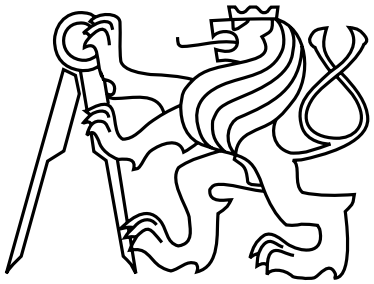
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Bachelor's Project

Mobile voting application for Android

Radovan Murin

Supervisor: Ing. Martin Komárek

Study Programme: Software technologies and Management, Bachelor programme

Field of Study: Software engineering

May 18, 2011

Aknowledgements

I would like to thank my supervisor, Ing. Martin Komárek, for his guidance and feedback during the creation of this work.

I would also like to thank my girlfriend Jane and my family for moral support.
Radovan Murin

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.
I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.Furthermore, this work is licensed under the Apache License, Version 2.0.

In Košice on Jan 2, 2011 .....

Abstract

The aim of this work is to design and implement an electronic voting application for the Android platform that will enable people to vote securely from anywhere. The application as a whole is aimed at being compatible with devices from many manufacturers and running different versions of the operating system. The application is also aimed at being localized.

Abstrakt

Cieľom tejto práce je navrhnúť a naimplementovat aplikáciu pre elektronické hlasovanie pre platformu Android, ktorá umožní ľudom hlasovať bezpečne z akéhokoľvek miesta. Celá aplikácia je zameraná na čo najvačšiu kompatibilitu medzi rôznymi výrobcami a verziami Androidu a jazykovú prenositeľnost.

Contents

List of Figures

List of Tables

Introduction

Voting is a delicate matter. Whoever the voter is, he or she usually wants to vote as transparently and freely as possible. In most cases, this is achieved by strict rules which often make it necessary for the voter to be physically present at the place where the voting takes place. But what about the one member of the board who is always traveling? What about a member of a council that is stuck in a traffic jam on the way to the council hall?
In today's world we are witnesses of a process that is merging two devices into one. The PDA - Personal Digital Assistant and the cell phone. This means that more and more people carry in fact powerful computers in their pockets that enable them to perform a lot of tasks without the need of being at their desks. It also enables a person to vote safely and easily while he is unable to attend a meeting in person.

This text, along with the source code, guides and installation files are located on the website http://code.google.com/p/mobile-voting-system/ .

Chapter 1

Description of the problem and goals

1.1 Description

The main idea is to have a system that enables people to vote electronically without the need of having to move to a voting machine. This should quicken the whole process of voting and eliminate the need for error-prone human vote counters and provide a new degree of user comfort, security and greater mobility. The problems of creating such a system can be divided into six areas:

1. Authentication and registration - determining whether or not the person behind the user name and password.
2. Data transfer and device security - ensuring that the devices communicate in a safe and efficient manner.
3. Voting - emulating the voting process so that it reflects real life voting, designing safe storage of the results.
4. Result presentation - presenting the voting results in an easy and accessible manner.
5. Device compatibility - making the resulting application to run on a large number of devices.
6. Ease of use - the software is aimed at the average user, the manuals and design must reflect that.

This work is based on Jakub Valenta's bachelor's thesis[1]. In his thesis, he created a fully functioning voting server,the part running on a personal computer and counts the votes, and a J2ME client, the part with which the voter actually votes. Now, in the conclusion of his bachelor's thesis, Jakub Valenta [1] states, that the ever changing and manufacturer specific implementation of Java ME, configuration CLDC[10] and MIDP[18] specification is the biggest obstacle in deploying his software on a larger scale. The aim of this work is to overcome this shortcoming and create a new application that will communicate with a server based on his work and improve the server.

1.1.1 Authentication and registration

The number of voters makes it tricky to register. The goal here is to make the registration process as quick and as unambiguous as possible.

1.1.2 Data transfer and device security

When voting on a delicate matter, privacy and voter intent cannot be tampered with. The voter has to be sure that his vote has been received without change and in the case if a secret ballot, unseen by a third party.
Another thing is physical security. The cell phone will never be safe from this perspective. People often leave them at hotels, public restrooms, they are often the target of small theft. Because of this fact, the phone application will have to be protected from unauthorized use. A more detailed analysis of this issues is in the chapter Analysis, section Security 2.5.

1.1.3 Voting

The voting process is described in Jakub Valenta's work [1] . The process is based on real life voting where people first see the question and possible answers. Then they choose their answer of choice and send their vote. The system acts just like a human vote counter and commissioner. It registers the participants, authenticates them, counts up the votes and publishes them on the web or wherever it is suitable.

1.1.4 Result Presentation

What purpose the voting would serve if the people wouldn't be able to see the results? The aim in this category is to present the results in an easily accessed human readable form.

1.1.5 Device compatibility

As stated in the introduction in this chapter, mobile devices tend to have different implementations of even standardized APIs(Application Programming Interface). This makes implementing one application for all devices impossible. Therefore, another goal is set - Multiple versions of the application. Each version will work on only one platform. By starting at the most widely used platform and continuing to the second, third...etc, it is possible to cover a rather wide range of devices.

1.1.6 Ease of use

The people that will be using this piece of software will not be IT professionals. This has to be taken in account when creating manuals and installation packages.The main aim is to make the software so easy to use that the average user will be able to benefit from it.

1.1.7 Already existing systems

Applications for the Android platform are distributed with the Android Market tool [16]. Presently, there are only a few "apps" that offer voting. All of which use an unsecured connection and use a server that has unknown security settings. Also none of these "apps" offer a J2ME application for a feature phone.
Apps for mobile voting (Android Market, state from 21.1.2011):

- "Handy Poll" by Marc Tan [15]
  The application Handy Poll is, as the name suggests, used mainly for polling. As an example I would state a situation where a student of psychology needs input data from a lot of people. He/She creates the questions and responses. Then, people see the questions and can respond to them without authentication.

- "android mobile voting" by Dare Labs for Sony Ericsson [13]
  Presently not functional. Advertised to be safe. I could not get it to work. Although the company website [14] is claiming that the application is fully functional.

1.2 Goals

The goal of my thesis is to enhance the voting system from Jakub Valenta's[1] thesis in such a way, that it will enable voting from a mobile device running Android[7] and thus broaden the spectrum of devices supporting this software. My aim is to make use of the larger capabilities that an Android device provides in comparison with a simple phone and enhance the user experience and create a baseline which future developers for different platforms can use. Another goal is to make the application usable in an international environment by introducing internationalization to the implementation.

1.2.1 Android side

The Android application is to be written from scratch in Java for Android (Dalvik VM[12]). It will enable the user to vote securely, provide the user with basic information about the candidates or choices - e.g. age, occupation, few words about ballot choices. The Android application will also be able to view the results of a closed question. In order to maximize the number of people able to use this software it will also be localized into several languages.

1.2.2 Server side

The main goal here, is to increase the amount of information the server provides about the question and itself without compromising the compatibility with the J2ME application. Next, is to implement a secure way of communicating with the clients so that the security will no longer be reliant on Wi-Fi encryption. Another goal is to be able to differentiate between clients connecting from the local network and users connecting from the Internet and enhance the authentication.

1.2.3 Feature phone side

The feature phone application will remain compatible with the edited server.

1.2.4 Summary

In summary the aim of this work is:

- To extend the capabilities of the server so that it will be able to communicate with the new device.
- To design and create a functioning Android voting application.
- Maintain compatibility with the existing J2ME client with minimal or no editing.
- Make the system user friendly.
- Ensure that the voting is safe.
- Ensure that stored voting credentials are safe after device theft.
- Provide the user a choice of language: English, French, Slovak and Czech. And alter/create the applications in such a way, that facilitates implementing new languages.
- Minimize the risk that the Android application will not work in future releases of android by using best practices for Android applications.

