Independent degree project – second cycle

Masters’ thesis AV, 30 higher credits

Master of science in Engineering
Computer Engineering

RFID in Rail
RFID tracing of rail-borne cargo

Gabriel Holmlund
Abstract

SCA transports timber via rail from railway terminals in western Sweden to saw and/or pulp mills located on the east coast. Sometimes timber wagons are lost due to damages inflicted when loading the wagons. Wagons can also be decommissioned for maintenance conducted by the Swedish Transport Administration. Both situations are unpredictable and cause many types of problems. If the wagons could be automatically traced these problems could be predicted and preemptive actions could be taken. All wagons used by SCA are equipped with RFID tags and RFID readers are installed at the railway terminals.

The purpose of this thesis is to equip the RFID readers already placed at the terminals with Internet access, collect the data read by the RFID readers and present this data in a single page web application. The actual data collection is done by an already existing system which pushes the train data to the back-end part of the application. The front-end part is made with the Ember.js JavaScript framework and the back-end part is made with ASP.NET Web API and SQL Server. The application is finished and verified with various tests but is not put in production due to a negotiation delay between SCA with a third-party consultant.

Keywords: Ember.js, RFID, SCA, ASP.NET, train, mobile networks
Acknowledgements

Thanks to…

Lars Stefansson, SCA Skog
Lars Jonsson, SCA Skog
Roger Engström, SCA Skog
Victor Kardeby, Mid Sweden University
Table of contents

Abstract ............................................................................................................. ii
Acknowledgements ........................................................................................ iii
Table of contents ........................................................................................... iv
Terminology ................................................................................................... viii

1 Introduction ............................................................................................ 9
  1.1 Background and problem motivation .............................................. 9
  1.2 Overall aim................................................................................. 10
  1.3 Solution benefits.......................................................................... 10
  1.4 Scope ............................................................................................ 10
  1.5 Concrete and verifiable goals .................................................... 11
  1.6 Outline ....................................................................................... 11
  1.7 Contributions ............................................................................... 11
  1.8 Ethical aspects ............................................................................. 12

2 Background ........................................................................................... 13
  2.1 RFID ............................................................................................. 13
  2.2 The GS1 RFID in Rail standards .............................................. 15
    2.2.1 Tag data format ................................................................. 15
    2.2.2 Tag placement .................................................................. 16
    2.2.3 Reader setup ...................................................................... 17
  2.3 Mobile network types ................................................................. 18
    2.3.1 GSM .................................................................................. 18
    2.3.2 UMTS and CDMA2000 ................................................... 18
    2.3.3 LTE .................................................................................... 19
  2.4 Data formats and frameworks .................................................... 20
    2.4.1 HTML ............................................................................... 20
    2.4.2 CSS .................................................................................... 20
    2.4.3 XML and XML Schema .................................................. 20
    2.4.4 JSON .................................................................................. 21
    2.4.5 AJAX ............................................................................... 21
    2.4.6 REST ............................................................................... 21
    2.4.7 MVC ................................................................................... 23
    2.4.8 Single Page Application .................................................... 23
    2.4.9 ORM .................................................................................... 24
2.4.10 Ember.js 24
2.4.11 ASP .NET MVC 25
2.4.12 Entity Framework 25
2.4.13 Microsoft SQL Server 25
2.5 The extended EPICS data format 25

3 Method ................................................................. 27
3.1 Examination of different data communication options for the RFID stations ......................................................... 27
3.1.1 Network service provider alternatives 28
3.1.2 Measuring accessibility and bandwidth 28
3.2 Web application development ........................................................ 30
3.2.1 Capturing user requirements 30
3.2.2 Front-end development 30
3.2.3 Back-end development 31
3.3 Verifying the application ................................................................. 31
3.3.1 Verifying back-end 31
3.3.2 Verifying front-end 32

4 Implementation ................................................................. 33
4.1 Detailed system requirements ......................................................... 33
4.2 System overview ................................................................. 35
4.3 Front-end implementation ................................................................. 36
4.3.1 Reference design 36
4.3.2 User interface 37
4.3.3 Technical description 39
4.4 Back-end implementation ................................................................. 41
4.4.1 REST API 41
4.4.2 Database schema 41

5 Results ................................................................. 43
5.1 Wireless Internet service provider measurements results and coverage map comparison ................................................................. 43
5.2 Verification results ................................................................. 44
5.2.1 Rendering results 45
5.2.2 Data push results 47

6 Conclusions ................................................................. 48
6.1 RFID data connection alternatives ................................................................. 48
6.2 Implementation conclusions ................................................................. 49
6.3 Application verification ................................................................. 49
6.4 Future work ................................................................. 50

References ................................................................. 51
Table of contents

Appendix A: Network tables.................................................................57
Appendix B: REST API description....................................................63
## Terminology

### Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GPP</td>
<td>3rd Generation Partnership Project</td>
</tr>
<tr>
<td>AJAX</td>
<td>Asynchronous JavaScript and XML</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CDMA</td>
<td>Code Division Multiple Access</td>
</tr>
<tr>
<td>CRUD</td>
<td>Create, Read, Update, Delete</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>EDGE</td>
<td>Enhanced Data Rates for GSM Evolution</td>
</tr>
<tr>
<td>EPC</td>
<td>Electronic Product Code</td>
</tr>
<tr>
<td>FDD</td>
<td>Frequency Division Duplexing</td>
</tr>
<tr>
<td>GCP</td>
<td>GS1 Company Prefix</td>
</tr>
<tr>
<td>GIAI</td>
<td>GS1 Global Individual Asset Identifier</td>
</tr>
<tr>
<td>GMSK</td>
<td>Gaussian minimum-shift keying</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td>GSM</td>
<td>Global Standard for Mobile Communication</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>HSDPA</td>
<td>High Speed Downlink Packet Access</td>
</tr>
<tr>
<td>HSUPA</td>
<td>High Speed Uplink Packet Access</td>
</tr>
<tr>
<td>IC</td>
<td>Integrated Circuit</td>
</tr>
<tr>
<td>IMT</td>
<td>International Telecommunication Union</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>LTE</td>
<td>Long Term Evolution</td>
</tr>
<tr>
<td>MVC</td>
<td>Model, View, Controller</td>
</tr>
<tr>
<td>ORM</td>
<td>Object Relation Mapping</td>
</tr>
<tr>
<td>PSK</td>
<td>Phase Shift Keying</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency Identification</td>
</tr>
<tr>
<td>SAW</td>
<td>Sound Acoustic Wave</td>
</tr>
<tr>
<td>SCA</td>
<td>Svenska Cellulosa AB</td>
</tr>
<tr>
<td>SGLN</td>
<td>Global Location Number With or Without Extension</td>
</tr>
<tr>
<td>SPA</td>
<td>Single Page Application</td>
</tr>
</tbody>
</table>
RFID in Rail – RFID tracing of rail-borne cargo
Gabriel Holmlund

Terminology
2015-07-08

STA Swedish Transport Administration
TDMA Time Division Multiple Access
UMTS Universal Mobile Telecommunications System
URL Uniform Resource Locator
W-CDMA Wideband Code Division Multiple Access
XML eXtensible Markup Language
1 Introduction

Svenska Cellulosa AB (SCA) is a Swedish manufacturer of several forest-related products. Among these are paper, pulp and solid-wood products. 2.6 million hectares [1] of Swedish forest is owned and maintained by SCA. SCA supplies about 100 countries with personal care products, tissue and other related products [2].

1.1 Background and problem motivation

In order to keep up with the demand of forest products, wood must be acquired. Depending on the physical location of the logging site, wood is either transported directly to a saw or paper mill or transported to an adjacent railway terminal using trailers. Wood transported to a railway terminal is unloaded on site and is later transported further using trains. Wagons can be taken out of commission for maintenance. This is mainly for two reasons:

1. Wagons may be damaged during loading or unloading due to mistakes made by the truck drivers or weather conditions causing the cars to de-rail. This requires on-site technicians in order to make the repairs.

2. Heat sensors are placed along the railway to measure temperature in the wagon wheels. High temperature usually means damage to the wheels and the wagon must be repaired. The wagon is detached from the train and is led to an appropriate intersecting track for service. The task to decommission wagons for maintenance is done by the Swedish Transport Administration (STA).

The number of train sets running for a week is decided by train schedulers. The schedulers assume that no wagons are out of commission. STA has no responsibility to report when or which cars were taken out for maintenance. The repaired cars are attached to a later transport to the same destination. A following consequence is that arriving train sets carries either too few or too many tonnes of wood.
Today there is no way for SCA to look into the railway transport process. There is no information available regarding where the train cars are at a certain time or if cars are lost. Lost cars have to be traced manually by time-consuming telephone calls.

To counter this problem, SCA has in collaboration with the STA bought and installed one RFID reader placed along the railway in Östrand pulp mill. RFID tracing technology is already in use by the STA [3]. Three more are placed in the Töva, Bensjö and Östavall railway terminals. Two more are to be installed in the Krokom and Hoting railway terminals. The installed readers have power but no way to communicate with other systems. Passive RFID tags have been installed on every train car in use.

