Kjøpmannsgata
Nygata
Nedre Bakklandet
Nedre
Skansegata
Erling Skakkes Gate
Sverres Gate
Tinghusgata
0.5 miles
Bispegata
Bispegata
Øvre Ba

Microsoft


Vestre Kanal
Fjordgata
Søndregate
Brattørveita
Ørjaveita
Gjelvangveita
Brattørgata
Krambugata
Kjøpmannsgata
E6
Olav Tryggvasons Gate
Sandgata
715
Prinsensgate
Gjetveita
Munkegata
Jomfrugata
Søndregate
Kjøpmannsgata
Sandgata
St. Olavs Gate
Dronningens Gate
Dronningens Gate
Krambugata
Trondheim
Kongens Gate
Kongens Gate
Kongens Gate
Smedbakken
Repslagerta Vei
Munkegata
NYgata
Prinsensgate
St. Jørgensveita
Kjøpmannsgata
Nedre Bakklandet
Sverres Gate
Dahlsgate
Tinghusgata
E. C.
Gamle Bybru
Brubakken
Skansegata
Suhms Gate
Bispegata
Bispegata
300 yd

Appendix D:

Java documentation

This appendix contains the full java documentation of the application.

mobileClient

Package **Class** Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class mobileClient

```
java.lang.Object
  └ javax.microedition.midlet.MIDlet
      └ mobileClient
```

All Implemented Interfaces:

javax.microedition.lcdui.CommandListener

```
public class mobileClient
extends javax.microedition.midlet.MIDlet
implements javax.microedition.lcdui.CommandListener
```

mobileClient

Author:

Rune Romundstad

Field Summary

java.lang.String[]	choicesMade
javax.microedition.lcdui.Image	image
java.lang.String	mapString
java.lang.String[]	msg
java.lang.String	myPosX
java.lang.String	myPosY
java.lang.String	result

Constructor Summary

mobileClient()   Constructor Sets up the choice form and the Command buttons.

Method Summary

void	commandAction(javax.microedition.lcdui.Command command, javax.microedition.lcdui.Displayable displayable) commandAction listen on the command buttons.
void	destroyApp(boolean unconditional) destroyApp is also a required function.
void	exitMidlet() exitMidlet created to make a controlled closure for the application.

`void`	`openBrowser(java.lang.String urlAddress)` openBrowser gets a urlAddress as parameter, and opens the Internet browser with that url.
`void`	`pauseApp()` pauseApp is a required function to define.
`void`	`startApp()` startApp is a required function for a MIDlet.

Methods inherited from class javax.microedition.midlet.MIDlet
`checkPermission, getAppProperty, notifyDestroyed, notifyPaused, platformRequest, resumeRequest`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

mapString

`public java.lang.String` **`mapString`**

---

msg

`public java.lang.String[]` **`msg`**

---

result

`public java.lang.String` **`result`**

image

`public javax.microedition.lcdui.Image` **`image`**

---

choicesMade

`public java.lang.String[]` **`choicesMade`**

---

myPosX

`public java.lang.String` **`myPosX`**

---

myPosY

`public java.lang.String` **`myPosY`**

Constructor Detail

mobileClient

`public` **`mobileClient`**`()`

Constructor Sets up the choice form and the Command buttons.

Method Detail

startApp

`public void` **`startApp`**`()`

startApp is a required function for a MIDlet. It takes the application out of the pause state and makes the connection to the server.

Specified by:

`startApp` in class `javax.microedition.midlet.MIDlet`

---

pauseApp

`public void` **`pauseApp`**`()`

pauseApp is a required function to define. It takes care of what that should happend when the application is for example waiting to start.

Specified by:

`pauseApp` in class `javax.microedition.midlet.MIDlet`

---

destroyApp

```
public void destroyApp(boolean unconditional)
```

destroyApp is also a required function. It is used to quit the application.

Specified by:

`destroyApp` in class `javax.microedition.midlet.MIDlet`

Parameters:

`unconditional` -

---

exitMidlet

```
public void exitMidlet()
```

exitMidlet created to make a controlled closure for the application. It calls the destroyApp function with the boolean value 'unconditional' set to true. It also calls the function 'notifyDestroyed' to tell the system it is finished.

---

commandAction

```
public void commandAction(javax.microedition.lcdui.Command command,

javax.microedition.lcdui.Displayable displayable)
```

commandAction listen on the command buttons. If pressing 'confirm' it reads the choices made in the menu and stores it for later use. Or if pressing 'exit' it calls the exitMidlet function.

Specified by:

`commandAction` in interface `javax.microedition.lcdui.CommandListener`

---

openBrowser

```
public void openBrowser(java.lang.String urlAddress)
```

openBrowser gets a urlAddress as parameter, and opens the Internet browser with that url.

Parameters:

`urlAddress` - is the Internet address to the specific prospect.

Package **Class** Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

MyCanvas

Class MyCanvas

```
java.lang.Object
  └ javax.microedition.lcdui.Displayable
      └ javax.microedition.lcdui.Canvas
          └ MyCanvas
```

All Implemented Interfaces:

javax.microedition.lcdui.CommandListener

```
public class MyCanvas
extends javax.microedition.lcdui.Canvas
implements javax.microedition.lcdui.CommandListener
```

MyCanvas

Author:

Rune Romundstad

Field Summary

`int[]`	`apartmentsX`
`int[]`	`apartmentsY`
`java.lang.String`	`imageString`
`int`	`imgX`

int	imgY
java.lang.String	link

Fields inherited from class javax.microedition.lcdui.Canvas
`DOWN, FIRE, GAME_A, GAME_B, GAME_C, GAME_D, KEY_NUM0, KEY_NUM1, KEY_NUM2, KEY_NUM3, KEY_NUM4, KEY_NUM5, KEY_NUM6, KEY_NUM7, KEY_NUM8, KEY_NUM9, KEY_POUND, KEY_STAR, LEFT, RIGHT, UP`

Constructor Summary
`MyCanvas(mobileClient midlet)` constructor makes sure it's only one instance of the midlet object.