Chapter 2

Analysis

2.1 Vision

This electronic voting system will be based on the existing work of Jakub Valenta [1] and will enable people to vote electronically without the need of being at a specific place or having multiple vote counters employed. This will eliminate the problems of counting errors and provide, new degree of security and mobility. The system will be primarily designed for group decision making where the majority decides but it will be also usable for candidate voting. The whole voting process will take place from a mobile device. The system will be highly portable and usable in an international environment.

2.2 State of the software at the beginning

2.2.1 Overview

The system in the received state was fully functioning and able to run a ballot with a J2ME device.(This was tested only in an emulator, my own J2ME enabled device could not run the application.).The system enabled the user to vote in a very simple way. The voting commisioner chose to create an election, inputed the questions and possible answers, chose the threshold percentages for a successful voting event, created voters with a simple form and started the voting process. The voters then could vote via their cell phones. After some time the voting is stopped and the server displays the results in the results tab.

The communication was handled by HTTP. Jakub Valenta[1] states that he chose this protocol because of the guaranteed delivery it provides. But while HTTP is a reliable protocol, it is the TCP protocol that provides this functionality. The benefits as I see them are mainly that it is a standard and HTTP parsers are easy to use and already implemented in Java. Another benefit of HTTP protocol is that the devices provide feedback to each other about the success of an operation. On the other hand, the question whether it is not too much of a load for a Java ME enabled device to parse such requests arises. But that is out of the scope of this work.

2.2.2 Negatives

Suppose there is a ballot question such as this: "Do you agree with proposal no. 22?" The purpose of mobile voting is to be able to vote from wherever the voter is right in the moment and it is possible that he will simply not know the details behind such proposals because he is not physically present at the meeting. This is a problem when using the current system as there is no simple way of providing the details about the question at hand.
Another negative is that SSL is not implemented in communication with the server. This means that anybody who has access to the Wi-Fi through which the voters vote can see all data en route between the devices. And although the connection between a cell phone and the BTS[25] is encrypted, this encryption has been proven to be weak[26] and breakable in real time. This fact prevents the use of a cell phone network as a data carrier without encryption on a higher network layer. What is worse than seeing the data is changing them. When using an open network (no access control) virtually anyone with the right software can alter the data.
As stated previously, in spite the fact that the majority of the currently sold cell phones support J2ME, the application is unlikely to run on most of them because of compatibility issues.[1] Another negative is the lack of internationalization. The whole application had hard-coded Czech messages.

2.2.3 Positives

On the other hand the work has it's positive features. The fact that the voting was made so simple made running it on a limited device smooth and without problems. The installation, had some faults but they were quickly resolved. The user interface was very comfortable and straightforward. This enabled very easy initial setup even without a tutorial.
The communication protocol is very efficient and therefore enables the use in areas without 3G signal. (it will even work swiftly with a basic GPRS connection. Documentation was excellent. All in all, the application did what it needed and did it well.

2.3 Server

A server is the part of the system that counts the votes, listens to connections and authenticates people. As this role is a practically a single point of failure, it needs to be well designed, and also well implemented. When creating an application such as this, there is the issue of the number of concurrent connections and operations. This system is designed to be used in a small to medium sized environment with a maximum of 200 concurrent connections. A larger number would require expensive hardware equipment and/or load balancing. Load balancing will not be used or implemented. The server is also responsible for advertising its presence on the local area network, offer a means of voter registration and providing simple information about how to connect. The first part is solvable by using a UDP[24] broadcast[23]. In short, this will send a datagram to all the computers within a network with the information needed to achieve a connection as well as a friendly name of the server. I chose UDP broadcasts because UDP is unreliable, meaning the sender does not wait for a response. Also this is the standard way of sending adverts through the network. The second

part is solvable by multiple ways. First, and this is how Jakub Valenta's system works, the user creation is strictly local, and the administrator,the person in charge of creating the elections, creates the users manually. This requires the voters presence and it is painfully slow (imagine 200 people before some event standing in line to enter their passwords). What would be better is, to create a web page with a simple form where users would input their names and login credentials. A script would then aggregate this data in a file, possibly an XML file for simplicity, and then another program would be used when verifying the voters identity. Although this would not mean the disappearance of the 200 people lines, it would fasten the process significantly, as only a personal number would have to be verified. The third problem, the information and registration web page, is solvable by implementing a simple web server into the present server software, by using a HTTP server(Apache for example) in concurrency with the server software or using a small web server library. Using an external web server - such as Apache, IIS, nginx, would be very easy from an implementation point of view, however for the end user the installation and configuration of any of these servers would be a hardly overcomable obstacle. When looking for an already java library with a simple HTTPs server, none offered a reasonable SSL option and therefore I decided to implement it myself. I have previously mentioned a personal number. This number or alphanumerical combination will be used to verify the identity of the registered user. It could be his ID Card number, Drivers license number anything person bound. Because it will be possible to vote from the Internet, a new system of verifying the connecting user origin will be needed. The system is necessary when the jurisdiction or some other agreement stipulates that the voter is to be present at a specific place. The Wi-Fi's limited range ensures this. Because the two possible mobile devices will be connected, the server will have to support multiple types of connections.

2.4 Client - Personal Digatal Assistant or Smartphone

The client is the part of the system that provides an interface for the voter to log in, vote and check the results. This part should be easy to use, lightweight and most importantly compatible with a lot of devices and safe. The safety of the mobile device is for a larger analysis and has a dedicated section that follows this one. The problem of user friendliness is really a matter of designing the application to behave consistently with the operating system. Explanation: an Android user is knows how Android applications usually work and expects this one to behave in this manner. It would be very confusing to use the same design principles for an iPhone, which has one useful hardware button, and an Android phone which has at least four. The next problem is being lightweight. The application is running on a very limited device. Limited in computing power, bandwidth, memory and electrical power. In the context of our application, computing power is not really an issue as no computationally intensive calculations are done on this end and computations are not very demanding. The bandwidth is also not a big concern as the data that is send is plain text and as such is not very large. The main problem that we need to combat here is power drainage and memory consumption. Every open connection on a cell phone lowers it's power levels considerably. The phone will therefore have to use the connection as little as possible and close the connection after every request. For this the HTTP protocol comes very helpful. As conference events and voting can involve people that speak multiple languages,

the mobile application should be designed to be easily translatable. This is done by using external strings when making any writings in the application.
Platform specific analysis:

2.4.1 Feature phone

The feature phone is the most limited of all mobile devices capable of running third party software. It runs a very basic implementation of Java ME, has very little computing power and no multitasking capabilities. This added up means, that the client has to be very basic.

2.4.2 Android device

As an application was already written for the J2ME platform the next logical choice is Android. The benefit of Android[7] is that while Java ME implementations may vary, Android is mostly consistent and a piece of software written and tested on one device should work on another. Forward compatibility is not guaranteed but as the changes are mostly additive, it is highly probable [4]. The question here is the choice of the base version on for which the application will be written. The distribution of versions are in the figure below.

Platform	API Level	Distribution
Android 1.5	3	4.7%
Android 1.6	4	7.9%
Android 2.1	7	35.2%
Android 2.2	8	51.8%
Android 2.3	9	0.4%

Figure 2.1: Version Distribution of Android [5] from http://developer.android.com/resources/dashboard/platform-versions.html retrieved on 22.1.2011

As it is clearly depicted in the table the prevalence of Android 2.1 and higher is overwhelming. Therefore Android 2.1 was chosen to be the minimum version supported by the application.

2.5 Security analysis

As it was stated earlier, mobile devices are susceptible to a number of threads.

1. Physical theft
2. Man in the middle attack

3. Malicious applications posing as legitimate software in order to trick the user into giving away his login credentials.