1.2 Overall aim
The overall aim of the project is to find a way to automatically trace the railways cars and make the information about the cars accessible within a pilot stand-alone system. A successful implementation would allow SCA to gain better insight into the timber hauling process. The outcome of the pilot system will work as a basis for further development.

1.3 Solution benefits
The solution will allow SCA to plan ahead in case cars are lost or gained. On-site unloading crew can access information concerning a train which will arrive later and plan truck placement more efficiently. Administrators can see when and where a train car is lost. SCA must pay a set fee to the STA for every transported car. If a car is lost due to maintenance, correct fees can be debited instead of paying for a full train. An RFID solution gives a better estimated time of arrival for a train set and stock-keeping becomes easier.

1.4 Scope
This study aims to find a way to trace railway cars hired by SCA. The entire solution might not be applicable in other systems. While it is a part of the project to enable data collection from the RFID stations, it is not a part of this thesis to program the stations. The RFID readers come pre-programmed with this software. Data collection is done by an already existing system hosted by Learningwell, a consultant company hosting RFID data collection services for the STA.
The implemented solution will be a web-based application. Making a web application designed to fit all screen sizes and web browsers is both time-consuming and requires access to devices with different prerequisites. Thus, the implementation will be developed to work with Google Chrome and desktop computers. The solution will be delivered as a single-user application but will be prepared to enable multiple users in the future.

Security will not be of concern while developing the application. The transmitted data is deemed as non-sensitive and non-interesting for external parties. Security measures can be added afterwards if required.

1.5 Concrete and verifiable goals

The goals of this thesis are the following:

• Examine different options to enable data communication with the RFID stations and possible implementation alternatives for the web application.

• Propose and realize a solution enabling communication with the RFID stations.

• Implement an application giving SCA a better view into the rail-borne transports.

• Verify implementation success by testing.

1.6 Outline

Chapter 2 contains background material and brief descriptions of different technologies used in this thesis. Chapter 3 holds the method used to fulfill the thesis goals. Chapter 4 holds implementation details and user requirements. Chapter 5 contains results from measurements and an evaluation of the implementation. In chapter 6 the results are discussed.

1.7 Contributions

All practical work and implementations presented in this report is done by the author. Parts of the theoretical work are taken from previous
courses. Other students experienced with the used programming frameworks helped with tips and instructions when asked for.

1.8 Ethical aspects
The usage of cookies in web browsers can be used to keep track of individual user information. Cookies can be used by advertisement companies to figure out browsing patterns in order to display advertisements directed towards the user. However, cookies can also be used to enhance browsing experience by remembering user preferences. Swedish law states that all websites using cookies must inform the user if cookies are used [4]. If cookies were to be used in this application, usage would only cover authentication and session management. Information stored in cookies would only exist client-side and server-side of the application.

This application does not intend to “replace” any employee hired by SCA. The work done by this application is meant to make the current employees’ situation more efficient and comfortable. Wagon tracing is (stated by SCA representatives), “both tedious and boring”.
2 Background

This chapter covers background material necessary to understand all terminology used in this report. This chapter is intended to be used as a reference. Chapter 2.1 and 2.2 gives a brief explanation concerning RFID and the standards used with RFID data. Chapter 2.3 contains a short description of different technologies used when implementing the application. Chapter 2.4 explains the data format used when receiving RFID data from the Learningwell service.

2.1 RFID

Radio Frequency Identification (RFID) [5] is a way to transfer data between a reader and a tag. When a tag and a reader is close enough, data can be exchanged.

The tags consist of an antenna and a microchip equipped with read only memory [5]. The electromagnetic field emitted by the reader activates the tag (with the exception of beacon tags) and causes the tag to transmit the contents of the memory chip. Transmitted data usually contain a unique ID number associated with the tag. Much like barcodes this method enables the association of a physical object to a digital ID.

Tags can either be active or passive [6]. There are two types of active tags – beacons and transponders. Both active types require batteries while passive tags do not. A beacon tag transmits memory data periodically, while transponder tags transmit data when a reader is detected. Passive tags are entirely powered by the readers’ electromagnetic field.

The ISO/IEC 18000 document is divided into seven parts and describes a set of common RFID frequencies. The following table contains these frequency bands with examples of reading distances using passive tags [7].

<table>
<thead>
<tr>
<th>Frequency Range</th>
<th>Frequencies</th>
<th>Passive Read Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Frequency</td>
<td>120-140 KHz</td>
<td>10-20 cm</td>
</tr>
</tbody>
</table>
Different frequency bands have different application areas. The low frequency band is utilized when short read distance and low data rate is required [7]. For example, when implanting a pet with an ID tag or having a tag in car keys for extra security. These tags are usually passive and have very small antennas which limit the reading range to a few centimeters. The high frequency band offers a higher data read range than the low frequency band, but does not tolerate environments such as metal or liquids as well. The high frequency band is heavily restricted and readers must work in narrow frequency ranges. The ultra-high frequency band is mostly used for item tracing. This band offers a longer read distance and tags are cheaper to produce than low or high frequency tags. A problem is that European readers operate within the 868-870 MHz range while U.S. and Canadian readers utilize the 902-928 MHz band causing incompatibility between systems. The microwave and ultra-wide bands requires in most cases active or semi-passive tags, making them more expensive to produce but offers a higher read distance than their lower frequency counterparts. Since semi-passive and active tags have an on-board battery, the tags require maintenance which makes these bands unfit for applications where such maintenance is hard or impossible.

The applications areas for each frequency band and statements regarding range are recommendations based on both theory and practice. Many factors must be included when deciding tag types and frequency band when setting up an RFID system. Larger antennas come with higher antenna gain which increases the reading range but large antennas might not be possible in every application. Reader and tag transmit power affects as well as the chosen frequency band affects the range and data rate. RFID readers typically output a power level between 1 and 4 watt [8].

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>High Frequency</td>
<td>13.56 MHz</td>
<td>10-20 cm</td>
</tr>
<tr>
<td>Ultra-High Frequency</td>
<td>868-928 MHz</td>
<td>3 meters</td>
</tr>
<tr>
<td>Microwave</td>
<td>2.45 &amp; 5.8 GHz</td>
<td>3 meters</td>
</tr>
<tr>
<td>Ultra-Wide Band</td>
<td>3.1 - 10.6 GHz</td>
<td>10 meters</td>
</tr>
</tbody>
</table>

Table 1: RFID bands and examples of passive read distances.
In the case of passive RFID, the Friis transmission equation can give a naïve idea of how the reader powers the tag:

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{\lambda}{4\pi R}\right)^2$$

(1)

Where $P_t$ is the transmitting power and $P_r$ is the power generated at the receiving antenna. $G_t$ and $G_r$ is the antenna gains in the transmitting and receiving antenna. $\lambda$ is the frequency wavelength and $R$ is the distance between the antennas. (1) can give an idea of how these parameters are connected but more refined theory is necessary when implementing a RFID system in practice. Factors such as interference, direction of antenna and fading must be accounted for. One study uses refined theory and uses monostatic and bistatic systems with environmental interference to produce 3-D data with the conclusion that different types of fading, the number of interference and tag misalignment affects the reading distance [9]. Another study compares Integrated Circuit (IC) tags and Sound Acoustic Wave (SAW) tags and concludes that SAW tags can have up to 30x the range of an IC tag [10]. The production environment causes a large impact on the actual performance and testing should be conducted with different setups before installing an RFID system.

2.2 The GS1 RFID in Rail standards

The worldwide standards organization GS1 [10] have a standardized method of tracing rail-borne objects which is in use by interested member countries. The use of standards comes in handy when transporting cargo across country borders. For example, 70% of cargo wagons in Sweden are foreign [11]. It would be hard to trace all wagons if different standards were in use.

2.2.1 Tag data format

The Electronic Product Code (EPC) tag data standard made by GS1 specified in The EPC Tag Data Standard [12] defines many types of data formats. This standard is used worldwide by companies and governments wishing to have a compatible way of tracing cargo in other participating countries. A small subset of these data formats has been adopted by STA and is used in the currently existing RFID tracing system. Below is a list of used components with brief explanation.
RFID in Rail – RFID tracing of rail-borne cargo
Gabriel Holmlund

Background
2015-07-08

GCP
The GS1 Company Prefix (GCP) [12] is a number allocated by GS1 to organizations wanting to track their objects. It is a unique number of variable lengths and is included in tag data attached to rail-borne property.

Example: 7332862

GIAI
The GS1 Global Individual Asset Identifier (GIAI) [12] is given to a RFID tag which can be attached to wagons, trailers or other means of transport. The GIAI number serves as the objects unique ID.

General syntax:
urn:epc:id:giai:CompanyPrefix.IndividualAssetReference
Example: urn:epc:id:giai:7332743.2200000002018

SGLN
The Global Location Number With or Without Extension (SGLN) [12] is a number assigned to a unit with a specific physical location. This number is assigned to RFID readers.

