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

March 31, 2011

Aknowledgements

Placeholder

Declaration

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 localised.

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

Chapter 1

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. 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.

Chapter 2

Description of the problem and goals

2.1 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 of error-prone human vote counters and provide. The problems of 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 voter is aimed at an average user, the manuals and design must reflect that.

This work is based on Jakub Valenta's bachelor's thesis[1]. In this 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 and MIDP specification is the biggest obstacle in implementing 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.

2.1.1 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.

2.1.2 Data Transfer and device security

When voting on a delicate matter, privacy and voter intent cannot be tampered with by for example a man-in-the-middle type attack. 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.
Next, the nature of how the Android platform enables parts of an application to launch within another application. The resulting "app" will have to deal with this.
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 Security3.5.

2.1.3 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 answers. Then they choose their answer of choice and send their vote. The system acts just like a human vote counter and counts up the votes and publishes them on the web or wherever it is suitable.

2.1.4 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.

2.1.5 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 one 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.

2.1.6 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.

2.1.7 Already existing systems

Applications for the Android platform are distributed with the Android Market tool [11]. 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 [10]
  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 [8]
  Presently not functional. Advertised to be safe. I could not get it to work. Although the company website [9] is claiming that the application is fully functional.

2.2 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[5] 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.

2.2.1 Android side

The Android application is to be written from scratch in Java for Android (Dalvik VM[7]). 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.

2.2.2 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 authentification.

2.2.3 Feature phone side

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

2.2.4 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 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 of the Android application a choice of language: English, French, Slovak and Czech. And create the application 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 3

Analysis

3.1 Vision

This electronic voting system will be based on the existing work of Jakub Valenta/citebakalarkaJV and will enable people to vote electronically without the need of a person. This will eliminate the problems of counting errors and provide a new degree of security. 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.

3.2 State of the software at the beginning

3.2.1 Overview

The system in the received state was fully functioning and able to run the ballot with a J2ME device.(This was tested only in an emulator).The system enabled the user to vote in a very simple way. The user 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 layer 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. 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.

3.2.2 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[13] is encrypted, this encryption has been proven to be weak[14] 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]

3.2.3 Positives

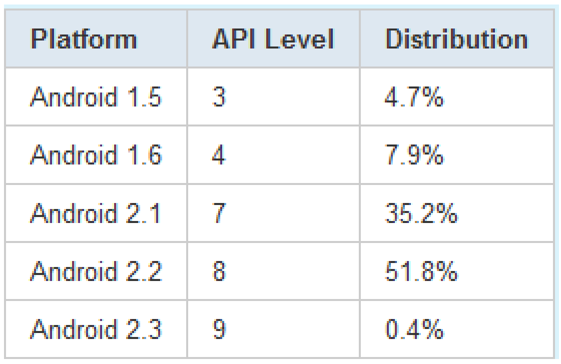
On the other hand the work has it's positive features. The fact that the voting was made so simple made running it on a limited device smooth and without problems. The installation was without problems and was quick. The user interface was very comfortable and straightforward. This enabled very easy initial setup 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 does what it needs and does it well.

3.3 Android analysis

The main problem concerning the Android system is its speed of development and different versions that are currently released. It was stated previously that Android is mostly forward compatible, but the as the version is higher the framework has more and more functionalities that make some of the programming unnecessary. The question here is the choice of the version on for which the application will be written. The distribution of versions are in the figure below.

As an application was already written for the J2ME platform the next logical choice is Android. The benefit of Android[5] 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. [2]

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.



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 3.1: Version Distribution of Android [3] from http://developer.android.com/resources/dashboard/platform-versions.html retrieved on 22.1.2011

3.4 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. The languages of choice are practically limitless but as, my predecessor I chose Java as the programming language to implement it. The reason for this is, that it would be pointless and very time consuming to implement it in a different language. Also Java SE is my preferred language. Next, when creating an application such as this 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[**?**] broadcast[**?**]. 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 info and registration web page, is solvable by implementing a simple web server into the present server software or by using a lightweight http server(Apache for example) in concurrency with the server software.

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.

3.5 Client

3.6 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. Rooted device

3.6.1 Physical theft

Physical theft poses a risk that the stored login information in the device. While 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 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.

3.6.2 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.

