Anton Magnusson

Jan Larsson

Implementation of TeliaSonera Incident Management Centre Site Attendance Application

Web Development
C-level thesis (15hp)

Date: 20130603
Supervisor: Kerstin Andersson
Examiner: Thijs J. Holleboom
Serial Number: C2013:07
Anton Magnusson
Jan Larsson

Implementation of TeliaSonera Incident Management Centre Site Attendance Application
Implementation of TeliaSonera Incident Management Centre Site Attendance Application

Anton Magnusson

Jan Larsson
This report is submitted in partial fulfilment of the requirements for the Bachelor’s degree in Computer Science. All material in this report which is not my own work has been identified and no material is included for which a degree has previously been conferred.

________________________________________
Anton Magnusson

________________________________________
Jan Larsson

Approved,

________________________________________
Advisor: Kerstin Andersson

________________________________________
Examiner: Thijs J. Holleboom
Abstract

TeliaSonera Incident Management Centre Karlstad has been situated in facilities placed in the bedrock of the Kroppkärr district of Karlstad for over ten years. From here TeliaSonera monitors parts of their different kind of networks all over the globe. One part is the monitoring of sites, a structure that holds equipment vital for a network to function.

Currently TeliaSonera Incident Management Centre receives about thirty phone calls a day from technicians from all over the world regarding the reporting of attendance on sites. The main objective is to create a mobile application that can substitute the need of a phone call.

The project resulted in the development of two web applications, one application with the purpose to replace the site attendance reporting by phone and another to monitor the incoming reports. The reason for developing a web application was to deliver a platform independent product under a short period of time. The site reporting application is resistant to attacks such as SQL-injections and Cross-Site Scripting.

Functionality of web applications is closing in on the functionality behind native applications. Features included in HTML5 allow the developer to take more liberty when programming web applications. These features are utilized in the search and filter functions of the monitoring application in this project.

The applications are written in HTML5, JavaScript and PHP and the data is stored in a MySQL database.
Contents

1 Introduction .................................................................................................................. 1

2 Background .................................................................................................................... 2
  2.1 Introduction ................................................................................................................. 2
      2.1.1 Requirements
      2.1.2 Scrum
      2.1.3 Structure of the design and implementation of the prototype
  2.2 Project discussion ...................................................................................................... 7
      2.2.1 Existing systems
      2.2.2 Prototype
      2.2.3 Security
  2.3 Summary .................................................................................................................... 13

3 Prototype design and implementation .......................................................................... 13
  3.1 Possible solutions ...................................................................................................... 14
      3.1.1 Native application
      3.1.2 Web application
  3.2 Choice for this project ............................................................................................... 16
      3.2.1 Sprint 1
      3.2.2 Sprint 2
      3.2.3 Sprint 3
      3.2.4 Sprint 4
      3.2.5 Sprint 5
      3.2.6 Sprint 6
  3.3 Summary .................................................................................................................... 46

4 Results .......................................................................................................................... 47
  4.1 Workflow ................................................................................................................... 47
  4.2 Prototype ................................................................................................................... 50
      4.2.1 The Web Application
      4.2.2 The Administrator Page
  4.3 Test Results .............................................................................................................. 54

5 Discussion ..................................................................................................................... 55
  5.1 Working with Scrum ................................................................................................. 55
  5.2 Prototype ................................................................................................................... 56
      5.2.1 Web application
      5.2.2 Administrator page
6 Conclusion .................................................................................................................. 57
  6.1 Project ................................................................................................................. 57
  6.2 Future Work ....................................................................................................... 58
  6.3 Summary ........................................................................................................... 58

7 References ............................................................................................................. 59

A Appendix – Site Attendance Application Questionnaire ......................................... 61
B Appendix – Administrator Page Questionnaire ........................................................ 62
C Appendix – Data model .......................................................................................... 64
List of Figures

Figure 1: The agile manifesto guidelines ................................................................. 4
Figure 2: Example of the Scrum-board model used in this project ......................... 6
Figure 3: Suggested layout of website - without filter ........................................... 9
Figure 4: Suggested layout of website - filtered ...................................................... 10
Figure 5: First version of the application GUI ....................................................... 20
Figure 6: First version of the administrator page ................................................... 20
Figure 7: Sprint 1 burndown-chart ........................................................................ 21
Figure 8: Application GUI Sprint 2 ........................................................................ 24
Figure 9: Administrator page Sprint 2 .................................................................... 25
Figure 10: Sprint 2 burndown-chart ...................................................................... 26
Figure 11: Application GUI Sprint 3 ....................................................................... 30
Figure 12: E/R diagram with the DepartmentPosts and Departments tables added .... 30
Figure 13: Administrator page displaying the filter function ................................... 31
Figure 14: E/R diagram with CommentPosts and AdminComments tables added .... 31
Figure 15: IMC-operator comments page Sprint 3 .................................................. 32
Figure 16: The Advance Search feature of the administrator page ....................... 33
Figure 17: Sprint 3 burndown-chart ...................................................................... 33
Figure 18: Administrator page Sprint 4 .................................................................. 36
Figure 19: IMC-operator comments page Sprint 4 together with the copy functionality. 37
Figure 20: Sprint 4 burndown-chart ...................................................................... 38
Figure 21: The Time Search feature displaying the jQuery widget “Datepicker” ...... 41
Figure 22: Administrator page Sprint 5 .................................................................. 41
Figure 23: Sprint 5 burndown-chart ...................................................................... 42
Figure 24: Application login web form ................................................................. 44
Figure 25: The database design Sprint 6 .............................................................. 45
Figure 26: Sprint 6 burndown-chart ...................................................................... 46
Figure 27: Project burndown-chart ................................................................. 50
Figure 28: Web application GUI................................................................. 51
Figure 29: Administrator page GUI ....................................................... 52
Figure 30: IMC-operator Comments GUI................................................. 53
Figure 31: The database in the final revision........................................... 54
List of tables

Table 1: Native Application Pros and Cons................................................................. 14
Table 2: Web application Pros and Cons................................................................. 15
Table 3: Project backlog......................................................................................... 16
Table 4: Project time plan...................................................................................... 18
Table 5: Sprint 1 backlog....................................................................................... 19
Table 6: Sprint 2 backlog....................................................................................... 23
Table 7: Sprint 3 backlog....................................................................................... 28
Table 8: Sprint 4 backlog....................................................................................... 35
Table 9: Sprint 5 backlog....................................................................................... 39
Table 10: Sprint 6 backlog...................................................................................... 43
Table 11: Final project backlog.............................................................................. 47
1 Introduction

Currently TeliaSonera Incident Management Centre (IMC) receives roughly thirty phone calls a day from technicians around the world reporting their attendance at sites TeliaSonera monitor. This means that personnel get tied up in phone calls that could have been avoided if the technicians could report their site attendance themselves. The reports that the technicians submit over the phone are entered into an information system manually by IMC personnel. In this case the human factor is a risk. The IMC operator could perceive or misinterpret what is heard over the phone. This in turn may lead to incorrect data in the database. It would be safer for the technician to report their information straight to the database, minimizing the human factor. TeliaSonera IMC wants the technicians to report their attendance via a mobile application that sends the information to a database that could be monitored by TeliaSonera IMC.

The objective is to create a mobile application that is suitable for all handheld devices and computers. This application needs to be more efficient than a phone call otherwise the technicians will not use it. The information that the technicians send via the mobile application must be monitored by TeliaSonera IMC, therefore a solution for presenting this data must be produced.

This project is bound only to create an application for sending information, an application for monitoring and the internal design of the database. The hosting environment and security validation will be handled by TeliaSonera.

Chapter 2 of this report introduces the reader to the company TeliaSonera, describes the problem given in this project more precisely and discusses the different aspects of the project solution. In this chapter readers are also introduced to Scrum and the project adaptations made by the developers.

Chapter 3 presents how a solution emerged sprint by sprint. In this chapter all sprints have a corresponding section where the planning, execution and demonstration phases are described thoroughly.

In Chapter 4 the reader is presented with the results of this project. It displays the workflow of this project, explains the functionalities and the usability of the end-product.

In this chapter the workflow of this project is displayed and the functionalities and the usability of the end-product are explained.
Chapter 5 discusses results that are presented in Chapter 4. Finally the report is tied together with the conclusions in Chapter 6. The questions “What have we learnt?”, “What can be achieved in the future?”, “What could we have done differently?” and “Did we succeed?” will be answered.

2 Background

This chapter aims to give the reader an understanding of the project by introducing the reader to TeliaSonera as a company and the problem given in this project, followed by a discussion covering different aspects of the project solution.

2.1 Introduction

At the end of 2011 TeliaSonera was Europe’s fifth largest telecom operator and had 27983 employees [1]. TeliaSonera IMC in Karlstad, Sweden, is responsible for monitoring and supervising TeliaSonera’s international network all year around, 24/7. The network stretches from Norway to Nepal and consists of services such as mobile, broadband, fixed voice, VoIP and TV. Services vary from country to country where TeliaSonera is present under different brands.

This project will concern and help the monitoring of TeliaSonera’s Swedish network in the first place and in the future might also be used to monitor the Finnish network, if the project turns out to be a success.

TeliaSonera hires several different service entrepreneurs such as Eltel and Relacom which provide them with technicians who repair and perform maintenance on TeliaSonera’s sites. Currently TeliaSonera IMC receives about 30 telephone calls each day, from technicians out in the field, concerning sites located in Sweden. The reason for them calling is usually to report site attendance (a site is a physical node in the network). This information is then entered into one of TeliaSonera’s information systems, the Drift Information System (DIS), where an Infra-DIS entry is created. The infra-DIS is created with the purpose to inform IMC, affected second level departments and customer support about which sites technicians are currently working on. When the technicians have finished their tasks they are supposed to call IMC again to report that they are finished and to verify that the site’s alarm has cleared. The attendance control should include more information than just presence. Information may include items such as the type of work, technician’s whereabouts (geographical location),
technicians name and phone number, if work has begun, reference number and technicians comments.

This project aims to help TeliaSonera simplify their staffs’ important task to monitor the network by reducing the amount of telephone calls coming from technicians. Together with TeliaSonera we have decided to create an easy-to-use mobile application for field technicians to report attendance and register information and status of current work, without calling IMC.

The introduction continues by presenting the requirements identified for the application that were decided during the first meeting with TeliaSonera IMC, followed by a description of the authors’ interpretation of Scrum and the way Scrum is applied to this project. Finally the project’s workflow structure is presented.

2.1.1 Requirements

During our first meeting with TeliaSonera IMC we identified the following requirements for the application together with the TeliaSonera employees Bo Hermansson, Senior Manager IMC, Christian Frisell, working with Change Management and involved in a similar existing system, Martin Hautamäki, Solution Manager, and Tommy Holm, former service technician and now IMC-operator.

- User-friendly
- Fast
- Connected to a database
- Platform independent

The application should be user-friendly because if it is too complex, the technicians will avoid using it and instead preferring a much easier phone call to IMC. For the application to be more efficient and quicker than a phone call it should take no more than 30 seconds to fill out the information. To accomplish those requirements the application could be implemented with a simple form, which could send the desired data to a database which is also connected to a webpage where IMC-operators could administrate the content of the database. Different entrepreneurs may have different cell phones and therefore the mobile platforms may vary. Therefore the application should be platform independent.