4. Malicious servers

5. Rooted device

2.5.1 Physical theft

Physical theft poses a risk that the stored login information in the device. The most obvious solution would be not to store login credentials on the device. This would be inconvenient for the end user. Therefore a better solution would be, to request a master code upon every application launch. Another possibility for the software would be making an IMSI[17] number check and in an event of a different SIM, or no SIM being inserted into the device, the app would wipe all user data on launch.

2.5.2 Man in the middle attack

A man in the middle attack is a common attack where the attacker has access to the transfer medium (in our case air). The attacker can then see the data being exchanged or, what is worse, alter it. In order to protect against such an attack the whole communication has to be encrypted, preferably by SSL. This, along with a Certificate signed by a trusted authority should ensure that the application will always know if the communication is being tampered with. Proper training manuals will have to warn the users about the dangers of unsecured connections.

2.5.3 Malicious software

The graphics interface of the application can be replicated and the user made to believe that he/she is entering his login credentials to the real application. This problem can be avoided by signing the application, and proper user manuals.
Next, a malicious program can try to copy stored data of the voting app. This problem is addressed by using a private storage mechanism provided by Android. [6]

2.5.4 Malicious servers

Because this software is open-source, anybody can alter its behavior and run it. This creates a potential security thread where users could connect to a false server that is advertising on the same local area network. Here the users would give their login credentials in false belief that they are giving them to a legitimate server. In reality they would be disclosing them to a third person who could then use them to login to the original server and vote on their behalf. The same problem goes for the server information web page and registration web page. The advertising issue can be solved by careful network settings which are discussed in the deployment section. The web page problem solution is identical to that of the Man in the middle attack problem solution.

2.5.5 "Rooted" device

Normally, an application run on an Android device cannot obtain root access to the device. This ensures that application security is kept intact. Hidden application per application data is protected from being read and altered by the wrong application. This, however, is not true when running a "rooted" device. The applications on that device can obtain root access and read sensitive data. To combat this risk, the stored password will have to be encrypted, and the user warned about the risks of having root access.

2.6 Business process model

The business process model was created by Jakub Valenta and it depicts the process how voting is held without the use of an electronic voting system. [1] B.1

2.7 Requirements analysis

The requirements analysis is comprised of the requirements the system must achieve to be complete. The system must support all requirements concerning voting from Jakub Valenta's thesis and in addition, those stated further in this section.

2.7.1 Functional requirements

Server

1. The system will provide the user with source IP address filtration capabilities.
2. The system must provide detailed information about the questions.
3. The system will present connection information on a simple web site.
4. The system will enable the users to vote publicly. [1]
5. The system will enable the users to vote secretly. [1]
6. The system will enable new user registration. [1]
7. The system will evaluate the results of the vote.[1]
8. The system will be able to present the results.[1]
9. The system will guarantee the explicitness and the indubitability of the vote. [1]
10. The system will ensure persistent storage of data. [1]

Client

1. The application must enable the voter to login.
2. The application must enable the voter to view question information.
3. The application must enable the voter to view the results of a closed vote.
4. The application must enable the voter to vote.
5. The application must provide basic safety guidelines when first run.
6. The application must remember the connection information and login data to multiple servers.
7. The application must warn the user when using a non-secure connection when security is medium.
8. The application must enable multiple connections to various servers.

2.7.2 Non-Functional requirements

Server

1. The system must support both HTTPS and HTTP connections.
2. The system must be compatible with the current J2ME client.
3. The system must be able to communicate swiftly with any mobile connection.
4. The system must be secured from being tampered with by unauthorised personnel.

Client

1. The application must use as little power as possible.
2. The application must be usable on a variety of screen densities and sizes.
3. The application must be usable on a device with or without a touch screen.
4. The application must be easy to use and fast to learn.
5. The application must provide an easy way to add a new language.
6. The application must store the user's login information in such a way that other applications will not be able to access it.

2.8 Use Cases

The use cases describe the probable interactions with the system and how the system will react to them. They are fully described in the UseCases file enclosed with the thesis. The user can choose the server he wishes to log into, then can display the details of the questions and vote on them. Time and the Android system itself play a minor role in the use cases because of how Android manages memory. For the sake of simplicity application components are represented as one activity. Further information concerning the use cases are in the

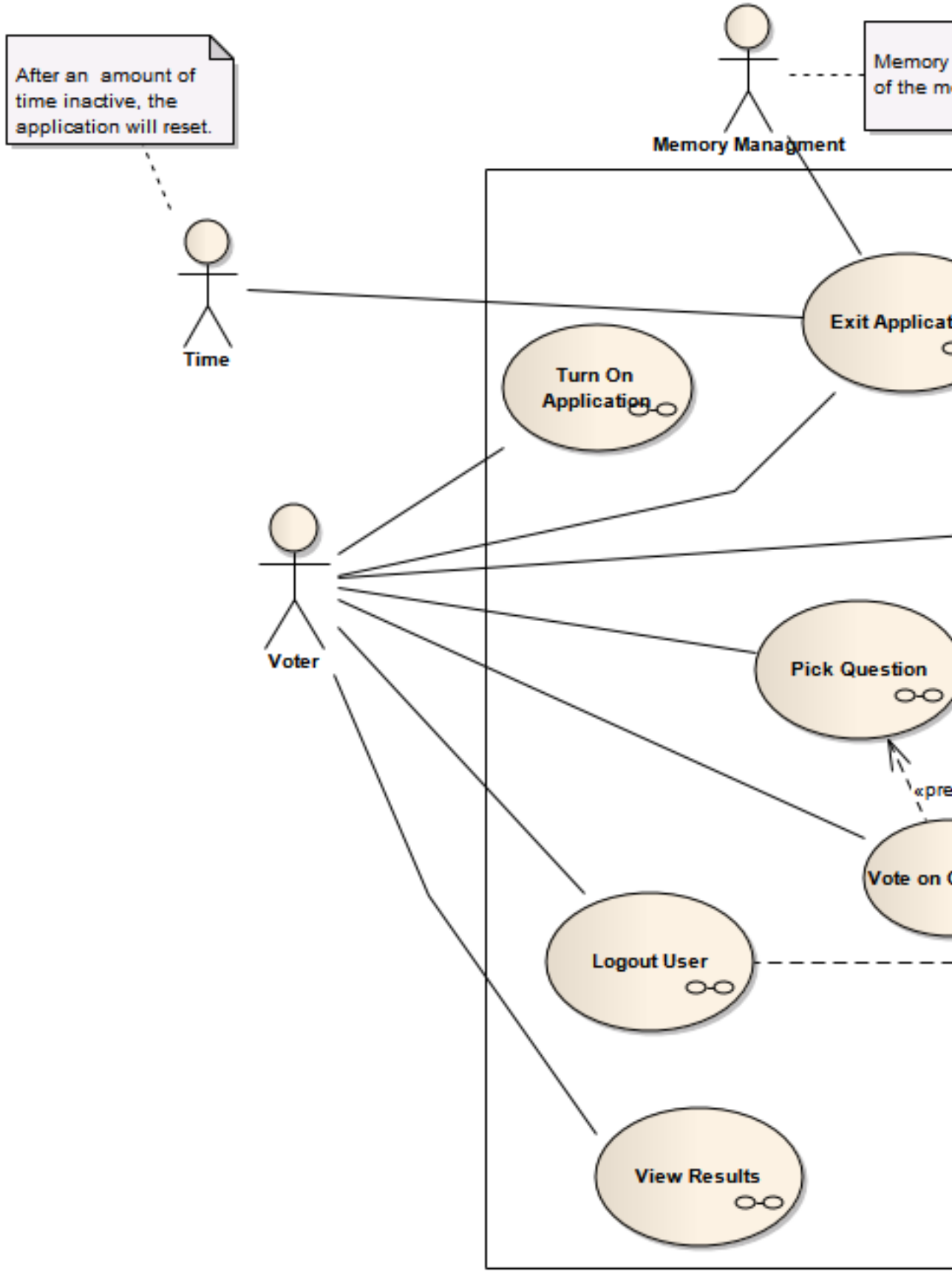


Figure 2.2: Use cases of the Android Application

Chapter 3

Design

The design of the "Mobile Voting System" is the original design of Jan Valenta which was extended in order to add functionality.