General syntax:
urn:epc:id:sgln:CompanyPrefix.LocationReference.Extension
Example: urn:epc:id:sgln:7332862.00030.0

2.2.2 Tag placement
One tag must be placed on each long side of a rail-borne wagon. The reason is that the wagon must be detectable for a reader regardless of travel direction. The standard defines four wagon sides called side A, side B, end 1 and end 2. Side A is the right hand side of the wagon while side B is the left hand side when facing the front of the wagon. If the wagon has a handbrake, that side is the back side (end 2). Otherwise, side A and B and can be chosen at will. Tag data contain a side indicator; either 1 or 2 which reveals tag location. Tags with side indicator 1 should be placed on side B towards end 1. Tags with indicator 2 should be placed on side A towards end 2. [12] This placement can be seen in illustration 1.
2.2.3 Reader setup

A RFID reader is placed along the railway track accompanied by two axle counters, one on each side of the reader along the track. See image 1.

When a train passes the RFID station, the unique GIAI numbers from the tag on each wagon is collected and sent to the Learningwell service. Axle counter events are also sent. Axle counter data is used to determine train direction and can be used to detect a missing tag. One axle counter event is generated every time a wagon axle passes an axle counter. A typical wagon has four axles which results in eight axle events for every wagon passing the RFID station.
2.3 Mobile network types
This chapter gives an introduction to the types of mobile networks relevant for this project. Chapter 2.3.1 to chapter 2.3.3 introduces these network types.

2.3.1 GSM
Global System for Mobile Communications (GSM) is a standard describing second generation cellular networks for mobile phones. It uses Frequency Division Duplexing (FDD) in combination with Time Division Multiple Access (TDMA) [13]. GSM is a single carrier system and usually operates in the 900 MHz band. GSM is a circuit switched network. General Packet Radio Service (GPRS) is a packet-oriented data service which can be used to send Internet Protocol (IP) packets over a GSM network. GPRS has a minimum data rate of 8 Kbit/s per time slot allocated by the base station. Enhanced Data Rates for GSM Evolution (EDGE) [14] is an improvement of GPRS. This can increase the bit rate to 384 Kbit/s and even further if more advanced EDGE technology is in use. Depending on code rates and technologies used, the data rate varies between the minimum up to the EDGE-enabled maximum. Compared to the other network types presented in this chapter, GSM with evolution technologies provides significantly lower data rates than the other network types. However, since GSM networks are the most established due to their age, the best reliability can be expected.

2.3.2 UMTS and CDMA2000
Universal Mobile Telecommunications System (UMTS) and the Code Division Multiple Access (CDMA)-based network CDMA2000 are 3G networks. A mobile cellular system can be branded as 3G if the network meets the IMT-2000 standards [15] made by International Telecommunication Union (IMT). The network is required to provide data rates of 200 Kbit/s downlink for the end-user. UMTS is the most common 3G network type. It is based on the GSM system but uses Wideband-CDMA (W-CDMA) instead of TDMA. W-CDMA uses a channel width of 5 MHz and usually operates in the 2100 MHz band. The UMTS systems come in several releases specified by the 3GPP group and performance varies depending on which release the service provider uses. Release 5 [16] introduced the High Speed Downlink Packet Access (HSDPA) which allowed users to reach downlink speeds of up to 10 Mbit/s and release 6 [17] introduced the High Speed Uplink Packet Access (HSUPA). Further
evolution adding more code schemes in later releases allows higher data rate.

Another network type branded as 3G is the CDMA2000 [18] networks. Like UMTS, it uses CDMA as a channel access method but uses 1.25 MHz wide channels and is thus useful in frequency bands where the total channel space is limited. CDMA2000 1xEvolution Data Optimized (EV-DO) is an evolution technology optimizing the network for data traffic. Like UMTS, CDMA2000 1xEV-DO comes in many revisions. Revision B uses multiple carriers to increase downlink speed. Depending on the number of carriers, the downlink data rate can reach 14 Mbit/s.

UMTS with extensions provide better data rate than both GSM and CDMA2000, but it takes up more channel width than CDMA2000. In cases where lower bandwidth is acceptable, CDMA2000 can be used to save bandwidth.

### 2.3.3 LTE

4G networks succeeds 3G networks in terms of data transfer speeds and should offer 1 Gbit/s downlink speed for stationary clients and 100 Mbit/s for mobile clients. These specifications are set in the IMT-Advanced specification. While both GSM and 3G networks supports circuit switching, 4G networks operates with packet switching only and can scale channel bandwidth between 5 to 20 MHz. Long Term Evolution (LTE) [19] is a network technology which in its first version offers 100 Mbit/s in the downlink and 50 Mbit/s in the uplink. It does therefore not classify as a 4G network, but operators choose to market it as 4G. LTE was introduced in 3GPP release 8 [19]. As with UMTS, LTE also comes in several releases where 3GPP release 10 [20] and later is referred to as LTE Advanced. LTE uses Orthogonal Frequency Division Multiple Access (OFDMA) in the downlink and Single Carrier Frequency Division Multiple Access (SC-FDMA) in the uplink as the multiple access method. LTE provides the best bandwidth of all the relevant network types, but it is also the newest. This can mean that operators have not extended the network with base stations to provide coverage on all sites.
2.4 Data formats and frameworks

Chapter 2.4.1 to 2.4.13 gives background information on data formats, frameworks and concepts used in this thesis.

2.4.1 HTML

Hyper Text Markup Language (HTML) is a markup language used to describe a web site or application. Every component seen on a web site is made of one or more HTML elements. Web browsers reads HTML documents and translates these into visual information displayed in the browser window.

2.4.2 CSS

Cascading Style Sheets (CSS) is used to give HTML elements visual properties. CSS is used for element placing, size, color etc. CSS can also be used to animate HTML elements.

2.4.3 XML and XML Schema

Extensible Markup Language [21] (XML) is a language used to describe data. XML is meant to be self-describing, meaning that no previous knowledge about the data structure is necessary in order to read it. XML requires the programmer to define own data-describing markup tags. This makes an XML document human readable depending on chosen tag names and structure. An example below describes a brown, striped cat with a weight of 4 kg.

```xml
<cat>
  <color>brown</color>
  <weight>4</weight>
  <pattern>striped</pattern>
</cat>
```

While XML gives the freedom to define arbitral data structures, it is often mandatory to follow a specific data structure. An XML Schema [22] is a special type of XML document describing how written XML should be formatted in order to follow the standards used in, for example, an organization. Written XML can be validated with the XML Schema to verify structural and data integrity.
2.4.4 JSON

JavaScript Object Notation [23] (JSON) is a data-interchange format meant to be lightweight and easy to read and write both by machines and by humans. Together with XML, JSON serves as one of the most common data-interchange formats across the web. Differing from XML, JSON is not a markup language. JSON structure and data can be pre-defined (like XML) with JSON Schemas [24]. An example below describes a brown, striped cat with a weight of 4 kg.

```json
"cat": {
  "color": "brown",
  "weight": 4,
  "pattern": "striped"
}
```

2.4.5 AJAX

Asynchronous JavaScript and XML [25] (AJAX) is used to send asynchronous requests from a web browser to a server. In web browsing context, an asynchronous request is one that does not reload the current page (the request happens in the “background”). The web browser can process the response and use JavaScript to take some action. AJAX is used to create fast and usable web sites or web applications.

The “X” part in the acronym implies the data-interchange format to be XML. While XML is an option other data formats can also be used, such as JSON or plain text.

2.4.6 REST

REpresentational State Transfer [26] (REST) is an architecture used on servers to handle incoming requests from any device using the HTTP protocol. It makes use of the existing Hyper Text Transfer Protocol (HTTP) verbs (GET, POST, PUT and DELETE) attached to the request header to provide service accordingly. The request payload (if any) provides data for the server to handle. The HTTP verbs correspond to an action carried out server-side. The REST Application Programming Interface (API) creator has full control over these actions. According to the Create, Read, Update, Delete (CRUD) model, the verbs should invoke the following actions:
HTTP verb | Action
---|---
GET | Read data.
POST | Create data.
PUT | Update data.
DELETE | Delete data.

Table 2: HTTP verbs and typical resulting actions when using a REST API.

Below is an example to further explain how REST works.

A REST-ful server provides an API for handling employee records. The base Uniform Resource Locator (URL) to the server is http://www.employees.com. The servers REST API is available at /api.

<table>
<thead>
<tr>
<th>Request</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /api/json/employee</td>
<td>Returns a list of all employees and related data</td>
</tr>
<tr>
<td>GET /api/json/employee/1</td>
<td>Gives data related to employee with id 1.</td>
</tr>
<tr>
<td>POST /api/json/employee</td>
<td>Creates a new employee in server database using data from request payload</td>
</tr>
<tr>
<td>PUT /api/json/employee/1</td>
<td>Updates employee with id 1 using payload data</td>
</tr>
<tr>
<td>DELETE /api/json/employee/1</td>
<td>Deletes employee with id 1 from server database</td>
</tr>
</tbody>
</table>

Table 3: Examples of REST requests and typical outcomes

The example URL contains “json” which indicates that JSON is used as data-interchange format. XML could have been used instead to handle the same data with “xml” in the path instead of “json”.
2.4.7 MVC

In web context, Model, View, Controller (MVC) [27] is a software programming pattern where an application is divided into three parts. The model represents data a user interacts with and logic tied to this data. The view is a human-friendly representation of the models and the controller handles user actions and invokes model methods.