2.1.2 Scrum

Scrum is an agile software development technique where the software developers plan their work according to a backlog. The agile side of Scrum is implemented by following the guidelines specified in the agile manifesto (see Figure 1) [2].
A backlog is a list of tasks and goals (stories) that the developers and the product owner agree to achieve during the time span of the project. A product owner is a person who represents the client at the initial project planning, sprint planning and the demonstration after each sprint. During the project planning, the team of software developers and the project owner decide which stories to include in the project backlog and then prioritize them according to their importance. A project is divided into time periods called sprints. A span of two to four weeks is the typical duration of a sprint. Before each sprint the software development team and product owner have a meeting and decides upon which stories from the project backlog to include for the upcoming sprint in a sprint backlog. Each working day of the sprint begins with a “Daily Scrum Meeting” which is a meeting where all the developers join in and together finds out what was finished the day before and decide on what should be done this day. When the team has decided what stories to take on for the day, the
members will determine whether a story is too comprehensive and if it should be divided into smaller tasks. This allows the group to solve different parts of a story simultaneously and individually. The meeting should have a time limit (about 15 minutes) so that it will not lead to a discussion without results and therefore a waste of time. To keep the time limit it is recommended to have the meeting standing up. By using time estimates for each story the developers decide how many and which stories to include for the sprint. The time estimate unit varies between different software-development teams, the unit can be whatever gives the whole team a good idea about how long it takes to complete a story. To provide a sprint overview the developers is supposed to use a “Scrum-board” (see Figure 2). A “Scrum-board” consists of a table where each column shows which stories or tasks have been not-checked-out, checked-out or done. Each story from the sprint backlog has its own row in the table. By splitting up each story into several smaller tasks, the developers can divide the workload among themselves. A “burndown-chart” is also available on the “Scrum-board” which shows the work progression during the sprint. The x-axis of the “burndown-chart” specifies the total amount of estimated time of the sprint stories and the y-axis shows the developers total amount of time available. After each working day the developers makes a mark on the “burndown-chart” to show the progression.

To predict how many stories that can be completed each sprint is based on the developers calculated velocity from the previous sprint. The velocity is in this case calculated by dividing the number of completed man-days points with the amount of days in a sprint. If all the stories for the actual sprint are completed before the sprint ends the developers look into the project backlog and consider if there are some stories that could be added to the sprint backlog. In the end of each sprint the developers demonstrate the stories completed during the sprint for the product owner to get his or her approval and feedback, this is called a sprint demonstration. After the sprint demonstration the developers have a retrospective meeting where the group evaluates the finished sprint, what went well, what went wrong, what can be improved for the next sprint [3].
Figure 2: Example of the Scrum-board model used in this project.

The developing method for this project has a Scrum approach but with some modifications. The main motivation for this approach is that Scrum is flexible and allows continuous communication between the different actors involved in the project. This gives the project a higher level of consistency overall and tackles problems and solution features more easily as they appear.

One of the Scrum modifications done in this project is that the developers decide on the stories together with the product owner but grades them on their own.

The reason for not grading the stories together with the product owner is the strong consensus between product owner and developers. The great amount of freedom given to the programmers by the product owner, TeliaSonera, in the development of the product has also affected that choice.

The time estimate unit used for each story in this project is “man-days”. One “man-day” is a full working day for one group member, e.g. a group consisting of three members working a full day results in three “man-days”. If the time span is two days the group has six “man-days” and so on. “Man-days” gives a better general idea of how much time is needed to perform a specific story, than for example hours which could be too precise and leaves less room for mistakes and poor time estimates by the group.

Due to the fact that this project only has two developers a “Daily Scrum Meeting” never takes place in the same sense as it was described. On office arrival the developers discuss what should be done today and starts to work.
The parts of Scrum that have been embraced are the “Scrum-board”, “Burndown-chart”, Backlog, “Sprints” and Sprint Demonstrations.

2.1.3 Structure of the design and implementation of the prototype

TeliaSonera have requested the following structure of the design and implementation of the prototype:

1. Investigating different options, to establish the exact function specification of the application and in which environment the implementation should be performed.
2. Developing and implementing a mobile solution for field technicians.
3. Developing and implementing a website solution for displaying information.
4. Producing documentation and instructions for field technicians and IMC-operators.
5. Testing of the application functionality.
6. Testing the application with field technicians at TeliaSonera sites.
7. Deploying the application.
8. Finishing up and handing over the source code and documentation.

The steps 1-8 describes the development process cycle of the project. If there will be more than one solution to the project steps 1-7 will iterate for each solution, then step 8 will be performed at the end of the project to summarize and document the different solutions.

2.2 Project discussion

In this section existing systems will be presented followed by a discussion of how an application could be implemented and various security aspects of the application.

2.2.1 Existing systems

There are not any attendance control systems available for TeliaSonera field technicians. Therefore the technicians are forced to call TeliaSonera to report their presence. As was mentioned earlier in this chapter, IMC creates an Infra-DIS entry for each site where work is being performed. This is not an optimal solution due to the fact that the Infra-DIS entry has to be created manually by an IMC-operator. Avoiding having to reserve an IMC-operator for this type of call saves time and workforce.
There is currently a similar system, to what this project is supposed to achieve, in use for a different part of TeliaSonera called Change Management System (CMS). This system consists of a simple web form, that when filled out sends an email to report attendance to the affected departments.

The functions of the prototype that will be developed in this project have a goal to extend the gathering of information in the CMS system, mainly with information from service technicians.

TeliaSonera wants to avoid having to create a whole new system because this means that someone needs to be assigned as responsible IT Solutions Manager (ISM) for this system. Therefore TeliaSonera would like to explore the possibility to extend the functionality of their already existing system, CMS, which already has an ISM.

2.2.2 Prototype

From the requirements given by TeliaSonera the prototype has to be an easy-to-use mobile application that the technicians can use for reporting site attendance as well as a website allowing IMC-operators to monitor the information sent to the database by the technicians. IMC-operators should also be able to manually enter/modify information in the database by using the website. The purpose of the website is to provide IMC-operators with an overview of the work being performed on site by technicians.

Before the prototype is deployed as a live system the application will be developed in a test environment. The test environment will be provided on Atlantis servers by Karlstad University Computer Science department. At the start the environment will contain a web server and a database that will be used for deploying test code and mock up data.

2.2.2.1 User interface

The User interface is divided into two main components: Mobile Application and Website. Both have different terms, conditions and obstacles that need to be fulfilled. Each component will be discussed in the Mobile Application and Website section respectively.

Mobile Application

The mobile application has to be very user-friendly and it should not take more than 30 seconds for the field technician to fill out the data requested by TeliaSonera and send it in. This is a very important aspect of the application and its usefulness. If it becomes too complex and takes too long time to use, the technicians will stop using it and make a phone call instead. Each entrepreneur has different agreements concerning mobile phone technology.
Therefore it is important for TeliaSonera that the mobile application supports different kinds of mobile platforms such as iOS [4], Android [5] and Windows Phone [6].

The mobile application may consist of a form which is easy for the technicians to fill out and send to the database. The form should have input for the technicians contact details, type of work which is about to start/is finished, additional information, reference number from another ticket system and the geographical position of the technician.

To improve the simplicity of the application, and to save time for the technicians, the application should be able to recognize reoccurring data items such as name and telephone number. Common work types being sent to the database should be easy to select via a menu to save time.

Functionality for guiding the technicians from their current location to their destination site is a desired option but not an important priority. This might be implemented in a way that a map displays a drawn line from the technician’s current location to the site. Another way can be to display nearby sites and the technicians own location on a map in the application.

**Website**

The website should present the information sent by the technicians to the database. IMC-operators should easily be able to view and filter this information using only mouse clicks or by typing filter conditions into search bars. There should also be an option to manually enter information or modify existing content in the database. It is important that the presented information on the website is updated in real time. The main purpose of this webpage is to allow IMC-operators to see where work is in progress and if there are any technicians on site. This means that the website and database should be well implemented to keep track of the ongoing work.

*Figure 3: Suggested layout of website - without filter.*
Figure 4: Suggested layout of website - filtered.

Figure 3 shows an example of how one of the IMC-operators, which also has been working as a field technician, would like to have the information from the database displayed without filters.

Figure 4 shows how the same operator would like to display the information filtered (list filtered on the underlined item).

2.2.2.2 Database system

Because the application only needs to post data in the database and the website only needs to present it, the internal structure of the database does not have to be complex.

Glancing at how the IMC-operators want the data to be presented and what type of information TeliaSonera requests from the technicians, one simple table in the database should suffice to store all the information. Then the website could retrieve and filter data in the database by simple queries.

The CMS system has a database under development and this project might be able to share this database.

2.2.3 Security

In this subsection different security aspects, relevant to the project, will be discussed. Things like SQL (Structured Query Language)-injections and Denial of service (DOS) attacks will be taken into consideration. If they could in any way exploit the application [7].

Attackers use SQL-injections to strike at weaknesses of a webpage. By entering commands into input fields that are designed to send data to a database, attackers can breach the security of a webpage and allowing them to modify or display data. A common example of exploiting input fields is the SQL-injection into user-/password-fields [8].

<?php
$_POST['username'] = 'aidan';
$_POST['password'] = '' OR ''='' ;

$query = "SELECT * FROM users WHERE user='$_POST['username']' AND passw ord='$_POST['password']'";
mysql_query($query);
?

The browser will interpret and make the following query to the database:
SELECT * FROM users WHERE user='aidan' AND password=''' OR '''='''

This will allow the hacker to exploit the user account ‘aidan’ without knowing the actual password. An SQL-injection concerning usernames and passwords will not be a problem in the project due to the request from TeliaSonera that we construct a login-free mobile application. The reason for this is that TeliaSonera wants to avoid the trouble of assigning the responsibility for another database to a new administrator. A login-function will need a database containing user credentials for the different user accounts. Even though a login will not be used the webpage still needs protection from SQL-injections by mischievous users, all input types of the webpage must be protected against this sort of attacks.

The website controlled by IMC does not need a password because the site could be deployed on a web server within TeliaSonera’s intranet and therefore limiting the access to TeliaSonera employees connected to the intranet.

If the solutions to this project were to be implemented as a native¹ application for each platform the applications could be distributed, with a username and password hardcoded in the source code, to each user individually. This will not only give the application a higher level of security but also removing the need for a credential database.

Although it will not provide a guaranteed system security, e.g. a phone could be inherited, without being factory reset and therefore still have access to the application, by an unauthorized user.

Another solution is to implement a web based application for technicians to post data to the database. But as mentioned earlier in this section having a web application without username and password can cause heavily incorrect data to be posted in the database. The only solution that can secure this application is to protect all its inputs from SQL-injections by limiting the users set of characters and by using PHP Hypertext Pre-processor (PHP) escape strings. PHP is a widely-used, server-side, general-purpose scripting language especially suited for web development. This scripting language can be embedded into the Hypertext Markup Language (HTML) code [9]. In a worst-case scenario a limit on the number of packages sent to the

¹ Native application is an application that is developed for a particular platform.
webpage needs to be set to reduce the risk of having denial of service. The expression denial-of-service refers to attackers transmitting packages with inefficient queries to machines causing them to crash or slow down. The amount of packages needed to cause a crash depends on the targeted machine.

DOS and SQL-injections is what The Open Web Application Security Project (OWASP) refers to as kinds of Attacks.

“Attacks are the techniques that attackers use to exploit the vulnerabilities in applications. Attacks are often confused with vulnerabilities, so please try to be sure that the attack you are describing is something that an attacker would do, rather than a weakness in an application” [10].

Another kind of attack is Cross-Site Scripting (XSS) [11]. XSS attacks are often divided into two main types: Stored and Reflected XSS. Stored XSS Attacks means that the attacker uploads harmful scripts, for example through a comment input field, which is stored in a target database. These scripts persist over time until an administrator removes that comment/post from the database. If the users are allowed to enter any type of character or any combination of characters into an input field they can for example enter -

<script>alert(“ERROR”);</script>

If a website fetches this input from a database and presents it as HTML the web browser would execute the code and interpret it as an alert with the message “ERROR”. As long as the input remains in the database and is displayed on the website the alert will appear each time a user access the website. This is a rather harmless script in relation to other scripts that for example logs and passes on your user credentials.

A security measure for protection against Stored XSS is to check and remove any HTML script-tags from all user inputs. This will prevent attacks like the one described above from happening.