3.1 Communication

Presently, the communication is handled by the HTTP protocol. This protocol has, in my opinion, and advantage in being standardized and so it facilitates implementing the communicating part of the application. The protocol, however, is not safe. It sends data as open text and that is not acceptable for this kind of an application.
Therefore, Android applications will send a special "Hello" message that will consist of a request with the method OPTIONS specified and no body. The server then will send XML encoded data with open ports and whether they are using SSL or not. If there is no secure method of connection available, the application gives a warning to the user. This behavior will depend on the security settings on the server side.
The feature phone will continue to use HTTP as altering it's programming in such a degree is beyond the scope of this thesis. If the security will demand it, the server will reject any non-encrypted communication attempted through a mobile network. The connection termination will be preceded by a "419 BAD ORIGIN" code. Feature phones will still be able to connect through a Wi-Fi connection. Provided that the Wi-Fi is protected. This will also be customizable via the server settings.

This differentiation of the origination IP of the connection will be possible because of the new IP checking mechanism that will be added. When a client connects, the server will perform an IP check to determine whether the client is connecting from the a permitted network or not. The easiest way to perform this check is to see if the IP address is from the private subnets. As a packet with a private IP cannot survive on the Internet(it is discarded by the first properly configured router), it is safe to assume, that such a packet came from the local network. The reason why this check is performed is to ensure that the voter is physically present at the voting site when the legislature of some other rules require it. VPN technology is to be dealt with on the operating system level.

It is crucial that the client will not send the password when the communication is to be encrypted. Therefore, the first "hello" message has to be without a password. If a faulty client implementation sends the password unencrypted in this first hello message the server will invalidate the credentials of that user and add a log about it.

Data in HTTP answers will be sent in the XML format created by Jan Valenta. A new tag will house question details. Furthermore a new DTD will be created to form the XML data that will be sent when election results will be requested.

3.1.1 Summary of codes used in communication

Original[1]

- 200 Request success.
- 400 The request was not understood by the server. The server understands only GET and POST request.
- 401 Authentication credentials bad.
- 403 Authentication credentials required and not present.
- 404 Nothing to send.

Extentions

- 419 Bad origin.

3.2 Server side

3.2.1 Minor Changes

The server GUI will have some minor changes from it's original version. The questions table will be enhanced with a new column called "Question Description". Next, a new security setting dialog will be added when creating a new election event and it will now be possible to use custom TCP ports in case the server will have those ports blocked by a web server, or some other application. The persistence layer will mostly remain unaltered apart from the new "details" attribute. Because of the added features a new settings dialog will be created that will enable the full customization of the server behavior as well as language settings.

3.3 Android side

The android application will be designed with best practices for the platform in mind.

3.3.1 Graphical User Interface

This section sums up the design of the GUI system of the Android device. For a more complete description and guidelines on how to add new features, please see the attached document.
The application will have a minimalist user interface that will enable to use touch screens with ease.The main screen will consist of a list of servers that were added in advance. After choosing the desired server and a successful connection the application will view possible questions. Again in the of a list. The user will then choose a question he wants to view the answers for. Possible answers to a question will be displayed as buttons. After the desired answer is selected, the user will be able to send his response by long pressing the question button. The send button is hidden in order to reduce the risk of accidental sending.
When designing the GUI, the possibility of a third person being able to look at the devices display has to be taken into account. The passwords are never shown as text, therefore no risk exists in that somebody will see the password. However, there is a risk that someone will, on purpose or by accident see what answers are highlighted. One secure way of displaying the chosen answer is, of course, not to display them at all. This, however, poses more dangers than benefits. It introduces uncertainty for the voter about which answer they have chosen as touch screens are not very precise. A compromise is, that the choice will be highlighted for only a few second after the user has made his choice. With this in mind the user inteface will print out a small discrete message with the chosen answer for a shot amount of time after choosing the answer.
Every action will be confirmed by a notification.[1]

3.3.2 Persistence

The application will need to store user login credentials, server information and temporarily store questions and related data. The data that needs to be saved even after a system reboot - credentials and server info, will be in a private SQL database that Android provides for every application. The database,on an non-rooted[8] phone, is completely private and no other applications can gain access to it. If the device should be stolen and rooted, the password encryption should offer the legitimate user time, to reset his password and prevent any damage.
Another problem is, what will happen when the device receives a phone call or switches to some other application [1] at some point during the voting process. Because Android typically runs on low memory devices, it often simply kills an application when it needs more memory. This however, is not without saving the data structures belonging to that application first. The user notices nothing. This is entirely done by the Android system.

3.3.3 Communication

As it was stated in the previous section, Android kills an application when it runs out of memory. This can happen to the voting application too. When this happens, the connection would normally be lost. The platform thinks of this and provides a component called "service". The service runs independently from the graphical interface and has a lower "to

kill" priority when memory shortage occurs.

Because of this, the communication part of the application will be in the "service" component and will dynamically bind with new views in case the former got killed. The subsystem will have to be efficient and conserve battery as much as possible. It will also have to take into account the specific problems of a portable device - sudden loss of connectivity [1].

Chapter 4

Implementation

4.1 Tools

4.1.1 Programming language

The programming language defines some key characteristics of the final product. Some languages offer great optimization and some are good at being portable across many platforms. The language chosen for this project was Java and Groovy[9]. While the application could have been written in any other language, I chose Java as the programming language to implement it. First, concerning all the programming languages I know, Java is my most preferred because of the vastness of it's libraries, secondly Java is platform independent and that means that the time intensive task of recompiling the program for different platforms is removed. The third reason is because the base program was already implemented in Java and Groovy and a complete rewrite would be pointless. In my work, I only used Java and the reasons for using Groovy in some parts of the program are discussed in Jakub Valenta's bachelor's thesis[1].

Java platform A Java platform is a specific set of standards that enable Java written code to be run.
As my predecessor I have chosen to use the Java SE[20] platform. Java SE is a general-purpose platform and is well suited for this type of project.

Android platform Android applications can be written in either Java or C/C++. Programming applications for this platform in C/C++ however, has a number of drawbacks. One of which is bad portability. I is unlikely that an application written in this way will be very portable. For this reason I decided to use the recommended Android platform, which uses the Java programing language. This also made implementing the whole system easier as I was using only one language, albeit different platforms.

4.1.2 Third party libraries

By a 3^{rd} party library I mean a piece of compiled code that is used as a pre made solution for common problems that is not included in the original collection of Java SE libraries or

Android libraries.

XML Pull[32] This is the only third party library which is used in the implementation. While a part of the standard Android library suite this is not a part of Java SE. This library XML serialization and does this in a much simpler way than the traditional Java SE serializers. An example of use in my implementation is the method providing a fall-back generation of the registration web page.

```
1       public String makeIntroPage(String IP, int port) throws
            XmlPullParserException, IOException {
2         XmlPullParserFactory factory = XmlPullParserFactory.newInstance(
3                 System.getProperty(XmlPullParserFactory.PROPERTY_NAME), null);
4         XmlSerializer serializer = factory.newSerializer();
5         String xml = new String();
6         StringWriter os = new StringWriter();
7         serializer.setOutput(os);
8         serializer.setPrefix("", NAMESPACE);
9         serializer.startTag(NAMESPACE, "html");
10        serializer.startTag(NAMESPACE, "head");
11        serializer.startTag(NAMESPACE, "meta");
12    }
```

Robotium[2] Robotium is a testing suite that enables GUI action scripting for the Android Platform.