The model is completely isolated from any type of GUI and only handles data manipulation and algorithms tied to model data. Examples are CRUD actions, sorting and filtering.

The controller handles user interaction such as clicks or keyboard events. These events can lead to invoked model methods or updates in the view.

The view is a representation of model data. Same model data can have many different views. Depending on user interest, the same data can be viewed as a pie chart or a table.

2.4.8 Single Page Application

A single page application (SPA) [34] is a web application intended to provide full functionality after a single page load from the server. In a single page application the page is only rendered once. Further requested data is loaded from the server using some asynchronous method (such as AJAX) and then inserted into the web page using JavaScript. The server provides the requested data in some pre-decided format (such as JSON or XML) and does not need to generate new HTML. Since no HTML generation is required, server load is reduced.

The purpose is to enhance user experience and reduce server load. Traditionally when a user clicks a link or otherwise requests data from the server a fresh page is rendered by the server. This causes the page seen by the user to be reloaded and re-rendered in the browser window. A drawback with this method is that the browser page goes blank when the page reloads. The server must build new HTML, send it to the client and the client has to re-render the entire web site. The HTML generation loads the server.
2.4.9 ORM

Object Relation Mapping (ORM) is a process where data from a relation database such as SQL Server is used to build objects in an object-oriented programming language. These objects acts as an abstraction layer between the programmer and the database. Database information can be manipulated, created or destroyed using these objects.

2.4.10 Ember.js

Ember.js is a front-end JavaScript framework used to develop user-responsive single page applications. It employs two-way data binding and comes with its own template rendering engine [28] [29]. Ember.js uses “convention over configuration” which means that naming conventions across the Ember.js modules enables Ember.js to automatically wire up modules instead of using a configuration file [30].

An Ember.js route is a module which handles a specific URL. Each route comes with one or more Ember.js controllers, and one or more templates [31]. The controller handles user actions and manipulates the template contents. The template holds the HTML for a section with extensions which is interpreted by Ember.js when the site is active. For example, www.employees.com/jobs would correspond to the “Jobs” route, the “Jobs” controller and the “Jobs” template.

Ember-data is a module which allows programmers to define models and set relations between models similar to the relations in a relational database [32]. An instance of a model is called a record. The collection of all records is called the store. When a user navigates to certain location on an Ember.js site the corresponding route module can load records from the store. These records will be available in both the attached controller and template. Records can also be found using the pre-defined relations.

When a route module or a controller module fetches records from the store, the Ember.js RESTAdapter [33] asynchronously queries the backend server for data and returns a collection of records to the route or controller. Records loaded from server-side are cached in the web browser, meaning that several queries for the same record are only loaded once from the server.
2.4.11 ASP .NET MVC

ASP .NET MVC is a web-development framework made by Microsoft. It is used to create dynamic web pages and web services. The models and controllers reside on the server while the view part is displayed in the web browser. The view part can be omitted and replaced with a Web API instead [34]. The server code can be written in any .NET compatible language such as C# or Visual Basic.

2.4.12 Entity Framework

The Entity Framework can be used when creating ASP .NET MVC applications. It is used to handle databases and serves as an ORM [34]. It allows the programmer to generate database tables based on models and simplifies data access. It can also be used to set up database constraints.

2.4.13 Microsoft SQL Server

Microsoft SQL Server is a relational database management system used to host and manage several databases. The Entity Framework uses SQL Server for persistent storage [35].

2.5 The extended EPICS data format

Data transmitted from the Learningwell service follows the GS1 EPICS data format [36] with extensions made by the STA. The data consists of an event list with elements containing event information. Events can origin from an RFID reader or an attached axle counter. The following example is an RFID event:

```xml
<AggregationEvent>
  <eventTime>2013-02-12T16:24:45.347+01:00</eventTime>
  <eventTimeZoneOffset>+01:00</eventTimeZoneOffset>
  <parentID>1521109</parentID>
  <childEPCs>
    <epc>urn:epc:id:giai:7332743.2200000002018</epc>
  </childEPCs>
  <action>OBSERVE</action>
  <bizStep>PassageE</bizStep>
  <readPoint>
    <id>urn:epc:id:sgln:7332862.00030.0</id>
  </readPoint>
</AggregationEvent>
```
The event contains date and time, time zone offset, the readers SGLN number in the <readPoint> tag, RFID tags GIAI number in the <childEPCs> tag. The <bizStep> value indicates train direction and can either be PassageE or PassageO. The <parentID>, <action> and STA extension tags preceded with trf: are not interesting in this project. The axle counter event is similar with some differences:

The GIAI number is replaced with a SGLN number referring to the axle counter. The <disposition> tag indicates an axle counter event. The SGLN number in the <readPoint> tag refers to the associated RFID reader.
3 Method

This chapter describes the chosen methods to achieve the thesis goals. Chapter 3.1 explains the method used to decide which data connection the RFID (see chapter 2.1) stations should have. Chapter 3.2 describes the application implementation method and chapter 3.3 explains the solution verification method.

During this project an iterative method with regular meetings with SCA representatives was used. By the end of each iteration, a partial solution was delivered and the next part could be planned. The iterations were not of equal length, since different parts require different amounts of time. In the beginning of the project, the iterations were shorter since the initial analysis required more meetings and the author had to grasp the situation, learn how the lumber logistics were handled and find and study existing technical solutions. The actual implementation had longer iterations.

3.1 Examination of different data communication options for the RFID stations

The RFID stations needed network access in order to communicate with the Learningwell service. There were three options when deciding network type:

1. Connect the RFID station to SCAs intranet on site.

2. Connect the station to fiber cable provided by the STA running along the railway track [37].

3. Install a wireless internet service provider modem in the station.

Option 1 requires an underground Ethernet cable running from the nearest switch to the station. SCA enforces heavy security policies on their intranet which would make communication with the Learningwell service difficult.
Option 2 is not meant for end-users and would be too expensive. The installation process would have to involve the STA and is more complicated than necessary.

Option 3 requires strong enough on-site network signals. It also requires a modem for each terminal.

Option 3 was chosen as the best method assuming that strong enough network signal is present. It also required a choice of network service provider. To determine which network service provider to use one must know the network conditions on the sites where the RFID stations are placed. One way to do this was to consult the coverage maps hosted by the network provider. Since coverage maps are based on assumptions of signal availability and quality made by the provider they cannot be trusted when deciding network provider. In this study network measurements will be done at site to get a result based on the actual environment.

The purpose of this measuring process was to determine which networks were available. The RFID stations do not transfer large amounts of data, which means that low bandwidth is acceptable. Unstable network signal or unavailability is not.

### 3.1.1 Network service provider alternatives

In this study measurements of Telia and Net1 will be done. Telia provide LTE, UMTS and GSM (see chapter 2.3) networks and have a contract with SCA since before this project. This contract is the reason why similar providers such as Tele2 or Telenor offering the same network types were excluded.

Net1 claims to cover 95% of Swedish land using the 450 MHz frequency [38] which differs from Telias higher frequency networks. The improved coverage using lower frequencies makes Net1 an alternative if Telia networks are not available at site. Net1’s networks use CDMA2000 (see chapter 2.3.2).

### 3.1.2 Measuring accessibility and bandwidth

Measurements will be done at each terminal listed in chapter 1.1 using a Samsung SM-T905 tablet equipped with a Telia SIM-card. An application for measuring bandwidth and response time named Bredbandskol-
len [39] were priory installed and used. While the tablet alone can only measure Telia signals, a battery powered Net1 R-90 [40] modem were set up on site. The R-90 serves as a Wi-Fi access point which the tablet can connect to.

When doing the measurements, both the tablet and the R-90 modem were placed in height with the RFID station. Five rounds of measuring were done for each network type. While response time and bandwidth can be seen in Bredbandskollen, the current signal strength can be seen in the tablets settings. In the Net1 case, signal strength can be seen in the R-90’s administrative web page accessible through the tablets web browser. The measurements were done in one minute intervals.

The measuring environment at the railway terminals was similar. The terminals have a similar outlay with a railway switch at the one of the ends of a rectangular storage area. The RFID station is placed close to the railway switch. The immediate surroundings of the RFID station were either woods or open area depending on the site. The Töva, Östavall and Östrand terminals had open area surrounding the station while the Bensjö, Hoting and Krokom terminals had woods. The measurements were all done during clear weather. Measurements were only done once at each terminal since it involved covering large physical distances by car.

