



TAMPEREEN TEKNILLINEN YLIOPISTO  
TAMPERE UNIVERSITY OF TECHNOLOGY

VADIM KORSAKOV  
A DESIGN FRAMEWORK FOR ATTENDANCE TRACKING SYSTEMS  
Master of Science Thesis

Examiner: Professor Tommi Mikkonen  
Examiner and topic approved by the Council of the Faculty of Computing and  
Electrical Engineering on 5 June 2013

## II

### **ABSTRACT**

TAMPERE UNIVERSITY OF TECHNOLOGY

Master's Degree Programme in Information Technology

**Korsakov, Vadim:** A Design Framework For Attendance Tracking Systems

Master of Science 46 pages and electronic information carrier with source codes.

March 2014

Major: Software Systems

Examiner: Professor Tommi Mikkonen

Keywords: attendance tracking systems, tripwire, API

An attendance tracking system is a software-hardware based design that is used for counting the number of people who pass through its sensor for a limited period of time. Visitors are usually divided into two classes: incoming and leaving. This thesis is dedicated to developing of attendance tracking system TRIPWIRE which is currently running at Demola New Factory (Tampere location).

The system has been created for the purposes of tracking amount of visitors in Demola New Factory Tampere location. Another purpose of this thesis work was to create a customizable system which is friendly for 3rd party solutions.

Both of the above issues have been solved by implementing a client-server application. The server part is designed so that it has an API, which provides a framework for the client-side software. The current API supports statistical data collection and analysis, authentication and - different access levels for different users. Client-side software communicates with the server via HTTP-requests, and it can be implemented using any programming language which supports this type of communication. On the hardware level, TRIPWIRE is implemented as a simple 2 infrared lasers based sensor which is tracking incoming and leaving visitors. For the current moment of time (January, 2014) the system is running in a test mode at Demola Tampere location.

The thesis describes the process of creating software for the ATS TRIPWIRE from background analysis till making the working prototype, testing procedure and further possibilities for use and maintenance. It also includes system source codes published on the electronic media enclosed as an appendix.

### III

#### **PREFACE**

This thesis is a result of work on a TRIPWIRE attendance tracking system. During the work on this project I gained a lot of useful skills. Especially, it was my first experience of a client-server application development. My part of the work was related to coding of almost all the software related to the project except the hardware embedded software system which was already pre-adjusted for interoperating with the server where all the other software (except client-side software) is situated.

I am grateful to all Demola personnel and especially to Jarkko Moilanen who was the Demola-side supervisor and to my academic supervisor from TUT Dr. Tommi Mikkonen

I hope you will enjoy reading my work.

Vadim Korsakov  
Tampere, Finland  
January 2014

## IV

### TABLE OF CONTENTS

1	INTRODUCTION.....	1
2	BACKGROUND.....	3
	2.1 Applications of ATS.....	3
	2.2 Commonly used technologies and devices.....	7
	2.3 The most known ATS producers.....	15
3	TRIPWIRE DEVELOPMENT PROCESS.....	16
	3.1 The need/problem behind the project.....	16
	3.2 Project requirements.....	16
	3.3 Risk analysis and management.....	18
	3.4 Planning architecture.....	19
	3.5 An example of client software implementation.....	30
4	TESTING PROCEDURE.....	34
	4.1 Determining testing methods and setting testing environment.....	34
	4.2 User stories based testing.....	35
5	ACHIEVED RESULTS AND CONCLUSIONS.....	43
6	REFERENCES.....	45

## V

### TERMS AND DEFINITIONS

ATS	Attendance tracking system a system for counting incoming and outgoing people for some place, building or location.
API	Application programming interface.
CR	Conversion Rate.
CRUD	Create/Read/Update/Delete.
DB	Database.
DBMS	Database Management Systems.
DENWER	The local webserver emulation software.
GUI	Graphical User Interface.
IR	Infrared Radiation.
JAVA Swing	Java-based GUI framework.
PHP	Personal Home Page - a programming language for backend parts of a web site or service.
REST/RESTful API	A type of web API which accepts requests as a webpage address without using constructions like <i>http://site-name.com/?variable=value</i> . Instead of that it is using more simple construction <i>http://site-name.com/value</i>
TRIPWIRE	An ATS working at Demola New Factory (Tampere).

## 1 INTRODUCTION

Attendance tracking systems (ATS) are designed for tracking the number of visitors counted by ATS sensors during a particular time frame. ATS also can be used for determining visitors' movement direction but usually in such systems visitors are divided into coming and leaving ones. ATS systems are installed in the entrances of buildings and their precision depends on a used technology. Nowadays such systems are using a variety of technologies starting from infrared sensors and laser beams, ending with advanced technologies computer vision with the elements of artificial intelligence.