4.1.3 Version Control

When used properly, version control prevents changes that turn out wrong to have a long lasting effect. The developer simply rolls back the changes made. In this project version control was also used as a kind of backup in case of hardware failure. The system used was SVN and the tool used was TortoiseSVN. The hosting server was code.google.com. I chose these technologies because I had previous experience with them and they proved to be reliable.

4.1.4 User Interface

Server The GUI of the server application is implemented using Swing[19] and Groovy[9]. As stated beforehand Groovy was not used in this project but rather inherited. Therefore, it will not be discussed. Swing is a great toolkit for creating a simple GUI such as used in this project. To design the user interface I used the standard design tool that comes in Netbeans.

Android The Android framework does not enable the developer to use any other GUI framework other than it's own toolkit. The icons for the project are from an open source icon library[27], edited in MS paint when needed.

4.1.5 Cryptography

Creating a TLS[22] connection requires the server, the listening part, to have a digital certificate. For testing and development purposes a self-signed certificate was sufficient.To generate this certificate, and all other default certificates used in the application I used OpenSSL[28].

4.1.6 Persistence

Server The persistence tools of this side is documented in Jakub Valenta's bachelor's thesis[1].

Android For persistence of the Android application data a SQLite[30] database was used along with "Shared Preferences"[6] mechanism. The SQLite database is used for structured data - the server information, and the latter,"Shared Storage" is for storing primitive unstructured data.

4.1.7 Used integrated development environments and software development kits

There are a number of powerful IDEs to use with Java. For this project I have selected to use Netbeans for the Java SE part and Eclipse for Android. Netbeans was chosen because I had good knowledge of this tool and was confident with it's use. Eclipse was the recommended IDE by Google to develop Android applications. Both these tools were are very intuitive and pleasant to work with.
As for the SDKs used. For the server part I used the Oracle Java SE SDK. For Android I used the Android SDK downloadable from http://developer.android.com/sdk/index.html. This SDK also provides the developer with an emulator of a real Android device. However, taking in accounts it's performance issues I would strongly recommend using a real smart phone for developing future versions of this software.

4.2 Server

The server is a desktop application implemented in Java SE.

4.2.1 Prologue - information/registration server

The Prologue server is the implementation of a voting information and registration server defined in the design chapter. It is a part of the application. The server is implemented using a standard "HttpsServer" java class. This class provides basic request handling. Every request is passed to a handler class. There are two handling classes that this server uses the "registeringHandle" and the "providingHandle". The former has voting registration enabled, while the latter only provides the information necessary to connect with a mobile device.

Connections and encryption Clients can only connect to Prologue using a TLS[22] connection. The connection uses certificates that are used in the handshakes. Java enables the use of a number of certificate formats. First, a Java keystore format was used but then was dropped in favor of PKCS12[29]. The PKCS12 format has the advantage of being more user friendly and more widespread. The user friendliness comes from the fact that the PKCS12 format is the only standard format that enables private key storage.

Web pages The server on its start fetches all files that names match "index_[language-code].html". The language-code is the same as the codes in the Accept-languages[31] field in a HTTP request header. The reasons for this will be discussed later on. These pages are parsed and the relevant information such as server IP addresses and ports are added to them. A special "<-PUBLIC_IP->" and "<-PORT->" tag denotes the place where where information should be placed. This web page then serves a the home page providing the registered users all the necessary information needed for the actual connection. When the system does not find any index file that could be used as a main page, then a hard coded one is used as a fallback. The registration page is a simple form page that sends the relevant data to the server via the POST HTTP method. This page's name has the form of "register_[language-code].html".

Registration and approval The registration and approval are very close processes and despite the approval is not part of the Prologue server it will be discussed here. As stated previously the registration page is a simple form page that posts registration data to the server. The server checks for mismatching passwords, duplicate user names and then saves the registered person to a file called "registrations.xml". This is done to ensure that if the server crashes for whatever reason the registrations will be kept safe on the disk. Once the registrations are stopped the approval table is brought up and approved registrants are saved do a file called "approved.voter". This has the same format as the export file of the voters from the original system and therefore is easily imported into the application.

4.2.2 Settings

Because a big number of features being added to the system a centralized repository of settings was required. A class called "GlobalSettingsAndNotifier" has been created for this purpose. The class contains a singleton that has all the relevant setting data that affects the system. The instance also contains a list of listeners that are notified upon a specific changes. The settings are stored in key pairs <String, String>. A list of settings can be found in the attachments. Upon close the relevant settings are saved into a "settings.conf" file and are loaded upon the restart.

4.2.3 Internationalization

The whole system has been adapted to use with a multitude of languages. Upon load the settings class loads the appropriate language and provides the whole system with a library of terms. The application uses .properties files for this use.

```
english .properties file:
regFormTitle=Registration form
nameFormInput=Name:
surnameFormInput=Surname:

czech .properties file:
regFormTitle=Registracni formular
nameFormInput=Jmeno:
surnameFormInput=Prijmeni:
```

4.2.4 Beaconing

When the voting is started, the server will begin broadcasting a UDP packet that advertises it's presence to the voting devices.

The form of the packet, where friendlyName denotes the name of the server so that the end user can differentiate between multiple servers being run on the same network, the listenPost denotes the port the server is listening on, and a server id which is generated by random and clearly identifies the server for the end device - cell phone. This is used to prevent the same server showing up more than once in the end device menu.

```
<serverinfo id='" + serverID + "'><friendlyname>" + friendlyName + "</
    friendlyname><port>" + listenPort + "</port></serverinfo>
```

4.2.5 Security

SSL/TLS The connection is switched to TLS every time that is possible. The clients built according to the new specifications made in in this project will always upon connection send an OPTIONS HTTP request. After which the server will reply with a list of secure ports that it is listening to.
Example of a body of a response to a OPTIONS request.

```
<listenports>
  <port secure="false">10666</port>
  <port secure="true">11109</port>
</listenports>
```

This form was designed for future use with load balancing.

IP filtration The IP filtration is a security feature that enables the voting administrator to limit the physical locations the voters are able to vote from. The filtration itself is implemented by using a list of networkAddressRange classes. These classes contain the information for one network. Meaning the binary representation of the IP address, and the subnet mask also in binary. A string representation of these two values is also included. The class publishes a method called "isAllowed" that is used to determine if the address is allowed to connect.

```
1  public int isAllowed(int[] address, boolean isSecured) {
2          if (isOnNetwork(address)) {
3
4              if (action.equals(ALLOW_ANY)) {
5                  return 1;
6              }
7              if (action.equals(ALLOW_SSL) && isSecured){
8                  return 1;
9              }
10             if (action.equals(ALLOW_SSL) && !isSecured){
11                 return 1;
12             }
13             if (action.equals(DENY_ACCESS)) {
14                 return -1;
15             }
16             return 1;
17
18       } else {
19             return 3;
20         }
21     }
```

The class networkAddressRange also has a static method called isOnLan(int[] address). This checks whether the given address is on one of the three network ranges defined in RFC1918 as private.

4.3 Mobile device - Android

The Android application is implemented in the Java programming language. It uses the Android class libraries and is executed in the Dalvik Virtual Machine. For me this was my first experience and therefore the code underwent multiple refactorizations as my competence and confidence increased.

4.3.1 GUI

The GUI is discussed in the of the attachments as it was analyzed separately.

4.3.2 Communications

General The communications is implemented in an almost identical way as the server communications. The Connections class handles all the communications that are performed by the device. The class is written to be executed in a different thread so as not to slow down the GUI. The connection first attempts an unencrypted TCP connection to the server. Then as mentioned above asks for a list of ports with the OPTIONS request and if there are secure ports available it then connects to them. If the handshakes goes well it requests the list of questions and sends them to the XML parser and then it sends it to the view to display them. The responses are handled by a class called "Interceptor". This class is used in a very similar way to the Handler class on the server part.

