university of groningen

faculty of mathematics and natural sciences

# Advanced Software Architecture

Train ticketing system

Aida Baxhaku (3242528)

Petar Hariskov (3226735)

Alexandra Matreata (3179265)

Eric Rwemigabo (3040356)

January 11, 2017

version 1.0.0

# Revision history

# Contents

# List of figures

# List of tables

# Glossary

**ET** Easy Ticketing

# Chapter 1

# Introduction and goals

## 1.1 Introduction

Traveling by train is one of the most common means of transport in the Netherlands and the Western Europe. The majority of people use trains on a daily basis, to go to school, work, etc. It is quite easy and comfortable, but one common problem, that everyone may relate to is the discomfort of train tickets. The printed tickets might get lost, or a passenger might forget to check in, another one is too late to buy one, etc.

However, we have come up with an alternative train ticket, which will facilitate the journey of many passengers: Easy Ticketing (ET). This is a new, promising and innovative technology that suits practically everyone who owns a smartphone. The idea behind ET is very simple: you download an app in your smartphone, open your personal account and connect it to your favorite payment method, and ET will do everything else automatically.

This means, every train will have beacons, which will connect to the passenger's phones via bluetooth, it will check the passenger in, it will automatically calculate the ticket fee and it will reconnect when the passenger leaves the train. There will be a beacon per every gateway. The beacon can detect the mobile in a diameter of 20m.

The aim of this new technology is to facilitate the journey of train passangers, and take the ticket purchasing to the next level. We plan to implement ET firsly in Groningen and after the initial success the aim of our company is to cover the whole Netherlands.

In this document we are presenting the architecture of our system. In the next chapters you will be introduced to the Stakeholders of the systems and their concerns, the requirements, functional and non-functional. In

section 3, we will briefly analyze the business context and the benefits that ET will brings in terms of business. In section 4, the solution strategy is presented, in terms of our main key drivers. Section 5 outlines the building block view, whereas chapter 6 and 7 introduce the runtime and deployment view. The architecture of our system will be concluded with the design decisions and quality scenarios.

## 1.2 Quality goals

The top three goals of the architecture of Easy Ticketing whose fulfillment is of highest importance to the major stakeholders (as agreed between them) are listed below:

**Security** In our system, we could view the importance of security in two major aspects: Firstly, hackers might hack the system and steal money from the account of the train company and secondly, passengers might find a way to fake their tickets. In both cases, this is unacceptable for our customer under any circumstance. Therefore the system must be designed in such way that potential data leaks, which are in theory inevitable, will not result in access to the main database or create damage to the company.

**Reliability** Customers are expected to rely on the functioning of Easy Ticketing. Problems with availability of the service are almost as undesirable as security issues, therefore availability must be taken into account throughout the whole system architecture. Breakdown of the system could lead to financial loss for the company, which is in any case not acceptable.

**Compatibility** Is an important driver for our system. Passengers may be in possession to different smartphones, which may result in incompatibility with the beacon, incompatible frequencies of bluetooth. It is very important to provide the passengers the possibility to purchase the ticket, for as many phone versions as possible.

# Chapter 2

# Requirements

In this section the main Stakeholders and the functional and non functional requirements of Easy Ticketing will be introduced.

## 2.1 Stakeholders

The following stakeholders are described along with their concerns. These concerns eventually determine the key drivers of the service.

### 2.1.1 Represented stakeholders

**Passengers** are the direct users of Easy Ticketing. The typical target customer is the traveler who uses the train systems regularly. Passengers are mainly concerned with the reliability and the availability of the system as well as the security often their data and the regular payment of the ticket. If the service is not available then they risk not having a ticket and eventually getting a fine. Another concern of the passengers could be the incompatibility of their phone with the beacon, which could risk the validity of their ticket.

**Customers** are the owners of the Easy Ticketing. The typical target are the large train companies, operating in the Netherlands. The companies are mainly concerned with the reliability of the system, since many passengers could end up not paying, given that there is a breakdown of the system. This could result in monetary loss for the company. Another main concern of our customer is the security. If the system is hacked the company loses money and this is certainly something that the company wants to avoid.

**Developers and maintenance team** is responsible for building the

software part of our system, test and debug it afterwards. They are mainly concerned with the maintainability, testability and portability of the system.

**Architects** will be the team that will design the system. Their main concern is satisfying the needs of all the stakeholders and finding the best solution. They determine the feasibility of desired properties and functions, and guarantee a satisfactory end-product.