In this thesis project, the need was to enable visitors counting at Demola New Factory Tampere office. It was also required to enable handling statistical data and visualization of it as a graph. In order to solve this problem a client-server application for this purpose has been developed. The project goal was not only making just one more visitors tracking system but also making it customizable for the 3<sup>rd</sup> party developers. Therefore, each company who will use TRIPWIRE in the future can customize its client software according to its own needs and goals using TRIPWIRE's API as a kind of framework.

The design is divided into two parts: server-side and client-side software. Server-side software consists of a database where each visitor coming or leaving Demola is registered (crossing laser beam is tracked) and of a server application which communicates with clients via HTTP-requests. The server itself is based on Apache (Source [20]).

Client-side software is customizable. However, for Demola visitors tracking purposes and in order to demonstrate how a possible related client software solution might look like it was decided to develop an "official" TRIPWIRE client which will be included in the basic distribution by default. If a possible customer wants to implement other functionality than the "official" client is already containing he can develop his own client or extend the existing one.

The thesis covers theoretical aspects of such systems and describes in details a development process of the ATS TRIPWIRE, which has been installed at Demola's Tampere office.

The thesis structure is divided in chapters. The first chapter is the

current introduction.

Chapter #2 is dealing with the current situation on the market of ATS and technologies they are based on and includes information about existing devices and solutions.

Chapter #3 is dedicated to describing architecture, design decisions and related issues. Also, chapter #3 contains goals set up by Demola administration which must be achieved and includes the software design process itself with the description of both: client and server-side software and which tools and programming languages are used for the development

Chapter #4 is mainly about testing. The testing has been performed using so-called "user stories". In other words, the testing was done by simulating a typical user activities and looking if software behavior in response to those activities is the same as it was expected.

Chapter #5 summarizes the thesis, describes what was done successfully and what could be improved in the future. Furthermore, the chapter describes concrete issues to be sharpened and improved in the process of further development, testing and maintenance.

## 2 BACKGROUND

The goal of this chapter is observing ATS general background, specification of the technologies which are used by those systems and a brief discussing of existing service and equipment providers. The material considering all those aspects is provided. This is done using source [1] as a reference.

### 2.1 Applications of ATS

The main function of any ATS is counting a number of people coming to or leaving from the building per a limited time frame. Such kind of information is important for a lot of organizations who deals with high number of customers coming every day. This is important not only for shops but also for many other companies who are not dealing with sales and retail but still require such kind of data. Those are office clusters, business centers, co-working spaces for freelancers, start-up business incubators, bars and restaurants, and so on.

In the following we introduce a set of ATS use cases which have been considered for this thesis. This use case list is definitely not finite and here we are focusing only on the most common but sometimes not so obvious attendance tracking system uses.

#### 2.1.1 Trading and marketing efficiency control

Tracking the number of visitors is important in shops and supermarkets. A relation between customers (or people who are actually buying products or services) and the whole number of shop visitors is called *Conversion Rate (CR)*. Conversion rate analyzing is one of the main values which is used for trading and marketing efficiency estimation. Low CR (high number of people leaving a store without buying anything) means there issues needed to be detected and eliminated.

Making CR dynamics tables and graphs (that are measuring conversion rate per a fixed period of time e.g. every week) and analyzing them helps to track circumstances which affect positively and negatively on number of customers. For example, if a store conversion rate has significantly increased after launching of a new promotion campaign it

will be good idea to increase marketing budgets and funding for repeating such campaigns in the future.

Nowadays metrics such as CPM (Cost Per Thousand) and SSF (Shoppers per Square – number of visitors per square meter) are used together with conversion rate for marketing efficiency estimation. *Cost Per Thousand* and *Shoppers per Square Meter* are also measured using attendance tracking systems statistical data analysis.

As we can see ATS data analysis is a widely used and precise tool in the fields of trade and retail when it is required to calculate marketing and trade efficiency.

### **2.1.2 Personnel working efficiency estimation and HR management**

The required number of employees in banks, stores and other mass service organizations is directly proportional to the number of customers and other visitors coming to the company office. Attendance tracking system data helps in efficient planning of the personnel work schedule. Also statistical analysis of this kind of data might show that number of employees must be increased or decreased depending on how many visitors are coming and a shop conversion rate value, which is also calculated based on ATS data, as it was already mentioned.