In order for and application to access the network on the Android device, it has to ask for this permission in the AndroidManifest.xml. The permissions are added by inserting <uses-permission android:name="android.permission.INTERNET" /> into the XML.

SSL/TLS The implementation of a secure SSL/TLS connection on this platform is an interesting problem. Of course, the platform provides the user with the necessary libraries to to this but they all use and list of trusted certificate authority controlled by the device vendor. [1] In essence this is the correct behavior. But, for our deployment, a signed certificate is often unneeded or impossible to obtain - e.g. having a trusted certificate for a local IP is pointless. This means that a custom trust manager[2] is to be used that can ignore the exceptions and let the user decide whether or not she or he wants to trust the certificate chain.

4.3.3 Device security and Cryptography

Sensitive data that is stored on the device is encrypted. All encrypting/decrypting is handled by a singleton of a Cryptography class. The encryption is performed using DES[3]. While proved to be crackable in a matter of days, this protection is sufficient as it is unlikely that a stolen device would go unnoticed for this period of time. The key used in the process is the master password that the user creates upon the application first run. This password is never stored in any persistent storage. Only it's hash is stored for login purposes. Currently, only the login password to a voting point(server) is considered sensitive data and only this is encrypted. The implementation, however, enables encrypting virtually anything in the future.

4.3.4 Persistance

Primitives The android application needs to store the server connection data and the master password's hash. This is done using the "shared preferences" mechanism that the platform provides. The access to this store a custom PreferencesStorage class was created. This class is responsible for initializing the store and retrieving information.

Structured data Along with data primitives, the application also stores information needed to connect to a voting point/server. This is stored in a SQLite database. The access class for this database is called DatabaseStorage. This defines and provides all operations that can be used on the database. All data is saved in a database called ServerStore and in a table called "Servers". The table is created by calling CREATE TABLE IF NOT EXISTS Server (id INTEGER PRIMARY KEY AUTOINCREMENT, IPAddress VARCHAR, portN VARCHAR, uName VARCHAR, pass VARCHAR ,Fname VARCHAR);

[1] A certificate authority is an organization that is trusted enough that we know that if it certifies that "A" is "A" then a person trying to connect to "A" can trust it.

[2] A trust manages is a class that's purpose is to determine the validity of a given certificate.

4.3.5 XML Parsing and XML Creation

Virtually all usable data is communicated in XML form. To convert XML data into usable data the XMLParser class was created. The class has methods of parsing all the possible XML types and decides the type of the XML automatically. For sending data, it has to be converted into XML form. The XMLMaker class was created for this purpose. It implements all the XML creating processes uses in this part of the project.

Chapter 5

Testing

Testing plays an important role in this project. Testing early and often provided this project with a lot of precious feedback on how adding features and code altered the behavior of the system and enabled problematic parts to be quickly identified and repaired. The previous project [1] already had tests that were run on it. As the core classes of its business logic remained unchanged, the tests were left as they were and their results are still valid.

5.1 Unit testing

Unit testing is testing aimed at individual classes and their methods. It is a white-box testing method meaning that it is created with respect to the inner workings of the object tested and is aimed at finding common error such as when a null string is inputed and errors in implemented algorithms. In this project the Unit testing method of testing was used mainly on conversion classes, where data had to be converted from one form to another. These classes include the XMLParser, XMLMaker, the Cryptography class in the android project and the networkAddressRange class in the desktop project. Other classes were not tested. The unit testing is performed by the JUnitTest framework[11]. I found this framework very comfortable to work with as it comes embedded with the two IDEs I worked in and I had some experiences with it from past school projects. The test results are enclosed in the attachments.

5.2 Integration testing

Integration testing is targeted at finding out how the individual components work with each other. As because this is a client - server application, this part of testing was very important and it's results were often the main guidelines of how the project should continue. The testing itself was done by creating a mock voting server and trying to connect to it with the client software. Different combinations of inputs were used and errors were logged in the project logbook.

5.3 GUI Testing

Server The server's GUI was tested by hand. This approach was used because of the minimalist interface and the the fast responsiveness of the application. After each major modification of the GUI the new feature was tester for function and how well it integrates to the environment.

Android The android's GUI was tested differently than the server's. Because the platform's emulator is very slow, and constant installation on a real device is also very time consuming a need for an automated GUI test was created. Of all the GUI testing platforms, Robotium was found to have the most suitable properties. It is simple, requires minimal setup and integrates very well with the emulator software.
The test are presently covering ten out of eleven use cases and all of them are reporting passed. The only use case I did not cover was viewing the results. The reason is that this use case uses an external application that is tested by it's developers.

```
1 public void testGoodLogin() {
2     solo.clearEditText((EditText)solo.getView(R.string.passwordField));
3     solo.enterText((EditText)solo.getView(R.string.passwordField), "11111111")
          ;
4     solo.clickOnButton("OK");
5     solo.assertCurrentActivity("Passes did not match", cvut.fel.mobilevoting.
          murinrad.views.ServerListView.class);
6
7   }
```

5.4 Compatibility testing

By the term compatibility testing, I mean testing how the applications fare under different Operating systems and system versions. For the server application this is not an issue as being programmed in Java SE it is guaranteed to run on any operating system that has a JVM. Windows and Ubuntu were tested and they worked.The Android application was tested primarily on devices using Android 2.1. This was the base version. And on all the tested devices that were running the version, the application worked as intended. The application was tested on all the major versions from 1.5 up to 3.0 Honeycomb. Regrettably, 3.0 had some issues that I could not solve. Other than that, the application is working flawlessly on Android versions up to 2.3.3, which is by far the latest version used on cell phones.

5.5 Test conclusion

Every application and system, has issues.This one is no exception an. As old ones get fixed, new ones arise and it is impossible to create a flawless application. The tests that the application is supposed to pass before deployment are intended to emulate a voting process and verify that the system works. The unit tests have been passed and integration tests have passed. Both sides communicate without issues. GUI testing on Android has variable results which are a result of my inexperience with creating applications for Android. There

was no GUI testing for the server, only a basic monkey test. A monkey test is an application that crates random mouse events and keyboard events and inputs them to the application. They were used to check that all data that is inputted is checked before it is used and to try if the restrictions in the GUI, e.g. Voting cannot be started without a voter, are functioning.

Chapter 6

Deployment

6.1 Overview

For a graphic view of the deployment model, please see the figure in the UML attachments B.2.

6.2 Server deployment

The server is the system part that "listens" to incoming connections. It is free to download at http://code.google.com/p/mobile-voting-system/downloads/detail?name=VotingPoint.zip and it is also enclosed on the enclosed DVD. However, I strongly suggest downloading the latest version. The installation manuals are enclosed with this thesis.

6.2.1 Hardware

The server application does not have any specific hardware requirements and will run on any computer that is able to run the Java Virtual Machine.However, for a smooth voting process the voting application should run on a laptop with a healthy battery to withstand short power outages or a PC with a UPS(Uninterruptible power supply). The slowest system the application was tested on was an Intel Core Duo, 1.66 GHz, 1500MB RAM laptop and it seemed to run without any issues. The test was performed by using OpenSSL[28] testing commands for SSL. The results showed that the maximum connections per second were around 200. This value, however, does not represent the real capacity as the sever was very sluggish and was consuming a rather large amount of memory. The real capacity is around 100.