Reflected XSS is a non-persistent type of cross-scripting which the attacker uses to manipulate the web servers response to a user request. This can be achieved by the attacker by using links to trusted webpages that are vulnerable to Reflected XSS. When the user clicks the link a HTTP request, containing e.g. a malicious script, is sent from the web browser to the web server. The malicious script that was sent with the HTTP request then manipulates the HTTP response message from the web server, hence the name Reflected XSS. The script is often designed to steal user information.
2.3 Summary

This chapter begins with a short introduction of TeliaSonera as a company followed by a description of the problem this project aims to solve. The goal is to make the monitoring of service entrepreneur technicians work on- or off-site more efficient. Currently the monitoring is handled by phone communication between the technicians and IMC-operators. This solution is very time-consuming and inefficient for TeliaSonera’s staff. The purpose is to make this more efficient by letting technicians easily send information using their mobile devices.

The list below identifies the key goals for developing an application suiting TeliaSonera’s requirements:

- User-friendly application
- Containing a form that should take 30 seconds or less to fill out
- Technician should be able to send data from the application to a database
- Display database contents on an admin page
- IMC-operators should be able to add /modify information via the admin page
- Platform independent

The concept of using Scrum as a development technique is presented to give the reader an understanding of how the work is proceeding within this project. Different “artefacts” used in Scrum such as ”Scrum-board”, sprints, retrospectives and different actors are described.

TeliaSonera requested that the prototype should be implemented as a mobile solution. The prototype is discussed in the project discussion section of Chapter 2. The section reflects over how the user interface should be designed and how the database system should be implemented. Another aspect of the discussion was the security of the system. Whether or not a web based solution should be protected by a password.

But the discussion leaves one important question to be answered, “Should the technician’s user-interface be implemented as a native application on each platform? Or as a web based application for all platforms?”

3 Prototype design and implementation

The previous chapter covered the background and requirements for this project. This chapter will describe the possible solutions, present the chosen solution and how work progressed throughout the project.
3.1 Possible solutions

This section will evaluate and discuss the pros and cons of different solutions to this project.

3.1.1 Native application

A native application is an application which is developed for a specific platform. Each platform requires its own version of the application because of the different types of languages used on the different platforms. As mentioned in the mobile application segment of Chapter 2, Android, Apple iOS and Windows Phone are some of the more common platforms but there are also other platforms such as Blackberry [12] and the upcoming Firefox OS [13]. The Android platform uses Java, Apple iOS – Objective C and Windows Phone uses the .NET framework and the developer can choose freely between Visual C++ or Visual C# and Visual Basic [14]. At the end of January 2013 Google Android owned 52.3% of the Smartphone platform market shares while Apple iOS had 37.8% and Windows Phone had 3.1% [15].

Table 1: Native Application Pros and Cons.

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Works offline.</td>
<td>Costs.</td>
</tr>
<tr>
<td>Full device features access.</td>
<td>Distribution.</td>
</tr>
<tr>
<td></td>
<td>Application per platform.</td>
</tr>
</tbody>
</table>

One of the advantages of using a native application mentioned in Table 1 is that a native application is usable even if the device is offline. The application can queue queries while awaiting an Internet connection, when a connection has been made the application can execute pending queries towards the database. Another advantage of a native application is the ability to use the full feature of the device (e.g. camera, SMS, phone services).

Because of Apple being the second largest smartphone mobile platform it will be inevitable to develop an application that supports iOS. The disadvantage of developing applications for Apple iOS is that the use of iOS integrated development environment (IDE) Xcode costs money and it only runs on Apples native operating system.

A native application is installed locally on the device it will be running on. This means that the application will have to be distributed to all the devices that technicians are using in the line of work. An ordinary application can be distributed through Google Play, Apple iTunes and Windows Phone Store, but this application should not be available for the general public.
and can therefore not be distributed in ordinary manners. The application would need to be installed manually on each device.

The main disadvantage of developing a native application is the fact that the developers have to develop an application per platform. This gives the developers a great deal of concern because of TeliaSonera’s requirement of the application having to support iOS, Android and Windows Phone. With three platforms involved, this means there would be at least three application development processes. The estimated amount of time needed to develop an application for only one platform would stretch across the whole project timespan.

### 3.1.2 Web application

A web application is essentially an application that is used over Internet via an Internet browser. One of the benefits of having a web application instead of a native application is that you only have to implement one application that works on all the different platforms as long as they support the device’s browser.

**Table 2: Web application Pros and Cons.**

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>One application.</td>
<td>Security.</td>
</tr>
<tr>
<td>No need for an framework</td>
<td>Limited device feature access.</td>
</tr>
<tr>
<td>Free.</td>
<td>Internet based.</td>
</tr>
</tbody>
</table>

When developing a web application one of the pros Table 2 address is that there is no need for an framework because you use languages like HTML5, JavaScript, PHP and Asynchronous JavaScript and XML (AJAX). These types of languages are free to use and an application can be developed using a simple text editor such as Windows Notepad. Therefore you can develop an application for free and distribute it via your webpage.

HTML is a markup language which is used to create web pages and display them in web browsers. The first standardized version (2.0) of HTML was published 1995 as Internet Engineering Task Force (IETF) Request For Comments (RFC) 1866 [16]. HTML5 is the current version of the language but it is a working copy and the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG) aim to set it as the new standard. W3C and WHATWG decided upon a set of rules but one is especially important for this project, HTML5 should be device independent [17]. Even though
HTML5 has made the web more versatile the control of device features is still limited, for example making mobile devices send a text using the phone's native SMS function is impossible. This you can achieve when developing a native application.

JavaScript is used to insert programming code into HTML webpages [18]. AJAX uses JavaScript to exchange data with a server and to update parts of a web page without having to reload the whole page [19].

The security aspects for an online application were discussed in the security section of chapter 2.

The most significant drawback of having a web application is that your users depend on having an Internet connection. This should only be a minor setback in this case because if the technicians cannot fill out the form themselves, e.g. not having an Internet connection, they can call IMC and ask an IMC-operator to assist them.

### 3.2 Choice for this project

The choice for this project is a web application that fits all TeliaSonera’s needs and requirements. Due to the fact that you only need to develop one application for all the platforms saves time and effort which made the choice final. This section describes how the work was carried out, sprint by sprint, to reach a solution for this project.

In the first sprint the goal was to plan the whole project with a time plan and grade each item in the backlog with a story point (see Table 3 and Table 4).

#### Table 3: Project backlog.

<table>
<thead>
<tr>
<th>Story Point</th>
<th>Story</th>
<th>How to demonstrate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>310</strong></td>
<td>Planning</td>
<td>-</td>
<td>Plan the project (time plan and so on and so forth).</td>
</tr>
<tr>
<td><strong>300</strong></td>
<td>Web form GUI</td>
<td>Show appearance of the web form.</td>
<td>Textboxes/Dropdowns/Buttons/Design.</td>
</tr>
<tr>
<td><strong>290</strong></td>
<td>Test environment</td>
<td>Test server and test database on Atlantis.cse.kau.se</td>
<td>Setup a development environment.</td>
</tr>
<tr>
<td><strong>280</strong></td>
<td>Connect form to test database</td>
<td>Send form content to database. Show where it is stored.</td>
<td>Connect to the test environment database.</td>
</tr>
<tr>
<td><strong>260</strong></td>
<td>Web server</td>
<td>Show the application and administrator page at the new domain in a web browser.</td>
<td>Export the application and administrator page from Atlantis to the same domain as CMS.</td>
</tr>
<tr>
<td>17</td>
<td>Geolocation</td>
<td>Show coordinates sent to the database.</td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>Let users send their coordinates.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>Save metadata.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>230</td>
<td>TeliaSonera DB Display the exported database in MySQL.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>Login Try to penetrate the application without login, then login.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>155</td>
<td>Multilingual Have multilingual support on the application.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>Nearby sites Display nearby sites coordinates on a map in the web application.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>Navigation Display the path from the current location to destination.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>Presenting the data Show all data available in the database.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>Geolocation 2 Convert GPS coordinates and show the RT90 representation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>Add posts from Fill a form, similar to the admin page, and send the information to the database.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>Filter data Filter data by clicking a cell within a post and order the posts by the values within the column.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>Sort data. Sort data by an attribute and display and order the posts by the attribute.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>Filter data by User should be able to click a cell within a post and order the posts by the cell value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>Filter data Sort data available in the database.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>Convert GPS coordinates and show the RT90 representation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>Presenting the data in the database. Show all data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>Nearby sites Display nearby sites coordinates on a map in the web application.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>Navigation Display the path from the current location to destination.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>Presenting the data Show all data available in the database.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>Geolocation 2 Convert GPS coordinates and show the RT90 representation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>Add posts from Fill a form, similar to the admin page, and send the information to the database.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>Filter data Filter data by clicking a cell within a post and order the posts by the values within the column.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>Sort data. Sort data by an attribute and display and order the posts by the attribute.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
20  SMS  Send a short message from the administrator page to a cell phone.  Allow IMC-operators to send texts from the administrator page to the technician’s phone using the phone number given by the technician in the report.

Table 4: Project time plan.

<table>
<thead>
<tr>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>January</td>
<td>February</td>
<td>March</td>
<td>April</td>
<td>May</td>
<td>June</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Progressreport</td>
<td>#1</td>
<td>#2</td>
<td>#3</td>
<td>#4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experiment design.</td>
<td>Experiment.</td>
<td>Evaluation.</td>
<td>Get ready for opposition.</td>
<td>Final draft</td>
<td>Opposition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Literature study.</td>
<td>Write report.</td>
<td>Write and hand in a draft of the report.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ch. 2</td>
<td>Ch. 3, 4</td>
<td>Ch. 4, 5</td>
<td>Ch. 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sprint 7 and Sprint 8 are marked with the colour red in the time plan because they are preliminary sprints that will be carried out if there is time. The weeks that are marked with the colour red contains a public holiday.

3.2.1  Sprint 1

This is the first sprint and the start of the project. Below the planning, execution and demonstration of Sprint 1 is described.

3.2.1.1  Planning

The planning started with selecting stories for the sprint backlog from the project backlog. In addition to producing a web form Graphical User Interface (GUI), to get feedback early in the project on how the product owner wants the application to look and that it contains all input that meets the requirements of the application, the project should also have a test environment up and running. The web form application should after this sprint be able to connect and send information to the database and the web form should also be able to retrieve the geographical location of the user after his or her approval (see Table 5).

The thought behind getting feedback on the GUI early in the project was to ensure that the application is developed in a way that fits the goals of having a user-friendly application with a form that is quick and easy to fill-out.
Table 5: Sprint 1 backlog.

<table>
<thead>
<tr>
<th>Story Point</th>
<th>Story</th>
<th>How to demonstrate</th>
<th>Description</th>
<th>Man-days</th>
</tr>
</thead>
<tbody>
<tr>
<td>310</td>
<td>Planning</td>
<td>-</td>
<td>Plan the project (timeplan and so on and so forth).</td>
<td>1</td>
</tr>
<tr>
<td>300</td>
<td>Web form GUI</td>
<td>Show the appearance of the web form.</td>
<td>Textboxes/Dropdowns/Buttons/Design.</td>
<td>2</td>
</tr>
<tr>
<td>290</td>
<td>Test environment</td>
<td>Test server and test database on Atlantis.cse.kau.se</td>
<td>Setup a development environment.</td>
<td>2</td>
</tr>
<tr>
<td>280</td>
<td>Connect form to test database</td>
<td>Send form content to database. Show where it is stored.</td>
<td>Connect to the test environment database.</td>
<td>2</td>
</tr>
<tr>
<td>250</td>
<td>Geolocation</td>
<td>Show coordinates sent to the database.</td>
<td>Let users send their coordinates.</td>
<td>2</td>
</tr>
</tbody>
</table>

3.2.1.2 Execution

From the first meeting with TeliaSonera an initial backlog was created and the stories graded according to their importance.

To get the project started a test environment was needed. So a request for acquiring a database and a web server on atlantis.cse.kau.se was made to the domain administrator at Karlstad University, Mohammad Rajiullah, who provided an account with the phpMyAdmin database tool and web server that supported PHP [20].