There is an example provided below which illustrates the considered issue for a concrete situation. For example: the number of visitors in the shop significantly increased between 5pm and 6 pm, during the evening rush hour for the most of the cities. However, the conversion rate for this hour decreased, despite of a long queue and fully loaded cashier. Analyzing the number of visitors and CR together with queue sizes for the considered time period it is possible to state that a lot of people are leaving without buying anything due to long waiting time in the queue. As a result, the number of cashiers must be increased in order to prevent long queues, increase CR and, hence, the store profit.

Another example illustrates how ATS statistical data helps to optimize human resource expenses. ATS statistical data is often used for making an optimal working schedule for cashiers and other personnel and regulating the number of them during the working day. For instance, our

store has 5 cashiers working. Careful shop attendance data analysis will show us periods of time during the day when most of the customers are coming and estimate when we require all of the cashiers and when only 2 cashiers will be enough. Consequently the company is able reduce salary expenses and make prices for its products lower.

### **2.1.3 Events and maintenance activities planning**

Activities such as cleaning, fixing malfunctions, airing and etc. must be performed only when number of visitors in the building is zero or the least possible. Attendance tracking system sensors are providing necessary statistics that helps to avoid inconveniences during maintenance activities in the building.

Another aspect of the same use case is event planning and organizing meetings for relatively large number of visitors. Many companies, as well as, governmental and non-profit organizations are organizing different kinds of events such as meetings and conferences where large number of guests is presented.

Attendance tracking system statistical data analysis is able to demonstrate how planned (or signed up) number of people is different from number of people that really attended. This kind of data might help in the future for improving the quality of such events (for instance, more careful planning of amount of food and drinks which are required for the guests, choosing room of an appropriate size and so on). An example of how this use case is working will be demonstrated in details in further chapters.

### **2.1.4 Preventing security issues**

Attendance tracking system statistical data is important for preventing and eliminating security risks in places where big numbers of people are concentrating. An exact information about how many visitors and staff members there are in the building for a particular moment of time is required in case of emergency situations, for instance especially when evacuation is required. Attendance tracking system data helps to determine how many vehicles and rescue teams are required for evacuation and providing medical or other help on time for every person

who needs to get it.

However, modern ATS are not able to provide 100% exact information if all of the people who were inside the building had left it. Measurement errors which are quite small but still appear in any system. At the same time, these kinds of errors are small and do not affect significantly on estimation of resources required for emergency situation handling.

### **2.1.5 Efficient resource management**

Many organizations need to have an ATS statistical data for requesting an appropriate funding from government or private investors and sponsors. Organizations such as libraries and museums are funded based on the number of people attending them and using their services. This kind of organizations are not always selling tickets. For example, many libraries, museum and even theaters may have an entrance free of charge. So the process of visitors counting is required to be automated using ATS.

Except funding management a lot of organizations require attendance tracking system statistical data for power management, water supply, sufficient food serving in cafeterias and managing many other resources that organizations are consuming and using.

### **2.1.6 Public transportation work optimization**

Attendance tracking system is a good way to measure how effectively public transport is working. Current bus transit system of Tampere is well organized and provides high quality services. However, city population is not spread equally in the city and installing an ATS sensor per each bus door might help to determine how many people enter or leave the bus at each stop. This information could help to remove bus stops which are rarely used or used by an extremely low number of passengers.

Also careful ATS data analysis might help to decide on which bus lines it would be wise to substitute high capacity heavy buses by smaller ones (or even minivans) which are consuming less fuel.

Another way to use ATS data for the considered use case is reducing stowaways on public transport routes. Unfortunately, there are quite many stowaways among passengers - usually they avoid passing bus card validating device and enter the back door. Comparing number of validated tickets with number of passes through ATS sensors can provide information about number of stowaways per bus route during a particular period of time. Applying such actions against stowaways as hiring ticket inspectors will be effective if number of ticket inspectors at each bus route will be in direct proportion to the number of stowaways. The same is true for other public transportation systems as trains and subways.

Public transportation work optimization based on ATS statistics data would reduce prices for tickets and improve general ecological situation in the city.

## **2.2 Commonly used technologies and devices**

In order to organize working of ATS as described above, modern ATS systems are using various technologies having their own advantages and disadvantages. In the following we introduce the most commonly used technologies and describe their pros and cons.

### **2.2.1 Infrared counters**

One of the most frequently used ATS are based on infrared visitors counters. These counters are designed using infrared beams emitters and can be **vertical** and **horizontal**. Even despite the similarity of both of these IR visitors counters types have differences which are described below.

**Horizontal counters** are working using infrared beam interrupts. A typical counter is looking like a device in illustration 1. A horizontal infrared visitors counter have beam emitter and beam receiver devices.