### 2.1.2 Non represented stakeholders

**Hardware companies** are the companies that provide beacons for our system. In our case they are non represented stakeholders, since they do not have much say

**Hardware maintenance team** maintains the physical side of the system, in our case beacons and takes care of handling malfunctions.

**Ticket inspectors** are the employees of the Train companies, that check the passengers, whether they have an available ticket or not. The are mostly concerned with the security and the reliability of the system.

**Third party payment systems** are companies that provide a paying system for the passengers.

## 2.2 Functional requirements

| | | |
|---|---|---|
| FR-1 | Must | There should be at least 2 different ways of payment. |
| FR-2 | Must | The beacons shall connect to a phone's bluetooth and application. |
| FR-3 | Must | The system MUST/WILL/HAVE TO charge a person based on account/travel location. |
| FR-4 | Must | The system should verify which cart sends which data (or train actually). |
| FR-5 | Must | Phones should be verified based on accounts. |
| FR-6 | Must | The system should provide a back-up to log out of the trip in case battery dies via touch-screen panels in stations. |
| FR-7 | Must | The beacons will start connecting to mobile devices once the train is 300m away from the station. |
| FR-8 | Must | Phone application should have beacon verification. |
| FR-9 | Must | The train will collect data from the carts/beacons and send them to the server after leaving a station. |
| FR-10 | Must | The system should charge on trip exit. |
| FR-11 | Must | Once a connection between a beacon and a phone is established, the system must send a notification to the user's phone. |
| FR-12 | Must | Application provides a user interface enabling the user to check his account, update amounts. |
| FR-13 | Must | The beacons register both logged in and not logged in users and match the account with the phone even if the user logs in later on during the journey. |

Table 2.1: Functional requirements

## 2.3 Non-functional requirements

### 2.3.1 Security

| | | |
|---|---|---|
| NF-1.1 | Must | One account is allowed to login at one smartphone at one time. |
| NF-1.2 | Must | Users will need to authenticate in order to login. |
| NF-1.3 | Must | The data information provided by the costumers will be safe and secure. |
| NF-1.4 | Must | Every request from a subserver component to the main repository must be verified so that the repository knows at all times who sent the request. |

### 2.3.2 Reliability

| | | |
|---|---|---|
| NF-1.1 | Must | The databases must be available 99.9999% of the time during a train journey. |
| NF-1.2 | Must | The system must be fault tolerant. |
| NF-1.3 | Must | The beacons inside the carts will be operational at all times when the train is working. |
| NF-1.4 | Must | The Server/database will be 99.999% available , with backups . |

### 2.3.3 Compatibility

| | | |
|---|---|---|
| NF-1.1 | Must | The system should be compatible with the 3 main os (ios, android, windows). |
| NF-1.2 | Must | Compatible frequencies. |

# Chapter 3

# System scope and context

This section outlines the main relations between the system and its environment, external systems or entities with which it interacts. The business and technical contexts in which the system will perform as well as different inter-component communication solutions will be presented in this chapter.

## 3.1 Business context

The system focuses on simplifying the management of information related to train traveling and ticket payment for its users. The system will provide an account for each user which will store information regarding travel distance, destinations, payments, etc. This will help both users travelling by train and the companies providing the travelling services by keeping track of all these components in a centralized manner.

The system will present the user with the choice of making an automated payment for each trip through the connection between the mobile app and the beacon in the train cart. Alternatively, an external payment system will be presented to the user I every train station where they can log in and perform the transaction.

A second important target for our system will be represented by companies providing travelling services by train, which may include governmental institutions or different private companies.

## 3.2 Logical view

The following diagram displays the main use cases of the system which will be further described in use case scenarios:

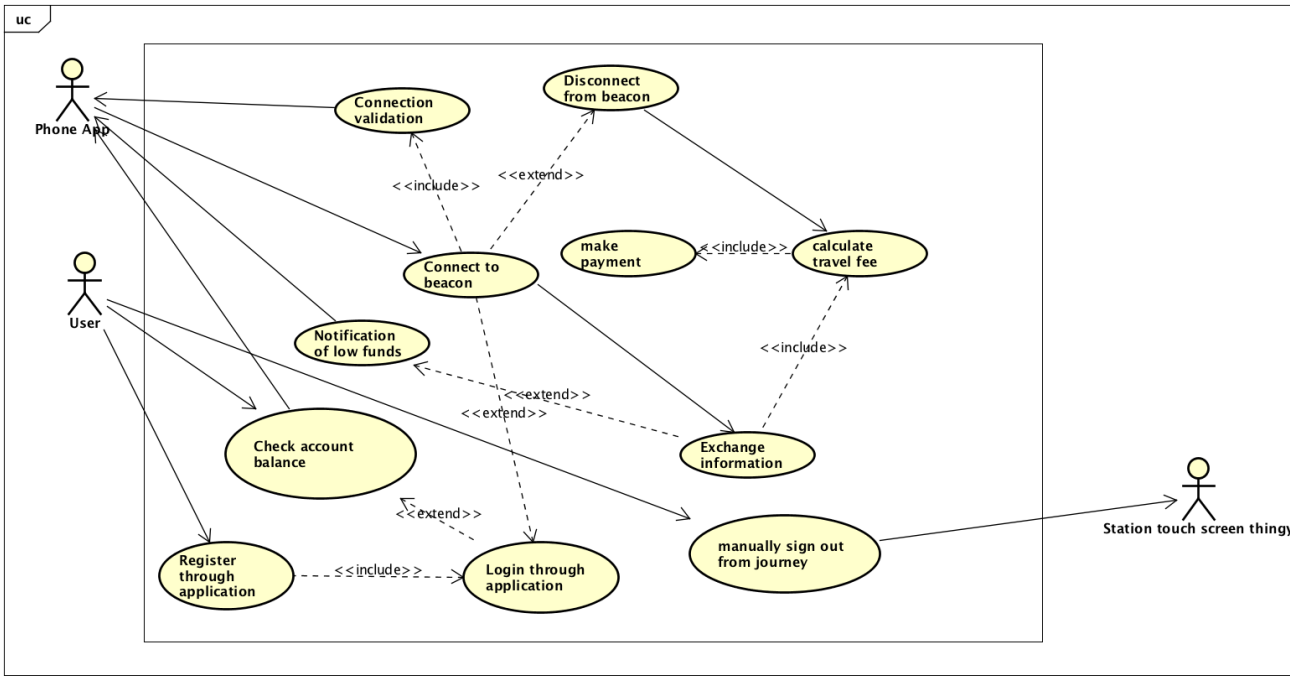


Figure 3.1: Main use cases

| | |
|---|---|
| **Usecase** | Connect phone to beacon |
| **Main actor** | Phone app |
| **Goal** | Connect to the beacon |
| **Preconditions** | • Bluetooth is switched on<br>• App is installed<br>• Train has left the station |
| **Main scenario** | • The beacon discovers the phone<br>• The phone receives a notification from the beacon<br>• The beacon saves the account information locally |
| **Exception scenarios** | • The app doesn't get the notification because the phone is off<br>• The app doesn't get the notification, due to a communication failure<br>• The beacon fails to detect the phone<br>• The app is installed but the owner doesn't have an account |
| **Postconditions** | • Phone is connected<br>• Information exchange between phone and beacon |
| **Related requirements** | NFR2, FR2, NFR3 |

Table 3.1: Main use cases -

| | |
|---|---|
| **Usecase** | Disconnect from beacon |
| **Main actor** | Beacon |
| **Goal** | Register a phone disconnection |
| **Precondition** | • Bluetooth is switched on<br>• Use case 1 |
| **Main scenario** | • The user leaves the train<br>• The beacon checks available connections<br>• The beacon sends gathered information to local server<br>• Local server sends information to main server<br>• Main server computes distance travelled by user |
| **Exception scenarios** | • Phone battery dies before user leaves the train<br>• The beacon malfunctions and cannot check for connections |
| **Postconditions** | Travel distance and price are computed |
| **Related requirements** | NFR3, FR3, FR10, FR9 |

Table 3.2: Main use cases - use case 2

| | |
|---|---|
| **Usecase** | Calculate travel fee |
| **Main actor** | Server |
| **Goal** | Charge user for travelled distance |
| **Precondition** | • Usecase 1<br>• Usecase 2 |
| **Main scenario** | • Main server receives information regarding users which are still connected<br>• Main server compares accounts currently connected with accounts connected for previous stop<br>• Users which were connected previously and are not connected anymore are identified<br>• Travel distance and fee are calculated for identified users |
| **Exception scenarios** | • Information cannot be transmitted between local and main servers<br>• Account is not identified |
| **Postconditions** | • Notification regarding payment is sent to user<br>• Payment is made |
| **Related requirements** | FR3, FR9, FR10, FR13 |

Table 3.3: Main use cases - use case 3

## 3.3 Technical context

The system will need to perform in a specifically designed technical context consisting of three main environments: a central server collecting all information and performing necessary computations, the setup inside the trains themselves (comprised of beacons and small servers which will collect information per train and send it to the central one every time the train leaves the station) and a login system in each station allowing users to manually check-out of their trip and perform payment.