The foundation of the web application consists of the index page and the corresponding style sheets and this is where the development begun. The index page consists of HTML code that creates textboxes for name, phone number, site and reference; dropdown menus for reference type and work category; text area for comments; and a custom made button for submitting the data. All these inputs create a web form application (see Figure 5).
Figure 5: First version of the application GUI.

The form is filled out by a user and transmitted to the database by clicking the Confirm button. The connection to the database is created by using predefined PHP functions such as `mysql_connect()` to connect, `mysql_close()` to disconnect and data is inserted in the database tables by making MySQL queries via PHP.

On the administration side of the application a simple web interface for viewing the data in the database was created in the same way as the web form, but instead of sending data to database the administrator page is retrieving it (see Figure 6).

Figure 6: First version of the administrator page.
This story was not one of the stories in the Sprint 1 backlog but the group thought of it as a tool to test the database connection and enables them to display the data within the database faster. Not only avoiding having to sign into phpMyAdmin this administrator page also make it easier for the product owners to get a better view of the data that is being sent to the database at the sprint demonstration.

![Figure 7: Sprint 1 burndown-chart.](image)

The work progression of Sprint 1 is displayed in Figure 7. As you can see it still remained two man-days at the last day of the sprint which means that not all of the stories were completed during this sprint. The group had underestimated the time required for implementing the “Geolocation” story and was not able to demonstrate the story due to malfunction in the transmission of the coordinates to the database and not having the time to fix it. This was left in the backlog to Sprint 2.

3.2.1.3 Sprint Demonstration

The different stories were demonstrated for the product owners according to the Sprint 1 backlog and the group received the following feedback.

- Restricted number of digits entered in the reference number box
- Change the work categories
- SMS-service
- English and Swedish version of the application
- TeliaSonera logotype
- Save meta-data
The reference number box, see Figure 5, next to reference type dropdown menu should have a restricted number of entered digits because the different reference types have different lengths of their reference numbers. The option “others” should be added to the reference types as well. Some of the work categories were misleading and were reduced in Sprint 2 to four options; Start planned work, Update IMC on critical/emergency matters, Job finished/Error fixed/Leaving site and other information to IMC (e.g. GPS-coordinates) from the six previous ones; Start planned work, Update IMC after troubleshooting, Update IMC on alarm, Send my position to IMC, Risky work on site, Work that could cause alarm and others.

The product owners also requested an SMS-service. When a technician wants to report that he/she is finished an OK response from IMC is needed for them to leave the site. The SMS-service should then be used to send this response to the technician via a SMS.

Because of the English skills may vary between the users and the possibility of this application being launched in Finland, the product owners requested multilingual support (English/Swedish) for the application.

The application should also be able to save input data from the user until the next time he/she uses the application so the user does not need to fill out the form twice and, making it more efficient.

3.2.2 Sprint 2

This is the second sprint for the project. Below the planning, execution and demonstration of Sprint 2 is described.

3.2.2.1 Planning

The calculated velocity of Sprint 1 was 1.8 points/day. Sprint 2 is three days long and therefore the team should be able to complete at least 5.4 points.

The stories selected at Sprint 2 planning meeting are listed in Table 6. The “Geolocation” story that was not finished in the previous sprint was added to the Sprint 2 backlog but with a lower time estimate (1 man-day instead of 2) because most of the work was done in the first sprint. The story “Presenting the data in the database” was completed in Sprint 1 but is added as a formality to show that it was finished and rated with 0 man-days as time estimate. An important requirement for the application is that it should not take too much time to fill out information; therefore “save metadata” story was a crucial function of the form. Saving the metadata information such as name and phone number, which is usually static data per technician and web browser, will allow the technicians’ chosen web browser to remember the
user entries in between the sessions. Improvements and additions to layout of the web form were requested at the demonstration of Sprint 1. This was a lot of minor fixes, such as adding the company’s logo to the web application and implement error handling of the user input that was already planned in the project backlog.

Table 6: Sprint 2 backlog.

<table>
<thead>
<tr>
<th>Story Point</th>
<th>Story</th>
<th>How to demonstrate</th>
<th>Description</th>
<th>Man-days</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>250</strong></td>
<td>Geolocation.</td>
<td>Show coordinates in the database or on a map.</td>
<td>Let users send coordinates to the database (Backlog from Sprint 1).</td>
<td><strong>1(2)</strong></td>
</tr>
<tr>
<td><strong>240</strong></td>
<td>Save metadata.</td>
<td>Turn on/off web-browser and see that the information is saved.</td>
<td>Save name/phone etc. So the users don’t need to enter it for each session.</td>
<td>3</td>
</tr>
<tr>
<td><strong>160</strong></td>
<td>Improve the layout of the web form.</td>
<td>Show the new layout.</td>
<td>Simplify and improve the web form. E.g. add a TeliaSonera logo.</td>
<td>2</td>
</tr>
<tr>
<td><strong>105</strong></td>
<td>Presenting the data in the database.</td>
<td>Show all data.</td>
<td>Show all data available in the database.</td>
<td>0</td>
</tr>
</tbody>
</table>

3.2.2.2 Execution

On the first working day of Sprint 2 the stories “Save metadata”, “Geolocation” and “Improve the layout of the web form” was checked out on the Scrum-board.

The “Save metadata” story was implemented with the help of functionality introduced in HTML5. In HTML5 the programmer can choose to save information with the help of sessionStorage or localStorage objects. These two objects allow the web browser to save data entries for a specific web page. The difference between sessionStorage and localStorage is that with the sessionStorage object data is only saved session wise and with localStorage object the data is saved locally on the device and will remain there indefinitely if it is not manually removed. For this application the localStorage object was used to save metadata with the motivation that the technicians may turn off their devices from time to time and then the data that was saved in sessionStorage would be lost. An example of what the application could look like when accessing it in another session when metadata has been saved in a localStorage object is displayed in Figure 8.
While one of the developers was implementing the function of saving metadata, the other was analysing the code that was written in Sprint 1 for the “Geolocation” story to find out what was causing the malfunction of the application. The malfunction was that the http POST containing the coordinates was empty when trying to submit to the database. Geolocation script calls within the form caused the fields, containing the coordinates, to clear before the information was submitted to the database. This error was fixed by adding two hidden textboxes, one for longitude and another for latitude, to the form and instead of calling the script inside the form it was called onLoad (when the webpage is loaded into the browser). Changes were made to the Geolocation script so it filled out the hidden textboxes. The longitude and latitude is only sent to the database if the user checks the “Use my location” checkbox.

Some of the improvements that were made in the “Improve the layout of the web form” story were that the TeliaSonera logo and a favicon (an icon in the web browser that is usually showed in the address bar next to the URL) were added to the web sites (see Figure 9). The TeliaSonera logo was downloaded from the TeliaSonera homepage and a favicon was generated from the logo.
Another feature of this story was to check the user input before sending it to the database. Controls were created for all the fields in the web application that confirms that all the inputs are entered correctly. MySQL real escape string is used to make sure that the substring from the user input is safe to put in a database query. This function evaluates the input substring and removes any characters that could change the original query causing errors and corruption in the database.

In this sprint the problem was not to underestimate the man-days of each story; instead the stories man-days were overestimated. The overestimation gave the team time to spare to implement other stories from the project backlog, stories such as “Multilingual”, “Geolocation Admin” and some of the functionality in “Filter data”. This also shows how man-days could provide room for adjusting the amount of work described in Chapter 2.

Figure 10 displays the work progress in Sprint 2. As you can see in the figure all the stories were completed but the line representing actual progress drops below the ideal progress of the sprint early on in sprint. This shows that the workload could have been increased.
The multilingual support of the administrator page was implemented by adding subdirectories with a translated copy of the index.html for each language (Swedish, English and Finish).

The “Geolocation Admin” story was implemented by using Google Maps API. A hyperlink that is connected to the map-button on the administrator page is generated by coordinates associated to the post. By clicking a map-button a new webpage is opened that parses the URL to get the coordinates and then displays them on a map using the Google Maps API.

At the end of the sprint there was still some time to spare so the basics of a search function were developed. The search function steps through an array each time the search string changes. This was implemented using an AJAX script that contains a listener that calls a search function when a key is released. The future functionality of the search function will be to allow the user of the administration page to search and filter posts from technicians. AJAX allows this function to give instant feedback from the database on events such as key released.

3.2.2.3 Sprint Demonstration

The different stories were demonstrated for the product owners according to the Sprint 2 backlog and the group received the following feedback and requests.

Mobile Application

- Add site attendance to the list of categories
- Demand mandatory comment when reference is missing
- Textboxes should be empty when using the application for the first time

Figure 10: Sprint 2 burndown-chart.

The multilingual support of the administrator page was implemented by adding subdirectories with a translated copy of the index.html for each language (Swedish, English and Finish).

The “Geolocation Admin” story was implemented by using Google Maps API. A hyperlink that is connected to the map-button on the administrator page is generated by coordinates associated to the post. By clicking a map-button a new webpage is opened that parses the URL to get the coordinates and then displays them on a map using the Google Maps API.

At the end of the sprint there was still some time to spare so the basics of a search function were developed. The search function steps through an array each time the search string changes. This was implemented using an AJAX script that contains a listener that calls a search function when a key is released. The future functionality of the search function will be to allow the user of the administration page to search and filter posts from technicians. AJAX allows this function to give instant feedback from the database on events such as key released.

3.2.2.3 Sprint Demonstration

The different stories were demonstrated for the product owners according to the Sprint 2 backlog and the group received the following feedback and requests.

Mobile Application

- Add site attendance to the list of categories
- Demand mandatory comment when reference is missing
- Textboxes should be empty when using the application for the first time

Figure 10: Sprint 2 burndown-chart.

The multilingual support of the administrator page was implemented by adding subdirectories with a translated copy of the index.html for each language (Swedish, English and Finish).

The “Geolocation Admin” story was implemented by using Google Maps API. A hyperlink that is connected to the map-button on the administrator page is generated by coordinates associated to the post. By clicking a map-button a new webpage is opened that parses the URL to get the coordinates and then displays them on a map using the Google Maps API.

At the end of the sprint there was still some time to spare so the basics of a search function were developed. The search function steps through an array each time the search string changes. This was implemented using an AJAX script that contains a listener that calls a search function when a key is released. The future functionality of the search function will be to allow the user of the administration page to search and filter posts from technicians. AJAX allows this function to give instant feedback from the database on events such as key released.

3.2.2.3 Sprint Demonstration

The different stories were demonstrated for the product owners according to the Sprint 2 backlog and the group received the following feedback and requests.

Mobile Application

- Add site attendance to the list of categories
- Demand mandatory comment when reference is missing
- Textboxes should be empty when using the application for the first time

Figure 10: Sprint 2 burndown-chart.

The multilingual support of the administrator page was implemented by adding subdirectories with a translated copy of the index.html for each language (Swedish, English and Finish).

The “Geolocation Admin” story was implemented by using Google Maps API. A hyperlink that is connected to the map-button on the administrator page is generated by coordinates associated to the post. By clicking a map-button a new webpage is opened that parses the URL to get the coordinates and then displays them on a map using the Google Maps API.

At the end of the sprint there was still some time to spare so the basics of a search function were developed. The search function steps through an array each time the search string changes. This was implemented using an AJAX script that contains a listener that calls a search function when a key is released. The future functionality of the search function will be to allow the user of the administration page to search and filter posts from technicians. AJAX allows this function to give instant feedback from the database on events such as key released.

3.2.2.3 Sprint Demonstration

The different stories were demonstrated for the product owners according to the Sprint 2 backlog and the group received the following feedback and requests.

Mobile Application

- Add site attendance to the list of categories
- Demand mandatory comment when reference is missing
- Textboxes should be empty when using the application for the first time

Figure 10: Sprint 2 burndown-chart.

The multilingual support of the administrator page was implemented by adding subdirectories with a translated copy of the index.html for each language (Swedish, English and Finish).