Method Summary	
void	`commandAction(javax.microedition.lcdui.Command c, javax.microedition.lcdui.Displayable d)` commandAction listen as in the other class on the commands.
void	`paint(javax.microedition.lcdui.Graphics g)` paint takes care of all the drawing to screen.
void	`pointerPressed(int x, int y)` pointerPressed listens if something is pressed at the screen.

Methods inherited from class javax.microedition.lcdui.Canvas

```
getGameAction, getKeyCode, getKeyName, hasPointerEvents,
hasPointerMotionEvents, hasRepeatEvents, isDoubleBuffered, repaint,
repaint, serviceRepaints, setCommandListener, setFullScreenMode
```

Methods inherited from class javax.microedition.lcdui.Displayable

```
addCommand, getHeight, getTicker, getTitle, getWidth, isShown,
removeCommand, setTicker, setTitle
```

Methods inherited from class java.lang.Object

```
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait,
wait
```

Field Detail

imageString

`public java.lang.String` **`imageString`**

---

link

`public java.lang.String` **`link`**

---

apartmentsX

`public int[]` **`apartmentsX`**

---

apartmentsY

`public int[]` **`apartmentsY`**

---

imgX

`public int` **`imgX`**

---

imgY

`public int` **`imgY`**

Constructor Detail

MyCanvas

`public` **`MyCanvas`**`(mobileClient midlet)`

constructor makes sure it's only one instance of the midlet object. It also creates and add the commands used.

Parameters:

`midlet` - is a object of mobileClient.

Method Detail

paint

`public void` **`paint`**`(javax.microedition.lcdui.Graphics g)`

paint takes care of all the drawing to screen. Both the images/maps and the drawing on the map.

Specified by:

`paint` in class `javax.microedition.lcdui.Canvas`

---

commandAction

```
public void commandAction(javax.microedition.lcdui.Command c,
                          javax.microedition.lcdui.Displayable d)
```

commandAction listen as in the other class on the commands. If 'exit' is pressed it calls the exitMidlet in the mobileClient class. If 'locate' is pressed it clears the display, sets the showFacilities true, sets the imageString and the explanation text and calls repaint to draw it all. If 'openBrowser' is pressed it calls the openBrowser function with the link set below in the pointerPressed function, as parameter.

Specified by:

`commandAction` in interface `javax.microedition.lcdui.CommandListener`

---

pointerPressed

```
public void pointerPressed(int x,
                           int y)
```

pointerPressed listens if something is pressed at the screen. And it gets where in the screen it have been pressed through the x and y inparameters. The functions then checks if the user have pressed at one of the apartments, and if it displays short information about the apartment below the map, and sets the link variable to the apartments prospect.

Overrides:

`pointerPressed` in class `javax.microedition.lcdui.Canvas`

---

mserver

Package **Class** Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class mServer

```
java.lang.Object
  └ javax.servlet.GenericServlet
      └ javax.servlet.http.HttpServlet
          └ mServer
```

All Implemented Interfaces:

java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

```
public class mServer
extends javax.servlet.http.HttpServlet
```

mServer

Author:

Rune Romundstad

See Also:

Serialized Form

Constructor Summary

mServer()	

Method Summary

void	`debug(java.lang.String text, java.lang.String message)` debug is used for writing out messages to the console, by taking to String parameters and displays them to the console.
void	`doPost(javax.servlet.http.HttpServletRequest req, javax.servlet.http.HttpServletResponse res)` doPost is the required function in a HttpServlet class.
java.lang.String[]	`findMap(double posX, double posY)` findMap is the function for find the right piece of the large map, and calculate the users position in the found piece.
double[]	`findPosInMap(double posX, double posY)` findPosInMap is the function for finding the users position in the large map

Methods inherited from class javax.servlet.http.HttpServlet

`service`

Methods inherited from class javax.servlet.GenericServlet

`destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletInfo, getServletName, init, init, log, log`

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

mServer

```
public mServer()
```

Method Detail

doPost

```
public void doPost(javax.servlet.http.HttpServletRequest req,
                   javax.servlet.http.HttpServletResponse res)
            throws javax.servlet.ServletException,
                   java.io.IOException
```

doPost is the required function in a HttpServlet class. Either doGet or doPost. doPost handles the request from the client by reading the ServletInputStream, and writing a response through ServletOutputStream.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Throws:

`javax.servlet.ServletException`

`java.io.IOException`

---

findPosInMap

```
public double[] findPosInMap(double posX,
                             double posY)
```

findPosInMap is the function for finding the users position in the large map

Parameters:

`posX` - is the users position in latitude and Euref89

`posY` - is the users position in longitude and Euref89

Returns:

a double array with an X and Y value for the users position in pixels for the large map. The values are the number of pixels from the top left corner of the map, in X and Y direction.

---

findMap

```
public java.lang.String[] findMap(double posX,
                                  double posY)
```

findMap is the function for find the right piece of the large map, and calculate the users position in the found piece.

Parameters:

`posX` - is the users X position in the large map, value returned from findPosInMap function.

`posY` - is the users Y position in the large map, value returned from findPosInMap function.

Returns:

a string array with: name of the right map piece together with an X and Y position in pixels in that specific piece.

---

debug

```
public void debug(java.lang.String text,
                  java.lang.String message)
```

debug is used for writing out messages to the console, by taking to String parameters and displays them to the console.

Parameters:

`text` - is the explaining text which shall be written out.

`message` - is the value the text variable explains.