After measuring the results was be evaluated and a decision whether Telia or Net1 should be chosen as service provider can be made. Industrial standard wireless modems must be purchased and installed at the RFID stations. Learningwell requires the modems to have OpenVPN support. The RFID stations are prepared with a 230v power outlet and a data interface which the modem should be connected to. Prior to installation the modems must be configured with OpenVPN keys and hostnames by Learningwell. Otherwise, the modems can not connect to the Learningwell service.

Axle counter calibration must be done before putting the stations in production. A special calibration block is dragged across the axle counters. This will be done in conjunction with modem installation.
3.2 Web application development

This chapter gives a description of the application development process. The application is divided into two parts, the front-end part and the back-end part. Chapter 3.2.1 describes how user requirements for the application were captured. Chapter 3.2.2 describes the method used for front-end development while chapter 3.2.3 gives the method for back-end development.

3.2.1 Capturing user requirements

To capture user requirements several brainstorming meetings were held with SCA representatives. During each meeting, previous noted requirements were re-evaluated and refined until a preliminary set of well-formed requirements was defined. Over time, addition, modification and removal of requirements were made to fit both scope of the project and user requirement. Demonstrations were held and SCA representatives had chances to touch and experience several prototypes and give feedback on design and interaction.

During the meetings a specific set of data of interest was defined by the SCA representatives. This data set was used when forming the entities used both in the front-end and back-end part of the application. The data set describes properties of trains, terminals and wagons.

No specific non-functional requirements were set due to the fact that the project is a pilot project intended to make a base for further development. However, unreasonable data propagation times make the application useless. The train data must be available within seconds after reception. The application must load within reasonable time, and site navigation must be fairly quick.

3.2.2 Front-end development

After a mutual requirement agreement with the customer a user interface mock-up was designed. SCA personnel already use a transport web application made for SCAs trucks and boats, which is used on a daily basis. Since SCA personnel already has experience with the user interface of the existing application, a modified interface was used in the front-end application.

After acceptance of the design, a design implementation using HTML (see chapter 2.3.1) and CSS (see chapter 2.3.2) was made. This allowed
SCA representatives to give feedback and opinions on the design and the user interface before the functionality was implemented.

Since no specific front-end application framework requirement was made, a single page application (see chapter 2.3.8) using Ember.js (see chapter 2.3.10) was made. Ember.js was chosen due to earlier experience using similar “convention over configuration” MVC (see chapter 2.3.7) frameworks. Knowledge necessary in order to implement full functionality was taken from the official Ember.js documentation, Internet searches or trial and error. The front-end application was implemented using Sublime Text 2 [41] since it is cross platform and comes with much functionality.

### 3.2.3 Back-end development

The back-end application will serve the Ember.js framework and static files to the clients. It will also serve as a REST (see chapter 2.3.6) API which is used by the front-end application. SCAs current web solution uses Microsoft products. This includes both the internal and external web. To simplify deployment, Microsoft ASP .NET MVC (see chapter 2.3.11) was chosen to use as back-end technology. A created back-end application could be deployed onto an already existing application server. This application server is connected to a database management system which holds the database. To simplify database management, Entity Framework (see chapter 2.3.12) was chosen as the ORM (see chapter 2.3.9). Knowledge necessary in order to implement full functionality was taken from the official ASP .NET documentation, Internet searches, guides made by Microsoft or trial and error. Since C# and ASP.NET is easiest to program in Microsoft Visual Studio [42], that particular IDE was chosen when implementing the back-end application.

### 3.3 Verifying the application

This chapter holds the methods used to verify the back-end and front-end application. Chapter 3.3.1 describes the method used to verify back-end and chapter 3.3.2 gives the method for front-end verification.

#### 3.3.1 Verifying back-end

The application is fully functional when RFID data is translated into visible and understandable information in the application. The applica-
tion should be able to present the train information within seconds after receiving data from the Learningwell service. To verify this, the application was deployed onto a web server and a client loaded the application in the web browser. Another machine pushed previously known XML data with the format described in chapter 2.4 to the application. Each push held data for either 15 or 30 wagons, which are the train lengths used by SCA. There is a high probability that two or more data pushes will be done at the same time when the system is in production. Therefore, up to 5 parallel data pushes were tested. A test round is considered successful if correct data is available in the application 15 seconds after pushing. Directly after sending the data, a stopwatch was used to keep time. After 12 seconds, the refresh button was hit in the web browser. The application is then loaded after 15 seconds, assuming a three second application load. Each RFID event has eight associated axle events (four axles on each wagon with two axle counters). The terminals SGLN numbers were mocked. The tests were conducted using REST Commander [43].

3.3.2 Verifying front-end

Rendering tests with different amounts of train data was made to evaluate the front-end application. Initial loading and rendering can be expected to take more time than navigation within the application since Ember.js has to set up the entire application. Both navigation initial load test were conducted. Initial load tests were done by doing a full refresh of the application with varying table sizes. Navigation tests were done by recording the click event when navigating from a terminal with no train data to a terminal with train data. Idle time when the client is waiting for data was ignored since network conditions are unpredictable. The test data was collected using Google Chromes timeline tool. A limitation of maximum 60 table rows was set since it is unlikely for a user to view larger tables without filtering. The rendering tests were done on a 2011 MacBook Air with the following specifications:

- CPU: 1.8 GHz Intel Core i7, Dual core
- Memory: 4 GB 1333 MHz DDR3
- Graphics: Intel HD Graphics 3000 384 MB
- Screen resolution: 1440 x 900
4 Implementation

This chapter gives user requirement details and implementation details. Chapter 4.1 contains the functional requirements set by the user and a series of data entities which the application will show. Chapter 4.2 contains a non-detailed system overview. Chapter 4.3 holds the front-end implementation and chapter 4.4 describes the back-end implementation.

4.1 Detailed system requirements

Before implementation a series of user stories based on meeting notes were constructed in order to clarify functional requirements. Below is a list displaying said stories.

- As a user, I want to see train data in table form so I can get an overview of the current situation.

- As a user, I want to add favorite terminals so I won’t have to search for it every time I use the application.

- As a user, I want to search the terminal database so I can select which trains to display.

- As a user, I want to filter the trains by date so I only see the trains I’m interested in.

- As a user, I want to be able to sort the train data by some property so can get a better overview.

- As a user, I want to be able to select loading terminal A and B (or more) and unloading terminals C and D (or more) so I can find the train data I’m looking for.
The customer specified a set of data which the application must show. This data was grouped into entities which are used in the front-end application, back-end application and in the database. These entities are listed below.

User:
- Name
- Favorite terminals

Terminal:
- Name

Train:
- Loading terminal
- Arrival date and time before loading
- Number of train wagons when arriving
- Departure date and time after loading
- Number of wagons when departing
- Destination terminal
- Arrival date and time for unloading
- Number of wagons when arriving at destination terminal

Wagon (intender for future use):
- Type of cargo
- Volume of cargo
4.2 System overview

A system overview can be seen in image 2.

Image 2: System overview.

RFID data is collected using an existing service provided by Learningwell cloud service. RFID data is pushed from the Learningwell cloud in XML (see chapter 2.3.3) form to the back-end application. Data is parsed and stored into a database connected to the back-end server. When a user logs onto the system via a web browser the same data is sent to the front-end application in JSON (see chapter 2.3.4) format.
4.3 Front-end implementation

This chapter describes the user interface and gives a technical description of the front-end application. Chapter 4.3.1 holds the reference design while chapter 4.3.2 describes the front-end applications user interface.

4.3.1 Reference design

Image 3 is a screenshot of the reference design used when building the application.

Image 3: Reference design

A large table covers most of the screen. Sorting and filtering is done in the table header. A slide-out menu is located at the left hand side next to a button which displays a map. A slider favorite button and user information can be seen in the top menu.
4.3.2 User interface

Image 4 is a screenshot of the user interface seen when starting the application by visiting the applications URL.

![User interface screenshot of “Hoting”, one of the terminals.](image)

Initially, no information at all is show. The user must either choose a terminal from the left-hand slide menu, search for a terminal in the top search input or set up a filter in the right hand side menu. Each table row represents a train loaded at the chosen terminal. The columns show the information listed in chapter 4.1.

If the user wants to sort the table on some property, a click on the column header containing the desired sorting property will display a sorted version of the table. A second click on the same column header reverses the table.

The heart icon to the top left indicate whether the chosen terminal is a user favorite or not. If the heart is filled, the terminal is a user favorite. Clicking on the icon toggles the terminal favorite, causing it to disappear or appear in the left hand side slide menu containing user favorites. See image 5.

Clicking a horizontal arrow in the leftmost column in the table displays details about the wagons related to the train row. The wagon data is
displayed in a transparent table row underneath the train row. This is a feature intended for future use since no individual wagon data is available in any SCA database.

Image 5: User interface with search results, favorite menu to the left and filters on the right.