The “Geolocation Admin” story was implemented by using Google Maps API. A hyperlink that is connected to the map-button on the administrator page is generated by coordinates associated to the post. By clicking a map-button a new webpage is opened that parses the URL to get the coordinates and then displays them on a map using the Google Maps API.

At the end of the sprint there was still some time to spare so the basics of a search function were developed. The search function steps through an array each time the search string changes. This was implemented using an AJAX script that contains a listener that calls a search function when a key is released. The future functionality of the search function will be to allow the user of the administration page to search and filter posts from technicians. AJAX allows this function to give instant feedback from the database on events such as key released.

3.2.2.3 Sprint Demonstration

The different stories were demonstrated for the product owners according to the Sprint 2 backlog and the group received the following feedback and requests.

Mobile Application

- Add site attendance to the list of categories
- Demand mandatory comment when reference is missing
- Textboxes should be empty when using the application for the first time
Product owners Bo Hermansson, Martin Hautamäki, Tommy Holm and Jonas, a former Relacom technician, participated at the demonstration and they all had different opinions on the web application and the admin page. One of the topics that came up was the user input control. If the technician selects the reference missing option and do not comment the IMC-operators might not have a clue of why the work was carried out. When the user select the reference missing option a comment from the technician is required.

Website

- Tab that displays active posts (i.e. jobs that have not been reported as finished)
- Tab that displays acknowledged post and IMC-operator comments
- Posts that reports that job has been finished should be marked red
- The admin page should automatically refresh periodically
- Categorization of each post
- SMS-service, mark that a text has been sent to technician
- Searching posts by clicking different cells

During Sprint 2 the group received the information that the administrator page might not be the one that is going to be used. Apparently the CMS group will develop a common administrator page for both the CMS application and this web application. But the product owners and the developers agreed to continue the work on the administrator page in this project. If IMC can design an administrator page according to their liking the CMS group can use it in the development of a new administrator page. Some of the requests of features on the admin page that came up during the meeting were to create one tab that displays all the current jobs that have not been reported finished and another that displays comments associated to different posts. Each post should also have the option to be categorized according to department that the work concerns. To make the monitoring easier the operator should be able to search the database by clicking the cells of the table presenting the posts. When clicking a cell containing for example a reference number a search of the database would be done and returning a new table containing only posts related to that reference number. Another improvement of the admin page, suggested at the demonstration, was to allow the IMC-operators to be able to acknowledge posts with the category “Start planned work” and display that information so co-workers are aware of it.

3.2.3 Sprint 3

This is the third sprint of the project. Below the planning, execution and demonstration of Sprint 3 is described.
3.2.3.1 Planning

The calculated velocity of Sprint 2 was 2 points/day. Sprint 3 is four days long and therefore the team should be able to complete about 8 points.

The stories selected at Sprint 3 planning meeting is listed in Table 7. After Sprint 2 demonstration the development team received feedback on the appearance of both the web form and the administration page. Some of the feedback regarded crucial changes and additions to the web form and administrator page. This meant that the team had to change the specification of the already completed stories “Improve the layout of the web form” and “Presenting the data in the database” from Sprint 2 which led to include new stories with slightly different specifications in Sprint 3; “Improve the web form” and “Improve the presentation of data”. When a story is demonstrated and does not fulfil the product owners expectations the story cannot be considered done and will be put as backlog stories for the next sprint. The reason for not putting the Sprint 2 stories as backlog for Sprint 3 is that the specification for the stories from Sprint 2 was fulfilled but new additions to the web form and the presentation of data was requested.

As mentioned in the section about the demonstration of Sprint 2 a filter function was very desirable. The “Filter data” story consist mainly of two types of filtering functions; “filter-by-click” and advanced search filter. These functions will be described in the execution of Sprint 3.

The “Sort data” story is similar to the filter story, letting the user sort posts by the values within the column – chronologically and alphabetically. The “Acknowledge posts” story intends to implement an acknowledgement function on the administrator page, allowing IMC-operators to mark post as being acknowledged. The reason for this was described at the end of the section 3.2.2.3.

Table 7: Sprint 3 backlog.

<table>
<thead>
<tr>
<th>Story Point</th>
<th>Story</th>
<th>How to demonstrate</th>
<th>Description</th>
<th>Man-days</th>
</tr>
</thead>
<tbody>
<tr>
<td>160</td>
<td>Improve the web form</td>
<td>Show the new appearance.</td>
<td>New requests for improvement of the web form (e.g. fix layout issues for smaller screens).</td>
<td>1</td>
</tr>
<tr>
<td>105</td>
<td>Improve the presentation of data.</td>
<td>Show the new improvements by displaying the data.</td>
<td>New requests for improvements of the web form (e.g. IMC comments on posts, categorize posts)</td>
<td>3</td>
</tr>
<tr>
<td>75</td>
<td>Filter data</td>
<td>Filter data by clicking a cell within</td>
<td>User should be able to click a cell within a post</td>
<td>2</td>
</tr>
</tbody>
</table>
a post. and order the posts by the cell value.

<table>
<thead>
<tr>
<th>74</th>
<th>Sort data.</th>
<th>Sort data by an attribute and display result.</th>
<th>Users should be able to click for example a column header and order the posts by the values within the column.</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>Acknowledge posts</td>
<td>Acknowledge post by for example checking a checkbox.</td>
<td>Users acknowledge posts so everyone knows that a post has been taken care of.</td>
</tr>
</tbody>
</table>

3.2.3.2 Execution

Sprint 3 started off with checking out the “Improve the web form” story and the “Improve the presentation of data” story. The “Improve web form” story was broken down in smaller tasks; “Fit smaller screens better”, “Mandatory comment when the reference type is ‘Reference missing’ ” and “JavaScript validation of input”.

There had been complaints of how the web application fitted the screen. The application did not adjust well to smaller screens (less than 4 inches). This was fixed by adjustments of the meta-tag inside the header from having a fixed size to a user defined instead and replacing the two-option On-site radio button group with a single checkbox. These changes made the appearance more compact and made the application fit the smaller screens better (see Figure 11).

Another improvement of the application was to make comments mandatory when reference is missing. This was an addition to the validation of the form before submitting it to the database. A JavaScript function was written and is called upon the submission of the form, the function then checks if the user input is valid. If validation returns true the form will be submitted to the database otherwise the browser will prompt error messages in popup boxes.
The “Improve the presentation of data” story concerning the Administrator Page was broken down into smaller stories such as “Assign posts to affected department”, “IMC-operator comments”, “Mark ‘Work Finished’ posts” and “Auto refresh”. The assigning of posts was solved visually by creating another column (“Affected Department”) in the display of posts containing select boxes (see Figure 13), allowing the IMC–operators to assign a post to the department it concerns. Logically this was implemented by adding two tables in the database, one containing the different departments and the other containing relations between posts and departments (see Figure 12). The reason for creating new tables was to minimize the amount of columns in the main table and to keep the database well structured.

**Figure 11: Application GUI Sprint 3.**

**Figure 12: E/R diagram with the DepartmentPosts and Departments tables added.**
Figure 13: Administrator page displaying the filter function.

Another request from the staff that came up during the demonstration of Sprint 2 was to allow the IMC-operators to make comments and notes about posts the technicians make. This was implemented in a similar way as the “affected department” by having one table dedicated for IMC-operator comments and another table for relations between posts and IMC-operator comments (see Figure 14).

To access the comments from IMC-operators a hyperlink is assigned to each post directing the user to a new URL which displays information about the post and the comments made regarding that post (see Figure 15).
To make the administrator page monitor posts closer to real-time an auto refreshing function of the administrator index page was added using a simple html-tag. The page is now refreshed every other minute.

Tommy requested that a post should turn red if it is reported within the category “Work finished/Fault Corrected/Leaving site” to make it more eye-catching on the screen. A post marked with a distinct colour should be easier to spot and will therefore increase the chances for IMC-operators to respond to the technicians more quickly.

The “Filter data” story intends to allow the users to filter posts by clicking the different cells (see Figure 13). By clicking a cell a JavaScript function using AJAX is called and makes queries towards the database to fetch the posts containing the value within the cell that was clicked. The matching posts are then displayed on the admin page. When the base function of the filtering was finished the developers noticed the possibility of using it as the base for an advanced search feature as well. After making some adjustments to the filter function an advanced search function was ready to use. The users can search by entering different combinations of column attributes and values which will return matching posts (see Figure 16).
The last story of Sprint 3 was to enable to the administrator page users to mark alarm as acknowledged. This was implemented by adding another attribute to the main table of the database indicating whether the post is acknowledged or not. The operator marks a post as acknowledged by checking the “Ack”-checkbox of the post (as seen in Figure 16).

Unfortunately the team did not meet the goals for this sprint because of an under estimation of the “Filter data”. The “Filter data” took one extra man-day, than what was expected, to complete and therefore there was no time for making a solution for sorting data (see Figure 17).

Figure 16: The Advance Search feature of the administrator page.

Figure 17: Sprint 3 burndown-chart
3.2.3.3 Sprint Demonstration

All the stories from the Sprint 3 backlog were demonstrated, except “Sort data” which was not completed, and the feedback was positive overall. Bo Hermansson and Tommy Holm participated in the demonstration of Sprint 3 and their request for future additions to the administrator page is listed below.

- Possibility for IMC-operator to quickly assist technician to report work finished.
- Allow IMC-operators to edit existing posts.
- Colour scheme for post categories.
- Search for active jobs at a certain point of time.
- Link from the application to instructions of how to use the application.

Tommy considered the possibility for IMC-operators to copy the content of an existing post into a new form and be able to repost it with a new category by a single click of a button. This could be done if technicians forget or are unable to report work finished in the application themselves.

IMC-operators should have the possibility to edit information in existing posts for example in the case of a technician having posted the wrong reference number.

One of the minor fixes of this sprint was to mark a post that is categorized as “Work finished/Fault corrected/Leaving site” with the colour red to make it stand out. The product owners changed their mind when seeing the result at the sprint demonstration and would like to have different colours that depend on if the alarm has been reported finished and acknowledged. All the posts should be red at start and if the post has been reported as work finished it should turn yellow and when the post has received acknowledgement it should turn to another colour, green.

Another change will be to let the assignment of affected department represent the acknowledgement of the post instead of a separate checkbox. Since when assigning a department to the post the post has been notified and will be taken care of.

When the stories suggested above have been finished, a feature which will search for posts that have on-going work at a given point in time, will be implemented. This would facilitate troubleshooting of related issues.

Except for these stories mentioned there were some minor additional fixes requested such as links to administrator page index and fix the colouring in Internet Explorer (IE).

The filter functions was such a success and filled all the operators needs so a sorting of the data might be excessive.
3.2.4 Sprint 4

This is the fourth sprint of the project. Below the planning, execution and demonstration of Sprint 4 is described.

3.2.4.1 Planning

The calculated velocity of Sprint 3 was 1.6 points/day. Sprint 4 is four days long and therefore the team should be able to complete 6.4 points.

The stories selected at the planning meeting of the fourth sprint are listed in Table 8. At the start of Sprint 4 issues with the administrator page adapting to the Internet Explorer (IE) browser was discovered. For example one of the issues was the green time-column background in IMC-operator comments was displayed as black in IE. This was addressed by adding it as a Story to the Sprint 4 backlog.

As mentioned in the demonstration of Sprint 4, Section 3.2.3.3, Tommy Holm and the other product owners requested an addition to the project backlog. This is what the story “Add posts from admin page” is about. IMC-operators should have the possibility to reuse existing posts for either submitting new information to the database or altering already existing posts.

The “Internet Explorer adaption and other fixes” story is about changing some of the code and add fixes to make the administrator site function as it is supposed to in Internet Explorer as well.

Table 8: Sprint 4 backlog.