3.6.3 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. [4]

3.6.4 "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 beiing read and altered by the wron 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.

3.7 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

3.8 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.

3.8.1 Functional requirements

Server

1. The system will provide the user with a wifi-only mode and a fully mobile mode.
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]

Android

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.
9. The application must notify the user when a question has been finished.

3.8.2 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.

Android

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 adhere to the Android best design practices.
6. The application must provide an easy way to add a new language.
7. The application must store the user's login information in such a way that other applications will not be able to access it.

3.9 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

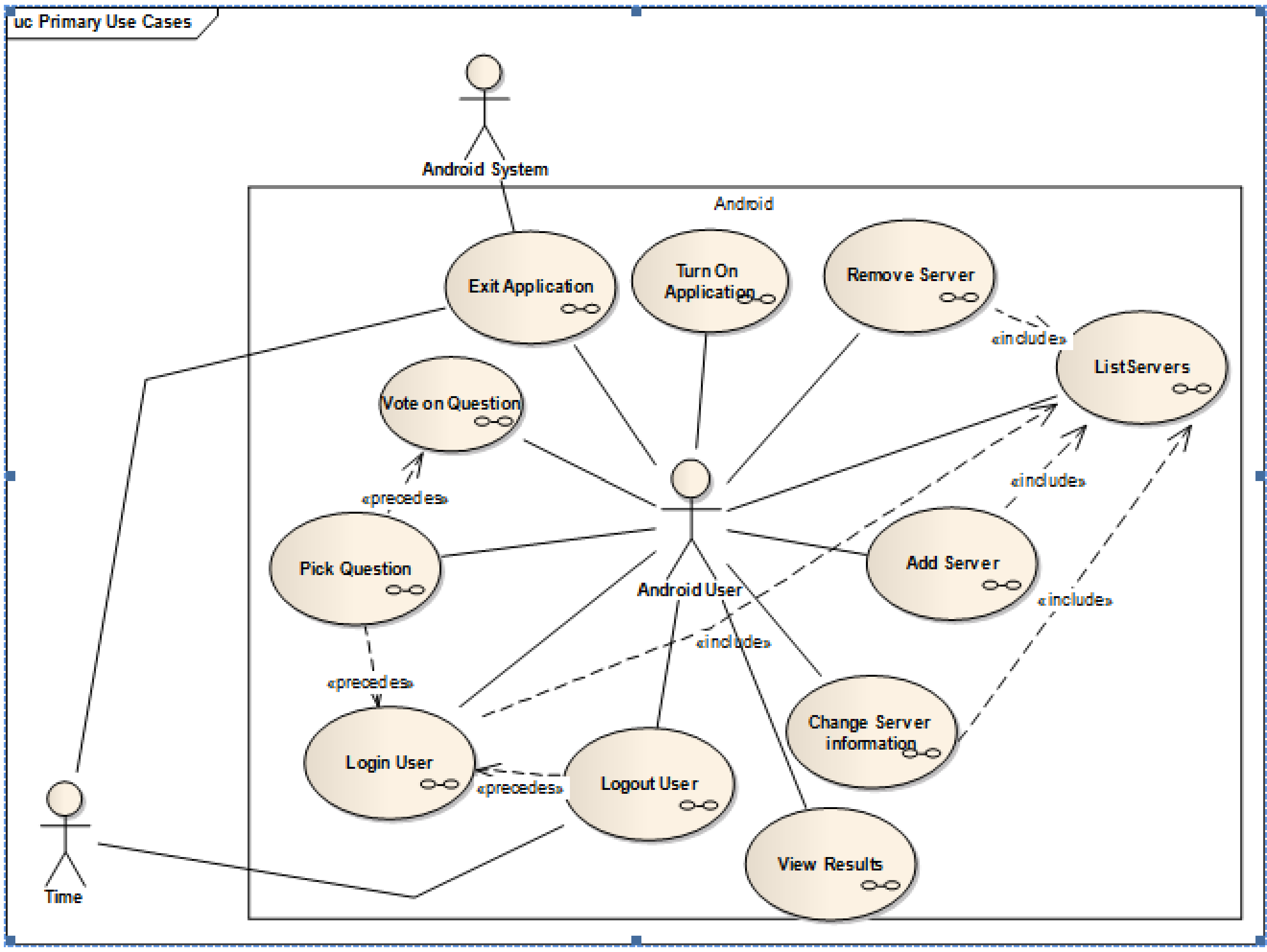


Figure 3.2: Use cases for the Android Application

3.10 Domain Model

This model describes the various relationships between objects in the problem domain. It is based on the Jakub Valenta's domain. This model describes how the client side on android sees the problem. The model is similar to that of the original, but takes to account that the use of this system will be purely for voting.