The right hand side slide menu contains the date filter and the terminal filter. The date filter allows the user to select dates from pop-up calendars. When one or two dates are added, the table is filtered with the chosen dates. The terminal filter allows the user to select trains loaded at one or more terminals and unloaded at one or more other terminals. When either one or more loading terminals are chosen, the user can click the “visa tåg” button to show relevant trains. Selecting no unloading terminal(s) shows all the trains departing from the selected loading terminal in the table.
4.3.3 Technical description

The single page application was built using HTML for element structure and CSS for styling. The only user interface component not made from scratch is the JQuery Datepicker [44] used to select dates for train filtering.

Diagram 1: Model relations in front-end application.

All front-end-logic is made with Ember.js. There are four defined data models: User, Terminal, Train and Wagon. The relation between these can be seen in diagram 1. The 1 – N relation between User and Terminal indicated user favorites. If the Terminal belongs to a User it is considered the users favorite. A Terminal instance can exist without a related User object. A Train is said to belong to a Terminal instance if and only if the trains “loading” property is the same as the Terminals “name”
property. Trains loaded at terminal A belongs to terminal A. A Train can have many Wagons.

When the user selects a terminal or sets up a filter, query parameters is entered in the browsers URL. The Ember.js controller for the terminal route reacts to these changes and acts accordingly. The reason to use query parameters is that the user can save the URL and use it in other places.

![Diagram 2: Filter hierarchy. Dotted lines indicates observed properties](image)

When the application is loaded and one or more query parameters is entered, observers in an Ember.js controller react and loads the proper data using the Ember.js RESTAdapter which in turn uses AJAX (see chapter 2.3.5) from the store into an array containing all the trains with relations to the selected terminal(s). This array is observed by a layered
structure of filters where each filter observes user inputs and the layer below. The filtered result is rendered into the visible table. See diagram 2.

4.4 Back-end implementation

This chapter describes the back-end implementation. Chapter 4.4.1 explains the REST API and chapter 4.4.2 describes the database.

4.4.1 REST API

One part of the back-end application is the REST API which serves the front-end with data. The API format is {rooturl}/api/{controller}/{argument}. The argument is optional. Similar to the front-end, the back-end also uses models and controllers but is omitting the view part. Controller classes inheriting from the standard ApiController class serves as the REST API. The controller classes must be named correctly. The controller contains methods matching the HTTP verb. For example, GET {rooturl}/api/terminal matches the GET method returning an array type in the TerminalController class. The controller loads data using Entity Framework objects and returns JSON data formatted meeting the Ember.js RESTAdapter expectations. Details covering the API can be seen in appendix B.

A special XMLController listens for new train data pushed from the Learningwell cloud. The controller contains a single method accessible via the API which parses and validates received train data using XML Schemas provided by Learningwell and finally saves it to the database. Data is received on the format described in chapter 2.4. Before saving it to the database, the SGLN number must be mapped to a terminal in the database.

Entity Framework models similar to the Ember.js models are used to build a relational database residing on an SQL Server running in the background. The actual database is auto-generated using the Entity Framework models.

4.4.2 Database schema

The generated relations are similar to the models Ember.js uses with some exceptions. See diagram 3.
Diagram 3: Database overview

The database has a many-to-many relationship between User and Terminal where the front-end application had a one-to-many relation. The front-end application handles one user at a time while the database has to store several users and their favorites. The terminal entity contains the SGLN (see chapter 2.2.1) number mapped to the terminal.
5 Results

In this chapter results from the used method are presented. Chapter 5.1 holds the RFID data connection examination results. The application implementation is described in chapter 4. Chapter 5.2 shows graphs from rendering tests of the front-end application and data push test results of the back-end application.

5.1 Wireless Internet service provider measurements results and coverage map comparison

The wireless service provider measurements described in chapter 3.1.2 have shown Telia to be a reliable network service provider at all sites while Net1 proved to be unavailable or unreliable for some sites. Net1 map data [45] does not give bandwidth information. There is no way to read any numbers from the map. Signal strength is divided into three types – Basic, Good and Very good. There are no explanations of the rating classes and what Signal-to-Noise Ratio the intervals correspond to. No information regarding bandwidth estimation is given. Normal bandwidth is claimed to be 1 – 7 Mbit/s [46]. Telia provide map data [47] with bandwidth estimation for 4G, 3G and 2G network types. Signal strengths are divided into the same three categories as Net1 uses. Table 4 contains measured data on a site where the RFID station was surrounded by open area while table 5 contains data on a site where the station was surrounded by woods. Appendix A contains all measured network availability and bandwidth for each geographical location compared to estimated bandwidth read from coverage maps provided by each service provider.

Terminal Östrand

<table>
<thead>
<tr>
<th></th>
<th>Telia 4G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean downlink speed</td>
<td>22,6982</td>
<td>7,1608</td>
</tr>
<tr>
<td>(Mbit/s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean uplink speed</td>
<td>7,3926</td>
<td>1,2352</td>
</tr>
<tr>
<td>(Mbit/s)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Mean response time (ms) | 40 | 233
Signal power range (dBm) | (-68) – (-69) | (-63) - (-63)

Table 4: Östrand data

Terminal Hoting

<table>
<thead>
<tr>
<th>Telia 4G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean downlink speed (Mbit/s)</td>
<td>22,0872</td>
</tr>
<tr>
<td>Mean uplink speed (Mbit/s)</td>
<td>13,0464</td>
</tr>
<tr>
<td>Mean response time (ms)</td>
<td>23</td>
</tr>
<tr>
<td>Signal power range (dBm)</td>
<td>(-91)–(-94)</td>
</tr>
</tbody>
</table>

Table 5: Hoting data

No Net1 connection was available at the Hoting terminal. Two of three terminals with woods surrounding the RFID station had no Net1 signal. The rest of the terminals had Net1 signal. Telia signal was available at all terminals.

As a result, four Conel LR77 v2 routers were purchased for installation. The LR77 v2 fulfills the requirement of OpenVPN support set by Learningwell. The router supports 4G network types with 3G and 2G fallback [48].

5.2 Verification results

This chapter shows the verification results using the method described in chapter 3.2.2 and chapter 3.2.3.
5.2.1 Rendering results

The results from the rendering test can be seen in diagram 4 and diagram 5.

**Diagram 4: Rendering times when the application is refreshed**

The number of rendered rows plotted against the rendering time. The x axis shows the rendered amount of rows while the y axis shows the render time in seconds.
Based on the Mid Sweden University template for technical reports, written by Magnus Eriksson, Kenneth Berg and Mårten Sjöström.

Results
2015-07-08

Gabriel Holmlund

Diagram 5: Rendering times when navigating to a terminal with train data without refreshing

As seen in the diagrams, the lines are fairly linear.
5.2.2 Data push results

The results from the data push tests can be seen in the following table:

<table>
<thead>
<tr>
<th>POST data</th>
<th>Visible within 15 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 wagons, 1 request</td>
<td>Yes</td>
</tr>
<tr>
<td>30 wagons, 1 request</td>
<td>Yes</td>
</tr>
<tr>
<td>15 wagons, 2 requests</td>
<td>Yes</td>
</tr>
<tr>
<td>30 wagons, 2 requests</td>
<td>Yes</td>
</tr>
<tr>
<td>15 wagons, 3 requests</td>
<td>Yes</td>
</tr>
<tr>
<td>30 wagons, 3 requests</td>
<td>Yes</td>
</tr>
<tr>
<td>15 wagons, 4 requests</td>
<td>Yes</td>
</tr>
<tr>
<td>30 wagons, 4 requests</td>
<td>Yes</td>
</tr>
<tr>
<td>15 wagons, 5 requests</td>
<td>Yes</td>
</tr>
<tr>
<td>30 wagons, 5 requests</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 6: Test scenario and results
6 Conclusions

This chapter holds the conclusions of the thesis work. Chapter 6.1 covers the examination of different network type alternatives for the RFID stations. Chapter 6.2 covers conclusions made when implementing the application and chapter 6.3 discusses the results gotten from solution verification. Chapter 6.4 holds ideas for future work.

6.1 RFID data connection alternatives

Examining the data connection alternatives for the RFID stations proved to be a fairly quick process. All involved parties agreed on the wireless network service provider option. The time-consuming part was to determine which service provider to use since it required a visit to all the railway terminals where RFID stations are installed or will soon be installed. The coverage maps provided by the wireless Internet service providers proved to be a bad way to determine the choice of service provider. In the end, Telia were chosen as the desired service provider. Since the LTE network is constantly growing, the Telia choice is more stable for the future. It is surprising that the Net1 alternative proved to be an unreliable service provider given the fact that they use lower frequencies and claim to have the best coverage in Sweden. There can be several reasons to why Telia proved to be the best service provider. Telia base stations might have been closer to the sites than Net1s’. Telia 4G were at some point available on all sites except one. This can mean that the channel access methods used in LTE is more suitable for non-clear environments such as woods. There is a higher probability that the Net1 station is farther away since their technology uses the 450 MHz band with better coverage per cell. If the signal travels a longer distance through obstructed areas, the probability of multipath propagation is higher than a communicating with a cell close by. A modem with more powerful antennas or a higher placement of the antenna might have improved the result.