<table>
<thead>
<tr>
<th>Story Point</th>
<th>Story</th>
<th>How to demonstrate</th>
<th>Description</th>
<th>Man-days</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>Internet Explorer adaption and other fixes.</td>
<td>Display the webpage from Internet Explorer.</td>
<td>Adapt some functionality in the code to work well with Internet Explorer. Various fixes in the code as well.</td>
<td>5</td>
</tr>
<tr>
<td>80</td>
<td>Add posts from admin page.</td>
<td>Send a filled out form from admin page.</td>
<td>Create a form, similar to the one in the application, and send the information to the database. The form should be filled out by clicking a post in the admin page and use the existing information.</td>
<td>2</td>
</tr>
</tbody>
</table>
3.2.4.2 Execution

The sprint started by checking out the two stories specified in Table 8 with focus on making the application work correctly in IE versions previous to IE 9 which is used by the most IMC-operators. The navigation bar that is displayed at the bottom of Figure 18 was created by adjusting and creating classes in the CSS. The position of the navigation bar is fixed which means that it is statically fixed to the bottom of the browser window at all times. This worked nicely in Firefox and Chrome but was experiencing issues in IE. The issues were that the navigation bar had a high level of transparency making it invisible and also put at bottom of the page. Since older versions of IE do not support CSS features like position: a fixed document type declaration has to be specified at the very first line of code on the index page.

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

The navigation bar was still transparent so a Direct X fix needed to be added to the CSS properties of the navigation bar to make it visible [21]:

```css
filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr='#A02182',endColorstr='#A02182');
```

Figure 18: Administrator page Sprint 4.

When the two stories of this sprints backlog was finished there were still man-days left to work with. One of the functionalities that were developed on the spare time was the “Active jobs”-tab. This tab displays all jobs that have not been reported as finished which will help TeliaSonera IMC to get a good overview of which sites has on-going work or for example
technicians on site (see Figure 18). By knowing which sites are active TeliaSonera IMC can identify the cause of alarms being triggered in other systems.

Figure 19: IMC-operator comments page Sprint 4 together with the copy functionality.

The “Add posts from admin page” story was implemented as a link from the IMC-operator comments page that sends the already existing information to a popup window. The popup displays a form similar to the one used by the field technicians but instead it is already filled out with the information from the associated post (see Figure 19). With the form filled out the users have the opportunity to alter the information and either submitting the form as a new post to the database or save the information in the existing post. If the user choses to create a new post the information is inserted to the database. If the user on the other hand choses to change the existing post the post is updated by making an update-query to the database containing the new values. This is helpful in the way that the IMC-operators can change the information of a post if the technician has for example submitted values with typographical errors or entered the wrong reference number.

Two more extra stories were taken on during Sprint 4 when there was still time left of the sprint after the “Active jobs” story was completed. The first was to fetch the table of posts in smaller chunks making the page load faster by not needing to retrieve the entire table of posts in the database all at once. And the other was to save application form input when the technician reports anything except “Work finished”, making it easier for the technician’s to
fill out the form once more when trying create another post regarding the same job. Often the content of the posts is almost the same when it regards the same job.

3.2.4.3 Sprint Demonstration

This sprint all the stories in the sprint backlog were completed in time for the demonstration, in fact all the planned stories were completed before the end of the sprint (see Figure 20). The remaining time of the sprint was used to implement some additional features such as displaying active jobs, form input profiles and splitting up table of posts into smaller chunks.

![Sprint 4 burndown-chart.](image)

*Figure 20: Sprint 4 burndown-chart.*

All of the stories from the Sprint 4 backlog (including the extra stories) were demonstrated at this demonstration. Product owners Bo Hermansson and Tommy Holm attended the demonstration. The participants was satisfied with what was completed this sprint but requested some small changes regarding the active jobs feature. The list of active jobs should not only contain jobs that have not been reported as work finished but also jobs that have been reported as “work finished” but has not been assigned to a specific department. To make the post that is not acknowledged easier to distinguish from the list of jobs - a similar color scheme to the one that was discussed at the demonstration of Sprint 3 (see Section 3.2.3.3) was decided to be implemented in the upcoming sprint.
3.2.5 **Sprint 5**

This is the fifth sprint of the project. Below the planning, execution and demonstration of Sprint 5 is described.

### 3.2.5.1 Planning

The calculated velocity of Sprint 4 was 1.75 points/day. Sprint 5 is five days long and therefore the team should be able to complete 8.75 points.

The stories selected at the planning meeting of the fifth sprint are listed in Table 9. As mentioned in Section 3.2.3.3 (the demonstration of Sprint 3) a feature allowing the users to search for active jobs within a time span based on a given time and time interval was discussed. This discussion led to the story “Time Search”.

The “Color scheme”-story was also requested at the demonstration of Sprint 3. Depending on the status of the post it should be marked accordingly (see Description of stories in Table 9).

Product owner Tommy Holm requested hyperlinking from posts with NMCP and PWIC references during the planning of this sprint. This story was added to the sprint backlog as “Ticket link”.

At the moment the web site is refreshed every other minute. This means if the user has entered a search query or a filter and the web site suddenly refreshes the query or filter will be reset. The developers suggested that improvements of how the table content is updated should be made. This was added to the sprint backlog as “Improved table update”.

*Table 9: Sprint 5 backlog.*

<table>
<thead>
<tr>
<th>Story Point</th>
<th>Story</th>
<th>How to demonstrate</th>
<th>Description</th>
<th>Man-days</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>68</strong></td>
<td>Time search</td>
<td>Use the time search function to search posts at a given point in time. Display the posts with in the given time.</td>
<td>Users should be able to use a search function to look for posts within a time span. The input is from which time to search and the search interval (in hours).</td>
<td><strong>2</strong></td>
</tr>
<tr>
<td><strong>60</strong></td>
<td>Improved table update</td>
<td>Submit a form to the database and at the same time display the admin page. Notice the new post appearing in the table without refreshing the page.</td>
<td>Table content gets dynamically refreshed without refreshing the whole page.</td>
<td><strong>2</strong></td>
</tr>
</tbody>
</table>
3.2.5.2 Execution

On the first day of Sprint 5 the stories “Time Search” and “Ticket link” were checked-out on the Scrum-board.

The “Time Search” has the parameters date, hour and minute of the day and an interval expressed in hours. The input field belonging to the date parameter uses a jQuery datepicker widget for capturing user input. The widget displays a calendar helping the user to pick a date for the search (see Figure 21). Both the input fields for date and time are masked to make it easier for the user to enter the input correctly. For masking the input the Masked Input plugin written by Josh Bush is used [22]. Masked Input 1.3.1 is licensed under the MIT license and helps the programmer to define what the input is supposed to look like [23]. For example the date input requires the format YYYY-MM-DD, the Masked Input plugin masks this input to “- - “ (4 blank space, hyphen, 2 blank space, hyphen, 2 blank space) each of the spaces representing a digit. A function then uses these parameters to make a query towards the database asking for posts on that timespan, the time span is defined as follows: from the given point of time, x, to x + interval.
The product owners wanted to implement a more sophisticated color scheme, one that clearly distinguishes ordinary posts’ from, posts with the category work finished. At first the developers thought of a color scheme suitable for color blind users, but a traffic light model of the color scheme was preferred as color blindness was not an issue at the time. In the future a color blind mode could be implemented if it becomes too big of a problem. The color red of the traffic light model represents a post that has the category “Work finished/Fault corrected/Leaving site” but has not received an acknowledgement. The color green represents a post that has the category “Work finished/Fault corrected/Leaving site” and has received an acknowledgement. The color yellow represents all other posts (see Figure 22).
The “Improved Table Update” story was implemented using jQuery functions that makes queries towards the database every five seconds and dynamically reconstructs a HTML division (div), a division can be used to divide the code into different sections, for each tab. The administrator page constructs a different query for each tab that reconstructs the divs accordingly. Until the tab is clicked the content of the tab is hidden. When a tab is clicked by a user it displays the already loaded information for that tab within the corresponding div.

3.2.5.3 Sprint demonstration

All the stories from Sprint 5 were demonstrated. Product owners Bo Hermansson and Tommy Holm participated in the Sprint demonstration. After the demonstration they were satisfied with what had been accomplished in execution of Sprint 5. The security aspects of the application were brought to attention once more by the developers since they thought that leaving the application open for anyone to use it poses a great threat. The developers and the product owners agreed on trying to implement a single login authorisation service for the users of the mobile application.

An SMS feature, as discussed at previous sprint demonstrations, is still a highly desirable function. The developers were granted access to a server and database hosted by TeliaSonera this sprint. Therefore the main focus of the upcoming sprint will be to move the application from the hosting on Atlantis to the new location, secondly adding a login feature to the web application and if there is time left an SMS feature will be implemented.

![Sprint 5 burndown-chart](image)

*Figure 23: Sprint 5 burndown-chart*

The “Time Search”-story and the “Ticket link”-story were heavily overestimated and only took one man-day in total which is represented by a steep decline in the Sprint 5 burndown-
chart between the dates 2013-03-20 and 2013-03-21 (see Figure 23). From 2013-03-22 and forward the two lines, Ideal and Actual, becomes each other’s tangent which indicates a good estimation of the rest of the stories.

3.2.6 Sprint 6

This is the sixth and last sprint of the project. Below the planning, execution and demonstration of Sprint 6 is described.

3.2.6.1 Planning

In this final sprint of the project the goal was to export the application and administrator page to a TeliaSonera domain. The calculated velocity of Sprint 5 was 1.4 points/day. Sprint 6 is five days long and therefore the team should be able to complete about 7 points. The security aspects were discussed once more at the demonstration of Sprint 5 and therefore the “Login”-story was added to the Sprint 6 backlog as well as the Project backlog. An SMS service where IMC-operators have the possibility to contact technicians via a text has remained as a highly desirable functionality of the administrator site. To complete this story the developers has to place an order for using TeliaSonera’s internal SMS service. Therefore the story “SMS” was added to the backlog for further investigations about who to contact and where to place the order (see Table 10).

Table 10: Sprint 6 backlog.

<table>
<thead>
<tr>
<th>Story Point</th>
<th>Story</th>
<th>How to demonstrate</th>
<th>Description</th>
<th>Man-days</th>
</tr>
</thead>
<tbody>
<tr>
<td>260</td>
<td>Web server</td>
<td>Show the application and administrator page at the new domain in a web browser.</td>
<td>Export the application and administrator page from Atlantis to the same domain as CMS.</td>
<td>1</td>
</tr>
<tr>
<td>230</td>
<td>TeliaSonera DB</td>
<td>Display the exported database in MySQL. Demonstrate the application sending data to the new DB and show it on the administrator page.</td>
<td>Export the database from Atlantis to a database provided by CMS.</td>
<td>2</td>
</tr>
<tr>
<td>200</td>
<td>Login</td>
<td>Try to penetrate the application without login, then login.</td>
<td>Simple login function for users of the application.</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>SMS</td>
<td>Send a short message from the</td>
<td>Allow IMC-operators to send texts from the</td>
<td>2</td>
</tr>
</tbody>
</table>
3.2.6.2 Execution

Sprint 6 started off by checking out the “Login”-story on the Scrum-board while awaiting directions from the CMS group about the export to the host at TeliaSonera’s domain. The implementation was finished in only one man-day - one less than expected. The login has a quite simple session-based implementation but it is secured from SQL-injections and XSS attacks. The session requires only a single login which means that the session is open until it timeout or the browser is closed (see Figure 24).

![Login Form](image)

*Figure 24: Application login web form.*

After having implemented a login protection for the application the team had a telephone conference with Christian Frisell, Göran Sporre and Svante Victorsson who have configured the hosting environment at TeliaSonera’s CMS group. Topics that were discussed concerned the separation of the application and administrator page, whether to use HTTP or HTTPS for communication between the server outside the firewall and the database inside, the current design of the application, security aspects and how to connect to the new environment.

The export stories of this sprint were more time consuming than first expected. A big setback was the differences in the configurations of the test environment on Atlantis compared to the actual environment provided by TeliaSonera. Differences such as the test environment allowing short tags (explanation: `<? ?>` instead of `<?php ?>`) and the actual environment only prompting server down or unavailable when there are errors in the code with no detailed debugging information.
Before exporting the database the naming of the tables were changed to keep the database in line with database conventions and to be more descriptive (see Figure 25).