A set of measures will need to be taken into consideration for the system to be able to operate inside this context by using these different components and environments. These measures will be related to the key attributes required of the system as follows:

- **security**: since the system will have access to sensitive information (such as location, financial transactions), every connection between components of the system will need to be secured and trusted. The main connections which will need to be verified each time they are created will be: beacon to mobile application, beacon to server inside the train, small local server to central server.

- **reliability**: for the system to be considered reliable, the main components should be provided with a back-up in case of failure, such as the central server, the beacons in each cart, etc. Also, users should be presented with alternative scenarios in case the main desired activity flow gets interrupted (e.g. possibility to manually check-out of a trip using login system in train station in case of beacon to phone communication being severed)

- **compatibility**: the system will need to be compatible with at least the main and most commonly used mobile OS and frequencies

## 3.4 External interfaces

| | |
|---|---|
| **Channel** | Beacon ⇒ Mobile app |
| **Description** | Beacons detect mobile phones in their proximity having installed the application and initialise a connection. |
| **Connection** | Bluetooth |
| **Protocol** | iBeacon |
| **Frequency** | The train has left a station and a mobile phone is in range of the beacon. |

Table 3.4: Interface - Beacon to phone

| | |
|---|---|
| **Channel** | Beacon ⇒ Local train server |
| **Description** | Beacons send information gathered every time a train departs from a station. |
| **Connection** | Bluetooth |
| **Protocol** | iBeacon |
| **Frequency** | The train has left a station. |

Table 3.5: Interface - Beacon to server

| | |
|---|---|
| **Channel** | Local server ⇒ Central server |
| **Description** | The local server situated inside the train sends information gathered periodically to the central server if an internet connection is possible. |
| **Connection** | Internet |
| **Protocol** | UDP |
| **Frequency** | Once every 20 min or when a connection is possible. |

Table 3.6: Interface - Local server to central server

# Chapter 4

# Solution strategy

This chapter aims to discuss and present a concrete solution space for the problems described in chapters **??** and 3. The solutions provided will directly follow the main key drivers of the project and focus on the problem space using views limited only to these most important attributes.

## 4.1 Security

## 4.2 Compatibility

## 4.3 Reliability

# Chapter 5

# Building block view

# Chapter 6

# Runtime view

# Chapter 7

# Deployment view

# Chapter 8

# Concepts

## 8.1 Domain model

## 8.2 Architectural patterns

## 8.3 Security

## 8.4 Compatibility

## 8.5 Reliability

# Chapter 9

# Design decisions

# Chapter 10

# Quality scenarios

# Notes

# Appendix A

# Time tracking

# Bibliography


[1] *Spring framework*. [Online]. Available: https://en.wikipedia.org/wiki/Spring_Framework (visited on 12/14/2016).

[2] *Top java web frameworks*. [Online]. Available: https://zeroturnaround.com/rebellabs/top-4-java-web-frameworks-revealed-real-life-usage-data-of-spring-mvc-vaadin-gwt-and-jsf/ (visited on 01/06/2016).

[3] P. Webb, D. Syer, J. Long, S. Nicoll, R. Winch, A. Wilkinson, M. Overdijk, C. Dupuis, and S. Deleuze, *Spring boot documentation*. [Online]. Available: https://docs.spring.io/spring-boot/docs/current (visited on 12/14/2016).

[4] *Spring boot*. [Online]. Available: https://jira.spring.io/browse/SPR-9888 (visited on 12/16/2016).

[5] *Spring boot*. [Online]. Available: https://github.com/spring-projects/spring-boot (visited on 12/16/2016).

[6] *Spring*. [Online]. Available: https://spring.io/team (visited on 12/14/2016).

[7] *Iso 25010 quality model*. [Online]. Available: https://disciplinas.stoa.usp.br/pluginfile.php/294901/mod_resource/content/1/ISO%2025010%20-%20Quality%20Model.pdf (visited on 08/19/2011).

[8] *Introduction to the spring framework*. [Online]. Available: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/overview.html (visited on 12/18/2016).

[9] *Spring boot*. [Online]. Available: https://github.com/spring-projects/spring-boot (visited on 01/06/2017).

[10] *Spring design patterns*. [Online]. Available: https://premaseem.wordpress.com/2013/02/09/spring-design-patterns-used-in-java-spring-framework/ (visited on 01/08/2017).

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design patterns abstraction and reuse of object oriented design", (visited on 01/08/2017).