A set of 4G modems were bought and should have been configured by Learningwell staff. However, the 4G modems were never configured properly due to unexpected negotiation delays between Learningwell and SCA. The modems were never installed in the RFID terminals due to this delay. Axle counter calibration necessary for correct RFID data
also suffered from this since it was supposed to be done in conjunction with modem installation. This means that the goal of examining and realizing a data connection with the RFID stations was partially reached. However, the delays did not cause problems while developing the application and did not impair the development schedule since the SCA application is only dependent on the train data feed provided by Learningwell. While developing, mock data using the same format the train data feed was used for testing.

### 6.2 Implementation conclusions

The Ember.js/ASP.NET implementation proved to be good. Most of the time spent on this thesis was implementing the front-end application. Ember.js proved to be more difficult to understand than expected. The Ember.js documentation was in some cases for older versions and made certain articles useless. Suggestions from Internet communities could also aim at older framework versions. Another JavaScript SPA framework such as Angular.js could have been less time-consuming. Implementing the ASP.NET back-end was not as time-consuming. The documentation was extensive and detailed. The community also proved to be large and Internet searches of an issue almost always came up with an answer either from the documentation or from the community. The resulting application was appreciated by SCA representatives. However, since the modems never got installed in time the goal of implementing a full solution is not entirely fulfilled. A full solution relies on data pushes from the Learningwell service. Besides modem installation, there must exist a mapping between the RFID stations SGLN number and a terminal name. These numbers were mocked when testing the application. The SGLN numbers can be obtained when installing the modems.

### 6.3 Application verification

The application verification results show that the application performs as required. Measuring rendering times showed that render times when initially loading the application is higher than when navigating the site. This can be expected since Ember.js has to set up entire site with underlying functionality. The higher load time when the application is initially loaded is acceptable since the application does not need to be reloaded to update the train data. Both graphs showed a linear tendency when the number of rows were increased. The application does not set any maximum row limit but it is unlikely for a user to load more than 60
rows at a time. The render times when navigating the site is low enough. The user experience might vary with different network conditions. If the network is slow it will cause train data to have long loading times. Sending parallel data pushes to the back-end did not seem to have much effect on results. The XML validation and database transactions are fast enough to handle several connections at the same time. These results are also depending on network conditions and the server hardware.

6.4 Future work

Both the front-end application and the back end-application can be improved. The front-end application can be extended with more filters and more efficient memory management. It can also be equipped with several statistics views showing different calculations. For example, showing how many wagons were lost in a month or the total number of wagons transported between the users favorite terminals. The application can also be extended to support several users with different access rights. When individual wagon data is available in some SCA database this information can be inserted in the application to show cargo type and volume. The back-end application could be fitted with a more extensive API to send already filtered data given some input parameter. It could also be equipped with WebSockets to enable pushing of data to the front-end. This would allow the tables to be updated without manually refreshing the table. The entire system can be merged with SCAs current transport system. This will probably be done by SCAs IT consultants. Having two separate applications doing similar work is unnecessary.
References

[1] SCA, “2.6 million hectares of forestland” [www]
Retrieved 2015-01-23

Retrieved 2015-01-23

http://transportnet.se/nyheter/trafikverket-rfid-prisade-igen/
Retrieved 2015-01-24

http://www.pts.se/cookies
Retrieved 2015-03-13


[6] Impinj, “The different types of RFID systems” [www]
http://www.impinj.com/resources/about-rfid/the-different-types-of-rfid-systems/
Retrieved 2015-02-02

http://www.eecs.harvard.edu/cs199r/readings/RFID-article.pdf
Retrieved 2015-06-21

[8] Clinton S. Hartmann and Lewis T. Claiborne, RF SAW, Inc., Richardson, Texas 75081
“Fundamental Limitations on Reading Range of Passive IC-Based and SAW-based RFID”
2007 *IEEE International Conference on RFID*
References

2015-07-08

[9] Abdelmoula Bekkali, Member IEEE, Sicheng Zou, Student Member, IEEE, Abdullah Kadri, Member, IEEE, Michael Crisp, Member, IEEE, and Richard V. Penty, Senior Member, IEEE “Performance Analysis of Passive UHF RFID Systems Under Cascaded Fading Channels and Interference Effects” IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, VOL 14, NO.3, MARCH 2015


References

2015-07-08

[16] 3GPP, “Overview of 3GPP Release 5 v0.1.1 (2010-02)” [www]
http://www.3gpp.org/ftp/Information/WORK_PLAN/Description_Releases/Rel-05_description_20100214.zip
Retrieved 2015-07-01

[17] 3GPP, “Overview of 3GPP Release 6 v0.1.1 (2010-02)”
http://www.3gpp.org/ftp/Information/WORK_PLAN/Description_Releases/Rel-06_description_20100214.zip
Retrieved 2015-07-01

http://www.evdoinfo.com/content/view/626/64/
Retrieved 2015-07-01

http://www.3gpp.org/ftp/Information/WORK_PLAN/Description_Releases/Rel-08_description_20140924.zip
Retrieved 2015-07-01

[20] 3GPP, “Overview of 3GPP Release 10 v0.2.1 (2014-06)” [www]
http://www.3gpp.org/ftp/Information/WORK_PLAN/Description_Releases/Rel-10_description_20140630.zip
Retrieved 2015-07-01

http://www.w3schools.com/xml/xml_whatis.asp
Retrieved 2015-02-10

http://www.w3.org/TR/xmlschema-0/
Retrieved 2015-02-10

Retrieved 2015-02-10

json-schema-core” [www]
http://json-schema.org/latest/json-schema-core.html
Retrieved 2015-02-10
References
2015-07-08


[27] Google, “MVC Architecture” [www]
https://developer.chrome.com/apps/app_frameworks
Retrieved 2015-02-12

[28] Ember, “Bindings” [www]
http://guides.emberjs.com/v1.10.0/object-model/bindings/
Retrieved 2015-05-26

[29] Ember, “Handlebars basics” [www]
http://guides.emberjs.com/v1.10.0/templates/handlebars-basics/
Retrieved 2015-05-28

[30] Ember, ”Naming conventions” [www]
http://guides.emberjs.com/v1.11.0/concepts/naming-conventions/
Retrieved 2015-05-28

[31] Ember, ”Defining your routes” [www]
http://guides.emberjs.com/v1.10.0/routing/defining-your-routes/
Retrieved 2015-05-28

http://guides.emberjs.com/v1.11.0/models/
Retrieved 2015-05-28

[33] Ember, “The REST adapter” [www]
http://guides.emberjs.com/v1.11.0/models/the-rest-adapter/
Retrieved 2015-05-28

[34] ASP.NET, “Hands On Lab: Build a Single Page Application (SPA) with ASP.NET Web API and Angular.js” [www]
http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/build-a-single-page-application-(spa)-with-
References

2015-07-08
Gabriel Holmlund

aspnet-web-api-and-angularjs
Retrieved 2015-05-05

[35] Microsoft developer network, “Code First to a New Database” [www]
https://msdn.microsoft.com/sv-se/data/jj193542
Retrieved 2015-05-23

[36] GS1, “EPC Information Services” [www]
http://www.gs1.org/sites/default/files/docs/epc/epcis_1_0_1-
standard-20070921.pdf
Retrieved 2015-03-17

[37] Trafikverket “Nätkapacitetstjänster” [www]
http://www.trafikverket.se/Foretag/Natkapacitetstjanster/
Retrieved 2015-05-21

[38] AINMT, ”This is AINMT”, [www]
http://www.net1.se/en/
Retrieved 2015-02-24

Retrieved 2015-05-20

[40] Net1, “Trådlöst modem R-90” [www]
http://www.net1.se/privat/kundservice/teknisk-support--
installation/Modem/tradlost-modem-gateway-r-90/
Retrieved 2015-05-28

[41] Sublime Text, “Sublime Text 2” [www]
http://www.sublimetext.com/2
Retrieved 2015-06-21

https://www.visualstudio.com/
Retrieved 2015-06-21

[43] REST Commander, “REST Commander” [www]
http://www.restcommander.com/index.html
Retrieved 2015-05-20
[44] JQuery, “Datepicker” [www]
http://jqueryui.com/datepicker/
Retrieved 2015-05-13

[45] Net1, “Täckningskarta” [www]
http://www.net1.se/privat/tackning/tackningskarta/
Retrieved 2015-04-24

[46] Net1 “ALLMÄN INFORMATION OM TÄCKNING” [www]
http://www.net1.se/privat/tackning/allman-information-om-tackning/
Retrieved 2015-04-24

http://www.telia.se/privat/support/tackningskartor
Retrieved 2015-04-24

[48] Conel, “4G LTE ROUTER CONEL LR77 V2” [www]
http://www.conel.com/4g-lte-router-lr77-v2/
Retrieved 2015-04-14
Appendix A: Network tables

This appendix contains tables presenting network measurement data and predicted bandwidth read from coverage maps. A short comment regarding the measurements for each site is given.