6.2.1.1 Connectivity

6.2.1.2 General

The server needs at least connectivity to a LAN(Local Area Network) for basic local voting(e.g. when the voting is to be held from a specific place) . For truly mobile voting,

the server needs a connection to the Internet. Although the for Internet connectivity the connection has to use the TCP/IP protocol suite, it should be noted that the LAN has to use this suite too. It is advisable to use cables instead of a wireless connection.
When Internet connectivity is required it is recommended that the server has his own external IP address to facilitate the configuration, however if that is not possible than at least access to the gateway device is needed in order to properly set-up port forwarding rules.
If voting is to be through the Internet, then the server should have a domain name and the name registered at a DNS. This would alleviate the shock the users often experience when working with IP addresses. If for some reason a static IP address is unavailable, the use of a dynamic DNS service is advisable. The dynamic DNS service serves just like a normal DNS, but the server/ADSL router has a small piece of software installed, that reports whenever it acquires a new external IP to the service and the service then updates the DNS entries. The whole process should take no more than 5 minutes and this time is acceptable.

6.2.2 Wireless Ethernet (Wi-Fi)

For rapid deployment, the computer should be able to behave as a wireless access point. One method that I tested and worked perfectly, was using piece of software called Connectify. This works only under Microsoft ®Windows®and does not provide many security features. However, for a quick and small deployment it is suitable.
For larger scale applications, a specialized Access Point is highly recommended. One that is able to block the forwarding of UDP broadcasts from the wireless interface. This reason for this requirement is, that the open nature of the server advertisements makes them prone to spoofing, and this would guarantee that only the adverts from the wired LAN are forwarded.

During testing, it was found out that firewalls block the ports used by the application. This means that the firewall has to be properly set up to allow traffic on the right port.

6.2.3 Operating system and software

The application is not bound to any specific operating system. It will run on any system that has a Graphical User Interface and has and iplementation of the Java Virtual Machine. The operating system should be running it's latest update.

6.3 Client deployment

The client is the part of this system that a voter uses to vote.

6.3.1 Hardware and software

Every client must be Wireless LAN enabled. The device can have a 2G or higher connection if the user wants to vote when a Wi-Fi connection is not available. The client devices must have their versions of the client application instaled

6.3.1.1 Android

The android device has to adhere to the Android OS Compatibility Definition Document [21].

6.3.1.2 Feature phone - J2ME enabled phone

Display and keyboard are necessary.

6.3.2 Connectivity

As it is with the server, for local voting it is sufficient for the client to be connected only to a LAN, but for remote voting Internet connectivity is required. In any case the connection has to use the TCP/IP protocol suite.

Chapter 7

Conclusion

The aim of this work was to enhance the mobile voting system originally created by Jakub Valenta in his bachelor's thesis. The goals were to implement SSL, add internationalization and create an Android client that follows the good practices for Android applications. These goals were supplemented with goals outlined in the analysis part of the project and developed as time progressed and I gained a better understanding of the area. The whole system performs well, however as I had no real knowledge of creating Android applications I was not able to resolve some performance issues on this side.The communication system is not handled by as an Android service but it is a part of a normal application. This does not affect the usability of the application but may affect performance. On the plus side the whole system is ready to be used in a real situation.For this, however, testing on a larger scale is recommended.

As for the continuation of this project, it needs more compatible clients. The analysis and design of the mobile client in this thesis is in my opinion a good foundation for the creation of other mobile clients - the iOS platform from Apple and BlackBerry OS from RIM are two platforms that come to mind as the targeted audience of this project is very likely to own one of these devices.

All issues and ideas concerning this project are logged on the project web site http://code.google.com/p voting-system/.

Bibliography

[1] VALENTA, J. Mobilní hlasovací zařízení [in Czech], 2010.

[2] web:code.google.com. robotium - It's like Selenium, but for Android™ - Google Project Hosting.
`http://code.google.com/p/robotium/`, Retrieved on 30. 4. 2011.

[3] web:csrc.nist.gov/. HTTP/1.1: Header Field Definitionse.
`http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`, Retrieved on 26. 4. 2011.

[4] web:developer.android.com. Android API Levels, .
`http://developer.android.com/guide/appendix/api-levels.html`, Retrieved on 8. 1. 2011.

[5] web:developer.android.com. Android Version Distribution, .
`http://developer.android.com/resources/dashboard/platform-versions.html`, Retrieved on 22. 1. 2011.

[6] web:developer.android.com. SQLite Home Page, .
`http://developer.android.com/guide/topics/data/data-storage.html#pref`, Retrieved on 26. 4. 2011.

[7] web:developer.android.com. What is Android?, .
`http://developer.android.com/guide/basics/what-is-android.html`, Retrieved on 26. 2. 2011.

[8] web:en.wikipedia.org. Rooting (Android OS).
`http://en.wikipedia.org/wiki/Rooting_(Android_OS)`, Retrieved on 17. 3. 2011.

[9] web:groovy.codehaus.org. Groovy - Home.
`http://groovy.codehaus.org/`, Retrieved on 25. 4. 2011.

[10] web:java.sun.com. Connected Limited Device Configuration (CLDC).
`http://java.sun.com/products/cldc/`, Retrieved on 08. 5. 2011.

[11] web:junit.org. JUnit.org.
`http://www.junit.org/`, Retrieved on 30. 4. 2011.

[12] web:market.android.com. dalvik - Code and documentation from Android's VM team - Google Project Hosting, .
http://code.google.com/p/dalvik/, Retrieved on 17. 3. 2011.

[13] web:market.android.com. android mobile voting, .
https://market.android.com/details?id=com.ts.sepolling&feature=search_result, Retrieved on 17. 3. 2011.

[14] web:market.android.com. dare work labs, .
http://www.thisisdare.com/work/labs/mobile_polling.html, Retrieved on 17. 3. 2011.

[15] web:market.android.com. HandyPoll, .
https://market.android.com/details?id=com.mlst.handypoll&feature=search_result, Retrieved on 17. 3. 2011.

[16] web:market.android.com. The android market, .
https://market.android.com/, Retrieved on 17. 3. 2011.

[17] web:mib.net.ua. GSM numbering plans.
http://www.mib.net.ua/2008/03/gsm-numbering-plans-en.html , Retrieved on 30. 4. 2011.

[18] web:oracle.com. Mobile Information Device Profile (MIDP), .
http://www.oracle.com/technetwork/java/index-jsp-138820.html, Retrieved on 08. 5. 2011.

[19] web:oracle.com. About the JFC and Swing (The Java™ Tutorials > Creating a GUI With JFC/Swing > Getting Started with Swing), .
http://download.oracle.com/javase/tutorial/uiswing/start/about.html, Retrieved on 25. 4. 2011.

[20] web:oracle.com. http://download.oracle.com/javase/™ - Google Project Hosting, .
http://download.oracle.com/javase/, Retrieved on 30. 4. 2011.

[21] web:static.googleusercontent.com. Android OS Compatibility Definition Document.
http://static.googleusercontent.com/external_content/untrusted_dlcp/source.android.com/sk//
Retrieved on 30. 3. 2011.

[22] web:tools.ietf.org. The Transport Layer Security (TLS) Protocol, .
http://tools.ietf.org/html/rfc5246, Retrieved on 25. 4. 2011.

[23] web:tools.ietf.org. RFC 919 - Broadcasting Internet Datagrams, .
http://tools.ietf.org/html/rfc919, Retrieved on 31. 3. 2011.

[24] web:tools.ietf.org. RFC 768 - User Datagram Protocol, .
http://tools.ietf.org/html/rfc768, Retrieved on 31. 3. 2011.

[25] web:wikipedia.org. Base transceiver station - Wikipedia, the free encyclopedia.
http://en.wikipedia.org/wiki/Base_transceiver_station, Retrieved on 17. 3. 2011.

[26] web:wireless.arcada.fi. Mobile and Wireless Communication Systems - GSM - Security. `http://wireless.arcada.fi/MOBWI/material/CN_1_6.htm`, Retrieved on 17. 3. 2011.

[27] web:www.openssl.org. Open Icon Library - Free/Open Icons, . `http://openiconlibrary.sourceforge.net/`, Retrieved on 25. 4. 2011.