**Figure 25: The database design Sprint 6.**

Furthermore the new hosting web server did not support Latin1 encoding, therefore the web application and administrator page encoding as well as the database collation had to be changed from Latin1 to UTF-8. This caused features to malfunction which in turn led to several changes codewise, in order to make the administrator page and the application to work correctly.

This was the first stage of the export of the application. All the servers are at the moment placed inside TeliaSonera’s firewall and therefore only reachable by TeliaSonera employees. The web application will be migrated to a server outside TeliaSonera’s firewall and communicate with the database inside.

An SMS service cannot be implemented by ourselves therefore other alternatives must be investigated. With TeliaSonera being, as mentioned when introducing the background of this project, one of the largest telecom companies in Europe this can be solved internally by using existing systems. Bo Hermansson suggested contacting Lars-Eric Larsson who might know more about this type of services and how to get permission to use them.

### 3.2.6.3 Sprint Demonstration

This sprint all stories except the SMS story was finished in time for the demonstration. The developers contacted Lars-Eric Larsson who responded that he will look in to this matter and get back to them. Therefore the developers could not complete this story and two man-days remained of the planning (see Figure 26).
This was the last sprint of the Prototype design and implementation phase of the project the developing team will now transcend to the testing phase. Due to this transition of the project the demonstration of Sprint 6 was an opportunity to discuss how the testing should be carried out. The developers would like to test the usability of both the application and the administrator page. Product owner Bo Hermansson will contact a service entrepreneur about having a meeting regarding pilot testing of the application, assuming that the server hosting the web application outside of TeliaSonera’s firewall is up and running.

For measuring the usability questionnaires A and B will be used.

Two different questionnaires will be used to measure the usability. The questionnaire regarding the web application will focus on how long it takes to fill out the form (see Appendix A (Swedish)). The administration page questionnaire is more task oriented, letting the users find information in the database by using the different functionalities (see Appendix B (Swedish)).

To make the tests feasible for the users; manuals for both the application and administrator page will be written (confidential, therefore they are not included in the appendices).

3.3 Summary

This chapter began with a presentation of possible solutions for this project where it stood between a native application and a web application. The choice for this project was to implement the prototype as a web application due to the nature of this project (see the requirements in Section 2.3). Essentially the goal was to supply the technicians with a form to

Figure 26: Sprint 6 burndown-chart.
fill out on their handheld devices which could be sent to a database monitored by TeliaSonera IMC. The drawback of having to implement several different native applications for multiple operating systems was too big and supposedly time consuming.

During this project the work was carried out by using the developers take on Scrum. The sprints of this project are represented in the subsections of Section 3.2 (Choice for this project), where each subsection presents the planning, execution and demonstration of each sprint. The planning section of a sprint describes the stories included in the sprint backlog. Further a description of what the goals are, how to demonstrate the stories and the time estimate of each story are given. The Execution section of the sprints describes how the stories were implemented. This section shows best how the appearance of the application changes from sprint to sprint. The Sprint Demonstration section shows the burndown-chart of the sprint and documents the feedback and requests for changes received from the product owners during the demonstration.

4 Results

This chapter presents the results of the experiment phase of this project that is described in Chapter 3 (Prototype design and implementation).

4.1 Workflow

The section shows how work progressed throughout the project (see Figure 27) and the stories included in the final backlog of this project (see Table 11).

<table>
<thead>
<tr>
<th>Story Point</th>
<th>Story</th>
<th>How to demonstrate</th>
<th>Description</th>
<th>Man-days</th>
</tr>
</thead>
<tbody>
<tr>
<td>310</td>
<td>Planning</td>
<td>-</td>
<td>Plan the project (time plan and so on and so forth).</td>
<td>1</td>
</tr>
<tr>
<td>300</td>
<td>Web form GUI</td>
<td>Show the appearance of the web form.</td>
<td>Textboxes/Dropdowns/Buttons/Design.</td>
<td>2</td>
</tr>
<tr>
<td>290</td>
<td>Test environment</td>
<td>Test server and test database on Atlantis.cse.kau.se</td>
<td>Setup a development environment.</td>
<td>2</td>
</tr>
<tr>
<td>280</td>
<td>Connect form to test database</td>
<td>Send form content to database. Show where it is stored.</td>
<td>Connect to the test environment database.</td>
<td>2</td>
</tr>
<tr>
<td>260</td>
<td>Web server</td>
<td>Show the application</td>
<td>Export the application</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>and administrator page at the new domain in a web browser. and administrator page from Atlantis to the same domain as CMS.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>Internet Explorer adaption and other fixes. Display the webpage from Internet Explorer. Adapt some functionality in the code to work well with Internet Explorer. Various fixes in the code as well.</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>Geolocation Show coordinates sent to the database. Let users send their coordinates.</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>Save metadata. Turn on/off web-browser and see that the information is saved. Save name/phone etc. So the users don’t need to enter it for each session.</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>230</td>
<td>TeliaSonera DB Display the exported database in MySQL. Demonstrate the application sending data to the new DB and show it on the administrator page. Export the database from Atlantis to a database provided by CMS.</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>Login Try to penetrate the application without login, then login. Simple login function for users of the application.</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>Improve the web form Show the new appearance. New requests for improvement of the web form (e.g. fix layout issues for smaller screens).</td>
<td>2+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>155</td>
<td>Multilingual Have multilingual support on the application (English, Swedish and Finnish). Show the translated application.</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>Presenting the data in the database. Show all data. Show all data available in the database.</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>Add posts from admin page. Send a filled out form from admin page. Create a form, similar to the one in the application, and send the information to the database. The form should be filled out by clicking a post in the admin page and use the existing information.</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>Filter data Filter data by clicking a cell within a post. User should be able to click a cell within a post and order the posts by</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Requirement</td>
<td>Description</td>
<td>Difficulty</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-------------</td>
<td>-------------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>Sort data.</td>
<td>Sort data by an attribute and display result. Users should be able to click for example a column header and order the posts by the values within the column.</td>
<td>1+1</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>Acknowledge posts</td>
<td>Acknowledge post by for example checking a checkbox. Users acknowledge posts so everyone knows that a post has been taken care of.</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>Time search</td>
<td>Use the time search function to search posts at a given point in time. Display the posts with in the given interval. Users should be able to use a search function to look for posts within a time span. The input is from which time to search and the search interval (in hours).</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>Improved table update</td>
<td>Submit a form to the database and at the same time display the admin page. Notice the new post appearing in the table without refreshing the page. Table content gets dynamically refreshed without refreshing the whole page.</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>Ticket link</td>
<td>Open the IMC operator comments of a post with a NMCP or PWIC reference. Click the reference number and redirect a new tab to an external ticket system with the information regarding that reference number. Redirection to external ticket system.</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>Color scheme</td>
<td>Show different posts with different status which affect the color of the post. A new post should be marked with the color blue. If the post is reported as work finished it should be marked with the color red. If the post is also has been acknowledged it should be marked with the color green.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>SMS</td>
<td>Send a short message from the administrator page to a cell phone. Allow IMC-operators to send texts from the administrator page to the technician’s phone using the phone number given</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
The project backlog changed over time as progressed - stories were added and removed from the original backlog (see Table 3).

![Project burndown-chart](image)

*Figure 27: Project burndown-chart.*

The project burndown-chart shows how the actual progress almost is a tangent to the ideal line which indicates a good time estimates overall. The goal of a project was to make the ideal progress and the actual progress intersect at the end. Unfortunately the “SMS” story remained unfinished which prevented the two lines from intersecting.

### 4.2 Prototype

This section will summarize what the work described in Chapter 3 produced. All the time and effort added up to two main components – the web application and the administrator page.

#### 4.2.1 The Web Application

This application consists of a form which is used to transmit information from the technicians working at a site to the database. It is especially fitted, visually, for mobile devices since the application is meant to be easy to use for the technicians when working in the field, far away from computers but close to an internet connection.

The final version of the application handles input from five input fields, two select boxes and two checkboxes (see Figure 28).
Figure 28: Web application GUI.

The input holds the category of work, the technician’s contact information, if he or she is on site, reference to other ticket systems, technician’s geographical position and comments about the work that is carried out.

When the information is filled out and the user clicks the submit-button a validation of the form is performed. This validation controls if the user input is entered correctly before posting. The data within the form tags is sent to the class databasehandler by using the Hypertext Transfer Protocol request method POST [24]. The class databasehandler is written in PHP and controls the connection between the application and the database. To send information to the database the databasehandler first needs to establish a connection to the database, receive the user input and then perform an insertion query towards the database containing the user input.

Before constructing the query all user input is sanitized from any special character that could harm the database and stripped from any tags to prevent XSS. If the insertion is a success the application will prompt a success message to confirm the user’s request.

To make it easier for the users all user input is saved in the browser web storage which prevents the need to fill out the form with the exact information twice.

The web application is protected with a login to secure the use of the application from outsiders.
4.2.2 The Administrator Page

The administrator page helps TeliaSonera IMC to monitor the work that is reported in the web application. This section covers the monitoring, search and database aspect of the final version of the administrator page.

4.2.2.1 Monitoring

The website is split up in tabs (Start, Active jobs, Denmark, IP, No department, Site, Transmission, TSIC, Voice, search tabs and a filter tab) (see Figure 29).

![Figure 29: Administrator page GUI.](image)

The start tab displays all posts in the database.

The department tabs display the posts that have been assigned to a specific department (Denmark, IP, Site, Transmission, TSIC and Voice) including the posts that have not yet been assigned to any department. IMC-operators assign posts to departments themselves by choosing which department the post belongs to in a select box.

The active jobs tab displays a list of all the sites where work has not been finished or not been acknowledged by IMC-operators.

The content of all the tabs changes dynamically along with the data coming in from the technicians, the web site redraws the displayed table each time a new post is made.

All of the posts are following the same traffic light-color scheme – green if the work is finished and acknowledged by IMC, red if the work is finished but not acknowledged and the rest is yellow. This makes the posts regarding work that has been completed stand out from the rest of the posts.

An IMC-operator also has the option to make comments about a post by clicking the “IMC-operator comments” link next to the post. This view presents the post in its entirety and
an input field for the comments (see Figure 30). When the comments have been made on the post a table of comments is displayed.

![Figure 30: IMC-operator Comments GUI.](image)

Via the IMC-operator Comments view the operators are able to change the content of a post or reuse the content to make a new post. This feature can be very valuable at the end of the day when a technician most likely has forgotten to report a job as finished. Then the IMC-operator can use the old information and make a new post regarding the same job and report it as finished.

4.2.2.2 Search functions

There are two different kinds of search functions – advanced search and time search. In the advanced search the user can specify name, phone, comment, category, site, on site and reference. To narrow the results the user can specify multiple conditions (e.g. Name: Anton, Site: KS1). This search function shows only results that matches the intersection of all conditions. The function reacts on each key up event making the website perform the search simultaneously to the user typing on the keyboard.

In the time search the user specifies a date, time of day and a time interval which the function uses to search for all post on that date during the specified time interval.

The cells of the posts are clickable. When clicking a cell the information inside the cell is used to filter the posts in the database, these posts are displayed under the filter tab.

When a user uses clicks for filtering cell information the resulting table is changed dynamically in the same manner as the different tabs automatically displaying new posts sent from the technicians to the database. Whilst the advanced search and time search tables show a snapshot of the database from when the search was made.
4.2.2.3 Detailed database description

In the final revision the database is a relational database with a UTF-8 collation (see Figure 31). The primary keys are marked with underline and foreign keys are marked with a plus in the summary of the database below the figure (see Appendix C for detailed information about the data model).