3.10.1 Server

The server domain model represents how the server sees the objects involved in voting. A full description is here [1] but as a quick reference the main purpose of the entities are briefly described.

Voting Entity The voting entity represents a vote event. It's attributes are isSecret used so signal if users vote secretly and minVotes - the minimum turnout of the event to be valid.

User The user entity represents a user within the system. It's attributes are self explanatory.

Vote This entity is the ballot. It is the form that voters fill out and send to the voting commission.

Question The question entity represents one question on the voting ticket. Its attributes are maxWinners - the number of winners, minPercent - the minimum percentage to be successful and text - an XML representation of the question and answers.

3.10.2 Android

From a clients perspective the domain model is a little different. It does not concern itself with the details about the voting event and does not participate in results calculation. On the other hand, the client has to deal with being connected to multiple servers and save users.

ServerData entity The server entity represents the a real server application running on a computer somewhere on the Internet.Its attributes are the settings that the client needs to create a connection to a server. It consists of an IP address or host name of the target server, the port through on which the server is listening. It also has a friendly name.
Next the server contains, the login data for one specific user. The password is encrypted. The user is an authorised user in the system that is created on the server.
The whole entity has a unique id that identifies it in the SQLite database.

QuestionData entity The questionData class serves as a container for the question text and the corresponding details and as a container for the possible answers. The states of the question here is quite similar to those on the server but the NOT_SET attribute is meaningless and therefore left out. The chosenAnswer attribute is the presently chosen answer that will be sent The other attributes are ANSWERABLE, meaning the question can be answered, and FINISHED. A finished question is open for result viewing.

Answer entity This represent the possible answer for the question that is stored on a server. Its attributes are percent and text. The former represents the percentage of people that voted for this question, this is only found when a question is in the "FINISHED" state, and and the latter represents the question text.