[28] web:www.openssl.org. OpenSSL: The Open Source toolkit for SSL/TLS, . `http://www.openssl.org/`, Retrieved on 25. 4. 2011.

[29] web:www.rsa.com. RSA Laboratories - PKCS 12: Personal Information Exchange Syntax Standard. `http://www.rsa.com/rsalabs/node.asp?id=2138`, Retrieved on 26. 4. 2011.

[30] web:www.sqlite.org. SQLite Home Page. `http://www.sqlite.org/`, Retrieved on 26. 4. 2011.

[31] web:www.w3.org. HTTP/1.1: Header Field Definitionse. `http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html`, Retrieved on 26. 4. 2011.

[32] web:xmlpull.org. XML Pull parsing. `http://www.xmlpull.org/`, Retrieved on 25. 4. 2011.

Appendix A

Abbreviations

CLDC - Connected Limited Device Configuration

MIDP - Mobile Information Device Profile

LAN - Local Area Network

SSL - Secure Socket Layer

TLS - Transport Layer Security

DNS - Domain Name Service

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

DTD - Document Type Definition

VPN - Virtual Private Network

IP - Internet Protocol

HTTP - Hyper Text Transfer Protocol

HTTPs - Hyper Text Transfer Secure Protocol

VM - Virtual Machine

IMSI - International Mobile Subscriber Identity

Appendix B

UML diagrams

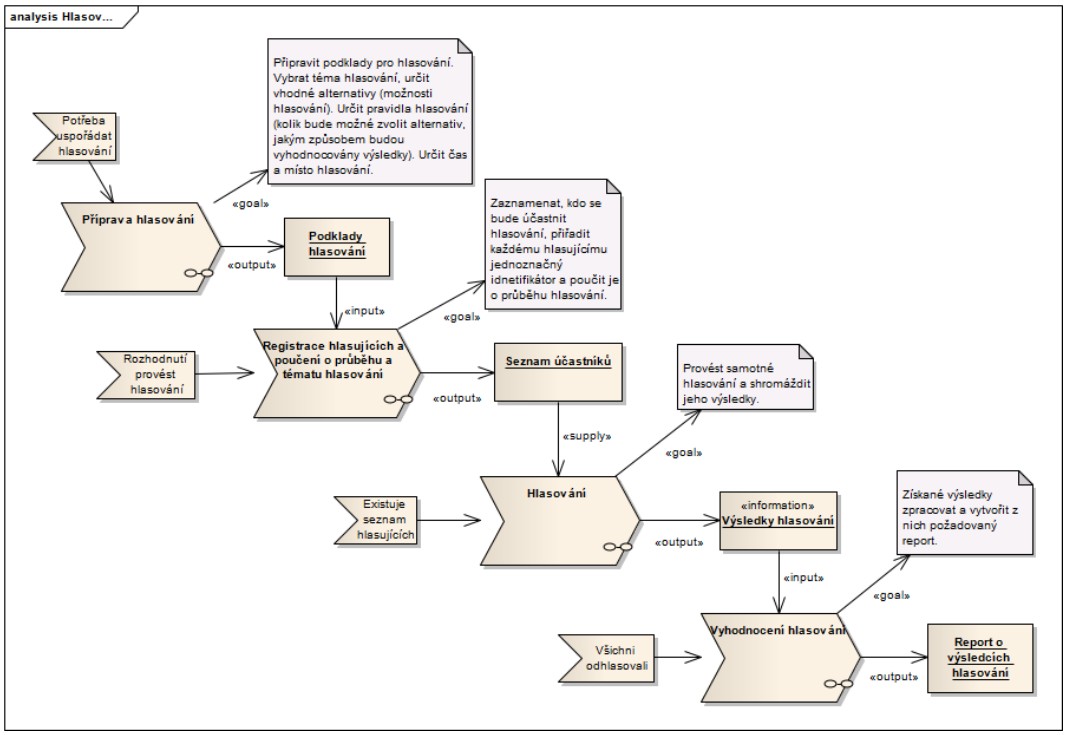


Figure B.1: Business process model of voting, model by Jakub Valenta

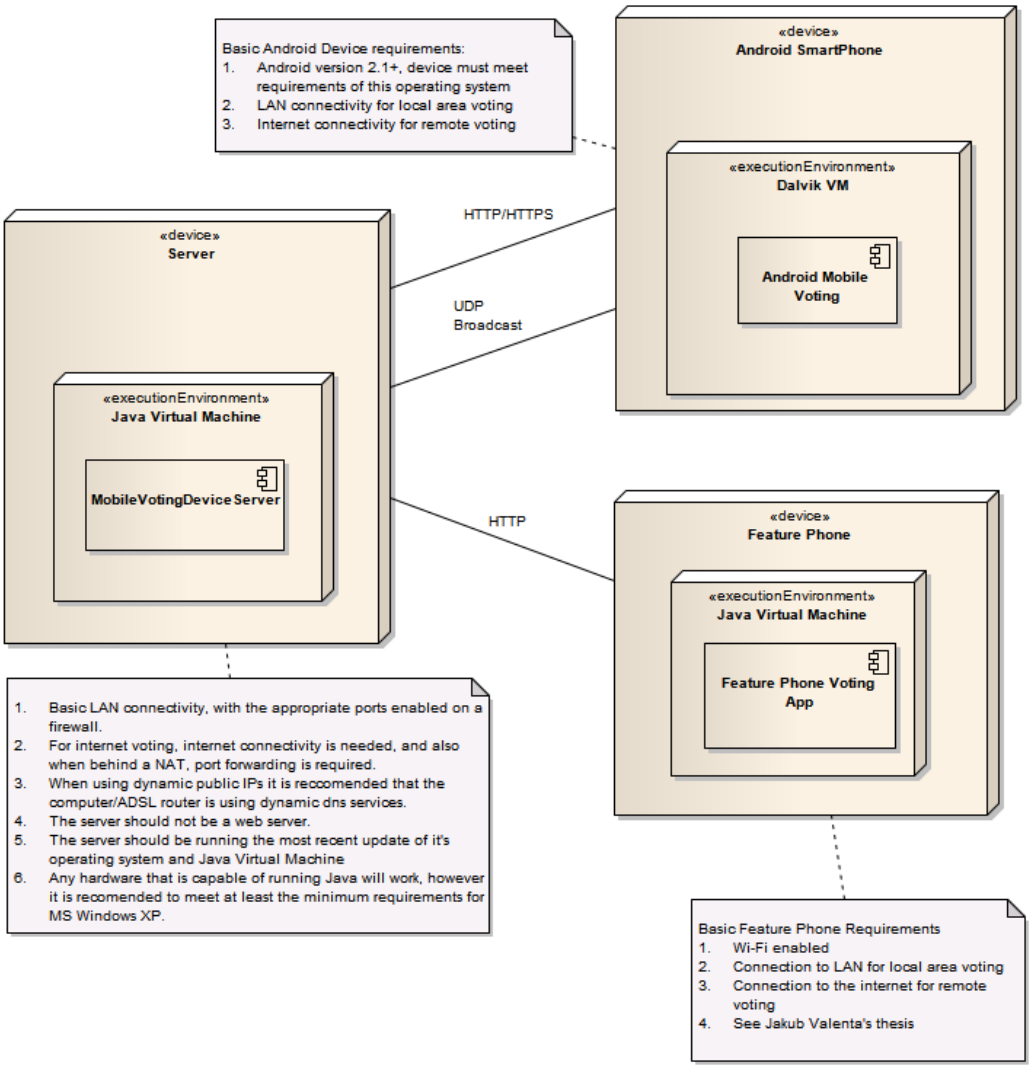


Figure B.2: The deployment diagram

Appendix C

Manuals

Appendix D

Enclosed CD Table of Contents

CD NOT YET CREATED