![Database Diagram](image)

*Figure 31: The database in the final revision.*

- **WorkReport:** `Id`, `Time`, `OnSite`, `Site`, `Coordinate`, `Reference`, `Name`, `Phone`, `Comment`, `Category`
- **Department:** `Id`, `Name`
- **Comment:** `Id`, `Message`, `Time`
- **AssignDepartment:** `Department_Id⁺`, `WorkReport_Id⁺`
- **WorkReportComment:** `Comment_Id⁺`, `WorkReport_Id⁺`

4.3 Test Results

In this section of Chapter 4 we will describe how the procedure of testing the application and web site would have been carried out. Unfortunately an official pilot test together with the field technicians could not been conducted as planned in Section 2.1.3, Structure of the design and implementation of the prototype, before the submission of this report. Instead we will describe the planning of the testing. The primary goals are to test the usability and user acceptance.
At a meeting with a service entrepreneur on the 17\textsuperscript{th} of May 2013\textsuperscript{2} we, together with Martin Hautamäki and product owners Bo Hermansson and Tommy Holm, decided how the testing would be carried out.

Before the testing both the users of the application as well as the administrator page will be given manuals instructing them how to use them. After the testing the users will have to fill out questionnaires. The application questionnaire (see Appendix A (Swedish)) targets how long it takes for the users to fill out the web form and how this time varies depending on age and gender. When testing the application ourselves we have noticed that the functionality differs on different operating systems, browsers and devices, therefore the testers are asked to specify those things in the questionnaire. The administrator page questionnaire (see Appendix B (Swedish)) is designed to make the results show if the testers can utilize the functionalities of the web site after reading the documentation. The questions ask the user to find information within the database (e.g. Find all posts with reference X. Which site is related to this reference?). If the testers find the information we are looking for we can determine whether the documentation and administrator page works as it supposed to.

In both questionnaires the testers are allowed to give their thoughts on the application or the administrator page.

The pilot testing will hopefully also help us discover the last bugs, if there are some left, before deploying the application to all service entrepreneurs.

5 Discussion

In this chapter the results of this project will be discussed and evaluated.

5.1 Working with Scrum

Having Scrum as a development method allowed us to be flexible in planning of the sprints and tackle the new requirements for the prototype as they came up. The product owners appreciated the frequently reoccurring demonstration meetings which gave them good insight to what had been accomplished and letting them interact with the evolving prototype. These are the two key factors of why we felt that Scrum would be a suitable approach for this project.

\textsuperscript{2}The report was due to the 24\textsuperscript{th} of May 2013
Sprint 3 is a good example of how we utilized the flexibility of Scrum. The product owners requested implementation of an acknowledgement functionality. After this was developed and demonstrated the product owners felt that the functionality was ambiguous and therefore it was removed in the upcoming sprint. If we had worked with another type of development method this might have been discovered too late and the functionality could have created new dependencies throughout the code. At this point it can be extremely hard to remove this functionality without disrupting those dependencies and could take days to remove. When working with Scrum we only lost one man-day of a sprint.

Another important part of Scrum is the planning where the burndown-charts provided us with a good overview of the velocity of completing stories in the previous sprint. This in turn helped us to determine which stories to take on in the next sprint. When reviewing the project burndown-chart in Figure 27 you might interpret our estimations to be cautious. We preferred rather to have room for errors and refinement of the completed stories than to overestimate and not being able to demonstrate the stories we had included in the sprint backlog.

5.2 Prototype

In this section of the discussion the web application, the administrator page and the user’s thoughts about them will be evaluated. Since we did not have the time to include the pilot testing in the report we cannot have any discussion over the test results.

5.2.1 Web application

The web application is a simple application that we and the product owners think meets the requirements set in the background for this project. In terms of functionality we do not see any advantages of developing this type of application individually for each operating system. The reason behind this being the major functionalities of the web application was completed as early as Sprint 2 out of the total six of this project.

5.2.2 Administrator page

At the demonstration of Sprint 2 it came to our attention that the administrator page might not be needed to monitor the incoming reports from technicians. The persons responsible for the Change Management System are going to develop a common administrator environment for both IMC and CMS. Because of this uncertainty of not knowing whether the administrator page was going to be used or not we were not sure of how much time we were going to put in to it. We and the product owners decided to continue working on our version of the
administrator page to provide a reference of how IMC would like the administrator page behave and function. In the end the administrator page became the main part of this project which introduced us to technologies other than the ones we are used to.

When we first started programming the test environment did not have the best conditions. The test environment hosted by TeliaSonera was not installed and configured which forced us turn to Karlstad University and their Atlantis domain. In the preparation of the last sprint Svante Viktorsson had setup the machine in a Telia domain and we were allowed to import the application and administrator page. Now we realized that the different configuration settings of the machines would matter. The new host required a more strictly interpretation of the code and did not prompt errors. This caused the migration of the application and administrator page more difficult than we anticipated and became a valuable lesson for us; test environment and production environment should be identical.

6 Conclusion

This chapter will present the writers final conclusions that came out of the discussion about the work that was carried out in Chapter 3. Since we did not have the time to include the pilot testing in the report we cannot make any conclusions over the usability of the application nor the administrator page.

6.1 Project

The progression throughout the project was good and the project owners were satisfied with the delivered product. By working with this project we have learnt more about different techniques behind how to develop web applications. We were introduced to the techniques AJAX and jQuery and our knowledge about HTML, JavaScript, PHP and MySQL-queries, which we have learnt in previous courses, were expanded. The courses in Database technology, Software Engineering, Java and Object-oriented design gave us a certain advantage that we made use of. The database technology was beneficial to the design of the database and queries towards it. Our knowledge about object-orientation was useful when creating PHP classes and using them as objects. Software Engineering, were we were taught how to apply Scrum on a project, helped us structure the planning and execution of work.

If we were to do this type of project again we could use bootstrap instead of building the administrator page from scratch to save time and enhance the esthetical part of the design.
Bootstrap [25] is a tool collection for creating applications and sites on the web developed by Twitter [26] and are freely distributed on GitHub [27].

6.2 Future Work

Even if all stories but the SMS story were completed the functionality of the prototype can still be improved and extended. Three major improvements and additions are:

- Alert
- Improved search
- Post structure

An alert function on the administrator page would help the operators at IMC not forget about jobs that have been active for too long without being updated. The function would use a JavaScript timer that triggers an alert to notify the operator.

The advanced search function of the administrator page could be more advanced. At the moment the search function makes a query that conjugates the user input. This could be improved by letting the user chose for which inputs to use conjunction or disjunction respectively, maybe even let the user to make own select-queries.

The administrator page could have a more sophisticated structure for organizing the presentation of posts. A structure more similar to the one used in the Active Jobs tab where the posts are displayed distinctly by site and reference, with possibility to expand and view all associated entries, would create a downsized appearance. By implementing this, the administrator page could possibly be loaded faster.

6.3 Summary

We had a good work progression throughout the project and overcommitted only twice during the sprints. This project has not only taught us a lot of new techniques but also how to apply what we have learnt during our time at the university. The applications fulfil almost every single request made by the product owners. The result that was delivered from this project is a result that we can take pride in and in the end was satisfying for both us and the product owners. This is not a completely finished product, there is definitely room for improvements, and will develop over time.
7 References


A Appendix – Site Attendance Application Questionnaire

TeliaSonera IMC Site Attendance Application Frågeformulär

Ålder: _____
Kön:
  o Man
  o Kvinna

Enhet: ________________________
(ex PC, Tablet, Mobil)

Märke: ________________________
(ex. HTC, iPhone)

Webbläsare: ________________________
(ex. Internet Explorer 8, Firefox, Google Chrome)

Tid: _____ (sek)
(Tiden det tar att fylla formuläret.)

Något som var oklart med applikationen eller användarmanualen?

__________________________________________________________

__________________________________________________________

__________________________________________________________

Övrigt:

__________________________________________________________

__________________________________________________________

__________________________________________________________
B Appendix – Administrator Page Questionnaire

TeliaSonera IMC Adminsida Frågeforumulär

Ålder: _____

Kön:
  o  Man
  o  Kvinna

Webbläsare: _______________________
(ex. Internet Explorer 8, Firefox, Google Chrome)

Försök att hitta följande information på administratörssidan:

- Alla ärenden med referens TS/RT/SF 99999999.
  - Vilken site gällde ärendet? ____________________________

- Alla poster gjordes från klockan 08:00 och 4 timmar framåt den 17:e april 2013.
  - Hur många poster? ____________________________

- “Acka” ett jobb som rapporterat som jobb färdigställt under tabben ”Active Jobs”.
  - Är den kvar?   Ja/Nej

- Öppna NMCP XXXXX i “porchen” genom administratörssidan.
  - Vad står det i porchärendet? ____________________________

- Alla jobb som teknikern “Anton” har gjort på site ABC?
  - Vad står det i kommentaren? ____________________________

Något som var oklart med applikationen eller användarmanualen?
Övrigt:
C  Appendix – Data model

In the data model we provide a detailed description for all the attributes, tables and relations existing in the database.

**WorkReport**

Description: Contains the information sent to the database from web application.

**Attribute**: Id  
Description: A unique number used to identify a WorkReport.  
Type: integer (11)  
Null: No.

**Attribute**: Time  
Description: The current date and time for when a WorkReport was entered in the database.  
Type: timestamp  
Null: No.

**Attribute**: Site  
Description: Name or id for the regarding site of a WorkReport.  
Type: varchar (50)  
Null: No.

**Attribute**: OnSite  
Description: Tells if the technician performs work that affects the site of a WorkReport.  
Type: varchar (3)  
Null: Yes.

**Attribute**: Coordinate  
Description: The geographical coordinates of the device that sends a WorkReport.  
Type: varchar (50)
<table>
<thead>
<tr>
<th>Null:</th>
<th>No.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attribute:</strong></td>
<td>Reference</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>Identifier for a ticket related to a WorkReport.</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>varchar (50)</td>
</tr>
<tr>
<td><strong>Null:</strong></td>
<td>No.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute:</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>Name of the technician sending a WorkReport.</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>varchar (50)</td>
</tr>
<tr>
<td><strong>Null:</strong></td>
<td>No.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute:</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>Phone number for contacting the technician who sends a WorkReport.</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>varchar (15)</td>
</tr>
<tr>
<td><strong>Null:</strong></td>
<td>No.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute:</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>Comments from the technician regarding the WorkReport.</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>varchar (250)</td>
</tr>
<tr>
<td><strong>Null:</strong></td>
<td>No.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute:</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>Category of the WorkReport.</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>varchar (250)</td>
</tr>
<tr>
<td><strong>Null:</strong></td>
<td>No.</td>
</tr>
</tbody>
</table>

| **Primary key:** | Id |
| **Secondary key:** | - |
| **Foreign key:** | - |

**Department**

**Description:** Contains a list of the different departments.
Attribute: Id
Description: A unique number used to identify a Department.
Type: integer (11)
Null: No.

Attribute: Name
Description: Name of a Department.
Type: varchar (50)
Null: No.

Primary key: Id
Secondary key: Name
Foreign key: -

Comment
Description: Contains a list of the different comments made by the IMC-operators.

Attribute: Id
Description: A unique number used to identify a comment.
Type: integer (11)
Null: No.

Attribute: Message
Description: The comment message made by a IMC-operator.
Type: varchar (250)
Null: No.

Attribute: Time
Description: The current date and time for when a Comment was entered in the database.
Type: timestamp
Null: No.

Primary key: Id
Secondary key: -
Foreign key: -

AssignDepartment
Description: Contains the information about which department a post belongs to.

Attribute: WorkReport_Id
Description: A unique number used to identify a WorkReport.
Type: integer (11)
Null: No.

Attribute: Department_Id
Description: A unique number used to identify a Department.
Type: integer (11)
Null: No.

Primary key: WorkReport_Id + Department_Id
Secondary key: -
Foreign key: WorkReport_Id, Department_Id

WorkReportComment
Description: Contains the information about which post a comment belongs to.

Attribute: WorkReport_Id
Description: A unique number used to identify a WorkReport.
Type: integer (11)
Null: No.
Attribute: Comment_Id
Description: A unique number used to identify a comment.
Type: integer (11)
Null: No.

Primary key: WorkReport_Id + Comment_Id
Secondary key: -
Foreign key: WorkReport_Id, Comment_Id