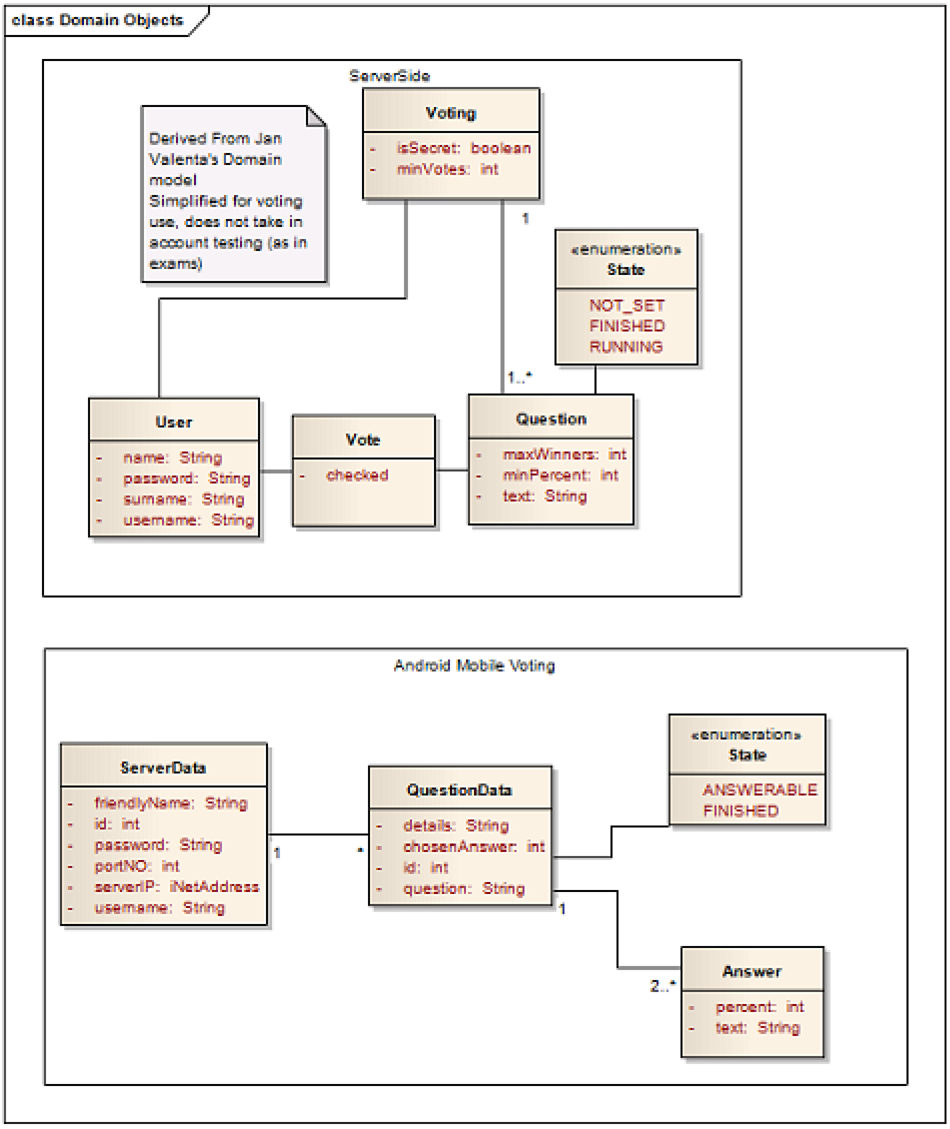


Figure 3.3: Domain Model of the System

Chapter 4

Design

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

4.1 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 from Android" message in the message body. The server then will either send a "426 Upgrade Required" code, which will indicate that the client has to try again using HTTPS, or will accept the connection without encryption. The message will have and optional port number, if the server is using a non standard port for secure connections. 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 LAN REQUIRED" code. Feature phones will still be able to connect through a Wi-Fi connection. Provided that the Wi-Fi is protected.

This differentiation 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 local 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 I perform this check 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.

4.1.1 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

- 426 Use HTTPS protocol
- 419 Local area connection required
- 466 Security was compromised by sending an unencrypted password.

4.2 Server side

4.2.1 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.

4.3 Android side

The android application will be designed with best practices for the platform in mind. It will be build using the MVC model[12].

4.3.1 Graphical User Interface (GUI) - View

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. Here, the application will behave in three different ways. First, it will normally display question details, and possible answers, second behaviour is to display the question results - as an ordered table when the question is closed and results published, third is to inform the user that he has already answered, and the results are not yet available.
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]

4.3.2 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[6] 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.

4.3.3 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 5

Implementation

Implementation in progress

Chapter 6

Testing

- Způsob, průběh a výsledky testování.
- Srovnání s existujícími řešeními, pokud jsou známy.

Chapter 7

Deployment

7.1 Overview

For a graphic view of the deployment model, please see the figure.

7.2 Server deployment

The server is the system part that "listens" to incoming connections.

7.3 Hardware

The server application does not have any specific haddware requirements and will run on any computer that is able to run the Java Virtual Machine.However, for a smooth voting proces 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.

7.3.1 Connectivity

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 takes no more than 5 minutes and this time is acceptable.
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.

7.3.2 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.

7.4 Client deployment

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

7.4.1 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

7.4.1.1 Android

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

7.4.1.2 Feature phone

Display and keyboard are necessary.

7.4.2 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 8

Conclusion

- Zhodnocení splnění cílů DP/BP a vlastního přínosu práce (při formulaci je třeba vzít v potaz zadání práce).
- Diskuse dalšího možného pokračování práce.

Bibliography

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

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

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

[4] web:developer.android.com. What is Android?, . `http://developer.android.com/guide/topics/data/data-storage.html#pref`, Retrieved on 26. 2. 2011.

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

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

[7] 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.

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

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

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

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

[12] web:msdn.microsoft.com. Model-View-Controller. http://msdn.microsoft.com/en-us/library/ff649643.aspx, Retrieved on 17. 3. 2011.

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

[14] 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.