Töva - Measurements

<table>
<thead>
<tr>
<th></th>
<th>Telia 4G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean downlink speed</td>
<td>22,5342</td>
<td>1,9012</td>
</tr>
<tr>
<td>(Mbit/s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean uplink speed</td>
<td>7,1002</td>
<td>0,475</td>
</tr>
<tr>
<td>(Mbit/s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean response time</td>
<td>26</td>
<td>90</td>
</tr>
<tr>
<td>(ms)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal power range</td>
<td>(-71) – (-75)</td>
<td>(-94) - (-97)</td>
</tr>
<tr>
<td>(dBm)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The 4G connection was stable and never fell back to 3G or EDGE during the measurements. The Net1 connection was also stable.

Töva – Map data

<table>
<thead>
<tr>
<th></th>
<th>Telia 2G</th>
<th>Telia 3G</th>
<th>Telia 4G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal strength</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Good</td>
</tr>
<tr>
<td>Bandwidth estimation</td>
<td>Up to 100 Kbit/s</td>
<td>2-10 Mbit/s</td>
<td>10-40 Mbit/s</td>
<td>x</td>
</tr>
</tbody>
</table>
### Östrand - Measurements

<table>
<thead>
<tr>
<th></th>
<th>Telia 4G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean downlink speed</td>
<td>22,6982</td>
<td>7,1608</td>
</tr>
<tr>
<td>(Mbit/s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean uplink speed</td>
<td>7,3926</td>
<td>1,2352</td>
</tr>
<tr>
<td>(Mbit/s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean response time</td>
<td>40</td>
<td>233</td>
</tr>
<tr>
<td>(ms)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal power range</td>
<td>(-68) – (-69)</td>
<td>(-63) - (-63)</td>
</tr>
<tr>
<td>(dBm)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The 4G connection was stable and never fell back to 3G or EDGE during the measurements. The Net1 connection was also stable.

### Östrand – Map data

<table>
<thead>
<tr>
<th></th>
<th>Telia 2G</th>
<th>Telia 3G</th>
<th>Telia 4G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal strength</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Very Good</td>
</tr>
<tr>
<td>Bandwidth estimation</td>
<td>Up to 100 Kbit/s</td>
<td>2-10 Mbit/s</td>
<td>10-40 Mbit/s</td>
<td>x</td>
</tr>
</tbody>
</table>
**Hoting - Measurements**

<table>
<thead>
<tr>
<th></th>
<th>Telia 4G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean downlink speed</td>
<td>22,0872</td>
<td>x</td>
</tr>
<tr>
<td>(Mbit/s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean uplink speed</td>
<td>13,0464</td>
<td>x</td>
</tr>
<tr>
<td>(Mbit/s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean response time</td>
<td>23</td>
<td>x</td>
</tr>
<tr>
<td>(ms)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal power range</td>
<td>(-91)-(-94)</td>
<td>x</td>
</tr>
<tr>
<td>(dBm)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

No Net1 signal was available.

**Hoting – Map data**

<table>
<thead>
<tr>
<th></th>
<th>Telia 2G</th>
<th>Telia 3G</th>
<th>Telia 4G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal strength</td>
<td>Very Good</td>
<td>Good</td>
<td>Very Good</td>
<td>Good</td>
</tr>
<tr>
<td>Bandwidth estimation</td>
<td>Up to 100 Kbit/s</td>
<td>2-10 Mbit/s</td>
<td>10-40 Mbit/s</td>
<td>x</td>
</tr>
</tbody>
</table>
Krokom - Measurements

<table>
<thead>
<tr>
<th></th>
<th>Telia 4G</th>
<th>Telia 3G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean downlink speed</td>
<td>24,5616</td>
<td>7,436</td>
<td>1,4014</td>
</tr>
<tr>
<td>speed (Mbit/s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean uplink speed</td>
<td>8,6882</td>
<td>1,0718</td>
<td>0,1178</td>
</tr>
<tr>
<td>speed (Mbit/s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean response time</td>
<td>23</td>
<td>68</td>
<td>168</td>
</tr>
<tr>
<td>time (ms)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal power range</td>
<td>(-100)-(-101)</td>
<td>(-94) – (-96)</td>
<td>(-97) – (-99)</td>
</tr>
<tr>
<td>(dBm)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After measuring 4G the tablet fell back to 3G. Measurements for 3G were also made. The Net1 R-90 modem lost connection several times and made the measuring process difficult.

Krokom – Map data

<table>
<thead>
<tr>
<th></th>
<th>Telia 2G</th>
<th>Telia 3G</th>
<th>Telia 4G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal strength</td>
<td>Very Good</td>
<td>Basic</td>
<td>Basic</td>
<td>Good</td>
</tr>
<tr>
<td>Bandwidth estimation</td>
<td>Up to 100Kbit/s</td>
<td>2-10 Mbit/s</td>
<td>10-40 Mbit/s</td>
<td>x</td>
</tr>
</tbody>
</table>
Bensjö - Measurements

<table>
<thead>
<tr>
<th></th>
<th>Telia 3G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean downlink speed</td>
<td>12,4936</td>
<td>x</td>
</tr>
<tr>
<td>(Mbit/s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean uplink speed</td>
<td>3,9836</td>
<td>x</td>
</tr>
<tr>
<td>(Mbit/s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean response time</td>
<td>54</td>
<td>x</td>
</tr>
<tr>
<td>(ms)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal power range</td>
<td>(-80)-(-82)</td>
<td>x</td>
</tr>
<tr>
<td>(dBm)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4G was not available but 3G provided good signal. No Net1 signal was available.

Bensjö – Map data

<table>
<thead>
<tr>
<th></th>
<th>Telia 2G</th>
<th>Telia 3G</th>
<th>Telia 4G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal strength</td>
<td>Very Good</td>
<td>Basic</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Bandwidth estimation</td>
<td>Up to 100 Kbit/s</td>
<td>2-10 Mbit/s</td>
<td>10-40 Mbit/s</td>
<td>x</td>
</tr>
</tbody>
</table>
Östavall - Measurements

<table>
<thead>
<tr>
<th></th>
<th>Telia 4G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean downlink speed (Mbit/s)</td>
<td>38,9272</td>
<td>3,472</td>
</tr>
<tr>
<td>Mean uplink speed (Mbit/s)</td>
<td>19,9474</td>
<td>0,8932</td>
</tr>
<tr>
<td>Mean response time (ms)</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>Signal power range (dBm)</td>
<td>(-76)-(-78)</td>
<td>(-65)-(-65)</td>
</tr>
</tbody>
</table>

Both network types provided good signal.

Östavall – Map data

<table>
<thead>
<tr>
<th></th>
<th>Telia 2G</th>
<th>Telia 3G</th>
<th>Telia 4G</th>
<th>Net1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal strength</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Very Good</td>
</tr>
<tr>
<td>Bandwidth estimation</td>
<td>Up to 100 Kbit/s</td>
<td>2-10 Mbit/s</td>
<td>10-40 Mbit/s</td>
<td>x</td>
</tr>
</tbody>
</table>
Appendix B: REST API description

This appendix gives a detailed description of the API and how it is used.

The API is divided into two parts – one part used by the front-end application and one part used by the Learningwell service to push new train data. Below is a list of valid REST requests and example return data.

**GET {rooturl}/api/terminals** :

```
{
  "terminals" : {
    "id" : 1,  //terminal id
    "name" : "HOTING",  //Terminal name
    "trains" : [1,2,3,4]  //Array of associated train id’s
  },
  {
    "id" : 2,
    "name" : "KROKOM",
    "trains" : [5,6,7,8]
  },
  //more entries...
}
```

**GET {rooturl}/api/train/:id** :

```
{
  "train" : {
    "terminal" : 1,  //associated terminal id
    "loading" : "HOTING",  //name of associated terminal
    "dateTimeIn" : "2015-02-21T17:43:11",  //date and time when the empty
    //train entered the loading terminal
    "dateTimeOut" : "2015-02-22T06:52:11",  //date and time when the
    //loaded train exited the loading terminal
    "wagonIn" : 30,  //number of wagons attached to the empty train when
    //entering the loading terminal
    "wagonOut" : 29,  //number of full wagons when exiting the loading
    //terminal
  }
}
“unLoading”: “TÖVA”, //name of the unloading terminal
“unDateTime”: “2015-02-22T10:02:12”, //date and time when the train entered the unloading terminal
“unWagon”: 28 //number of wagons when entering the unloading terminal
// “wagons”: [1,2,3,4] - not included yet, intended for future usage.
}

GET {rooturl}/api/user/:id :
{
  “user”: {
    “name”: “Lars Stefansson”, //users full name
    “favorites”: [1,2,3,4,5,6] //id’s of favorite terminals
  }
}

GET {rooturl}/api/wagon/id :
Intended for future use.

POST {rooturl}/api/xml :
The input request body data has the format explained in chapter 2.x
Returns HTTP status 200 if successful, otherwise HTTP status 500.