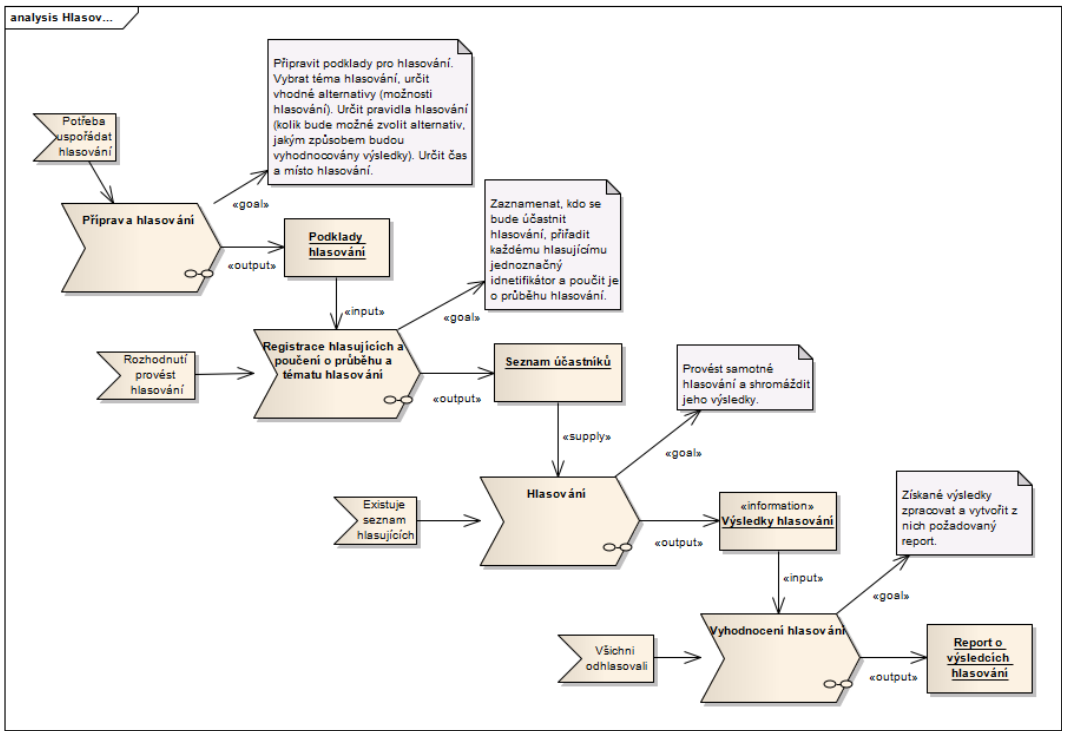
Appendix A

Abbreviations

⋮

Appendix B

UML diagrams


analysis Hlasov...
Potřeba uspořádat hlasování
Příprava hlasování
«goal»
«output»
Podklady hlasování
«input»
Rozhodnutí provést hlasování
Registrace hlasujících a poučení o průběhu a tématu hlasování
«goal»
«output»
Seznam účastníků
«supply»
Existuje seznam hlasujících
Hlasování
«goal»
«output»
«information» Výsledky hlasování
«input»
Všichni odhlasovali
Vyhodnocení hlasování
«goal»
«output»
Report o výsledcích hlasování

Připravit podklady pro hlasování. Vybrat téma hlasování, určit vhodné alternativy (možnosti hlasování). Určit pravidla hlasování (kolik bude možné zvolit alternativ, jakým způsobem budou vyhodnocovány výsledky). Určit čas a místo hlasování.

Zaznamenat, kdo se bude účastnit hlasování, přiřadit každému hlasujícímu jednoznačný idnetifikátor a poučit je o průběhu hlasování.

Provést samotné hlasování a shromáždit jeho výsledky.

Získané výsledky zpracovat a vytvořit z nich požadovaný report.

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

Appendix C

Installation and how to use

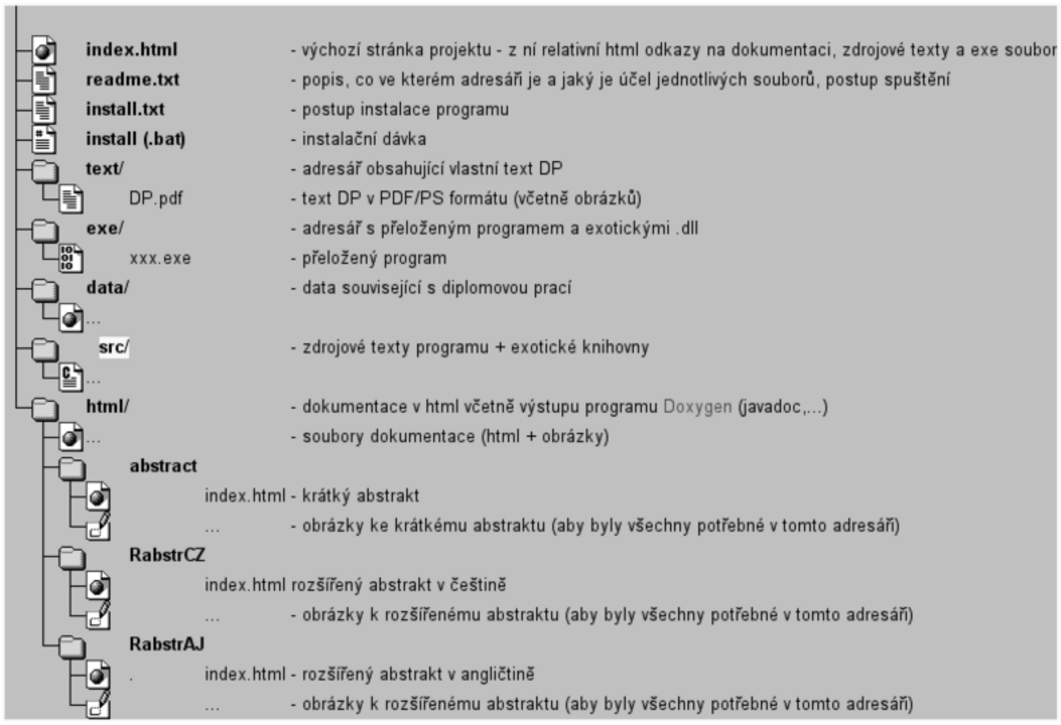
Tato příloha velmi žádoucí zejména u softwarových implementačních prací.

Appendix D

Enclosed CD Table of Contents

Tato příloha je povinná pro každou práci. Každá práce musí totiž obsahovat přiložené CD. Viz dále.

Může vypadat například takto. Váš seznam samozřejmě bude odpovídat typu vaší práce. (viz [?]):



```
index.html          - výchozí stránka projektu - z ní relativní html odkazy na dokumentaci, zdrojové texty a exe soubor
readme.txt          - popis, co ve kterém adresáři je a jaký je účel jednotlivých souborů, postup spuštění
install.txt         - postup instalace programu
install (.bat)      - instalační dávka
text/               - adresář obsahující vlastní text DP
    DP.pdf          - text DP v PDF/PS formátu (včetně obrázků)
exe/                - adresář s přeloženým programem a exotickými .dll
    xxx.exe         - přeložený program
data/               - data související s diplomovou prací
    ...
src/                - zdrojové texty programu + exotické knihovny
    ...
html/               - dokumentace v html včetně výstupu programu Doxygen (javadoc,...)
    ...             - soubory dokumentace (html + obrázky)
    abstract
            index.html - krátký abstrakt
            ...        - obrázky ke krátkému abstraktu (aby byly všechny potřebné v tomto adresáři)
    RabstrCZ
            index.html rozšířený abstrakt v češtině
            ...        - obrázky k rozšířenému abstraktu (aby byly všechny potřebné v tomto adresáři)
    RabstrAJ
    .       index.html - rozšířený abstrakt v angličtině
            ...        - obrázky k rozšířenému abstraktu (aby byly všechny potřebné v tomto adresáři)
```

Figure D.1: Seznam přiloženého CD — příklad

Na GNU/Linuxu si strukturu přiloženého CD můžete snadno vyrobit příkazem:
`$ tree . >tree.txt`
Ve vzniklém souboru pak stačí pouze doplnit komentáře.

Z **README.TXT** (případne index.html apod.) musí být rovněž zřejmé, jak programy instalovat, spouštět a jaké požadavky mají tyto programy na hardware.

Adresář **text** musí obsahovat soubor s vlastním textem práce v PDF nebo PS formátu, který bude později použit pro prezentaci diplomové práce na WWW.