A beam is directed from emitter to receiver. If beam is crossed, the receiver is not getting it any more (beam interruption) so the system counts one visitor (illustration 1).



*Illustration 1:  
Horizontal ATS counter  
working scheme.*

*Source [2]*

Horizontal counters are divided into two categories: computer connected and not connected. Computer connected counters have more superior functionality. Statistical data output is sent to a computer (server) which has all the required software for handling this kind of data.

Non-PC connected counters are cheaper. They use simple LCD display for data output. However, they require a constant monitoring of statistics by a human (e.g. every hour a special person must register the number of visitors in a special diary) this is less convenient comparing with automated statistics monitoring. Non-PC connected counters statistical data can be seriously affected by a human factor (a person who is writing statistics can make a mistake).

**Advantages.** Infrared (IR) horizontal counters are not expensive comparing with counters based on other technologies and comparing to vertical infrared visitors counters. IR horizontal counters are the most simple for installation. In the most cases even a simple double sided adhesive tape is enough for mounting mount the counter on the wall. Hence, installation is also quite cheap and takes a little amount of time including an initial beam calibration. IR horizontal counters are quite accurate. The average accuracy is 95% (entrances width up to 1.5 meters and number of visitors up to 100 visitors per hour).

**Disadvantages.** IR horizontal counters are inappropriate for wide entrances (1.5 m width and more) and, unfortunately, quite vulnerable for data falsification might be easy - sometimes shop personnel may

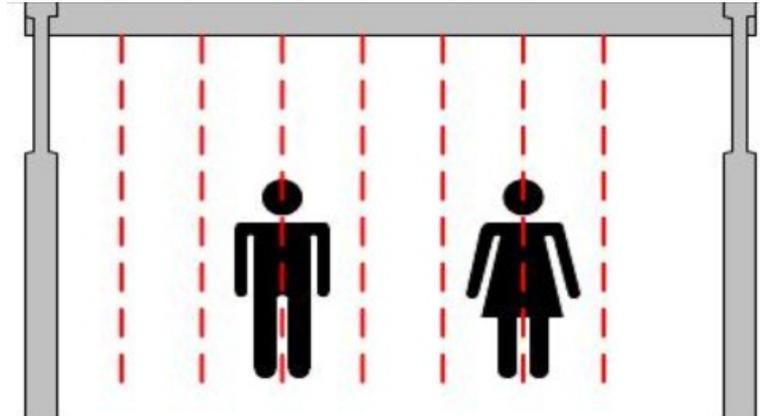
cover laser transmitter in order to make number of visitors lower and artificially increase conversion rate in order to make “good” reports for the top management and auditors. Moreover from time to time, IR beams must be recalibrated in order to prevent a significant accuracy decreasing.



*Illustration 2: A typical infrared beam based horizontal ATS sensor.  
Source: [3]*

**Vertical IR** counters are quite similar to horizontal. Their working principle is demonstrated on illustration 3. Vertical ATS counters are usually more precise than horizontal. Their accuracy might be very close to 100%. A typical vertical IR counters looks like a straight thin block and usually mounted above the entrance. Beam direction is vertical and beam emitters are mounted 1 per 50-60 cm along the width of entrance.

A typical 1.5m wide entrance usually contains 2-3 beam emitters. Measurement accuracy is 95% and higher. However, accuracy may decrease significantly if visitors intensity, the number of people passing through sensor beams during some period of time, is high.



*Illustration 3: Vertical ATS counter working scheme.  
Source: [3]*

**Advantages.** The main advantage of the vertical IR counters are quite accurate which is close to 100%. However, high 100% accuracy is possible when visitors intensity is relatively low.

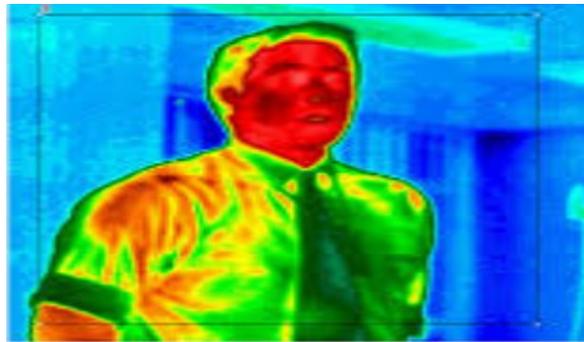
**Disadvantages.** Comparing to horizontal ones, vertical IR visitors counters are more expensive. They are also more complicated for installation: mounting (illustration 4) of such devices takes longer time, and not as cheap as for horizontal counters. Accuracy decreasing might happen quite often if visitors intensity is relatively high (however modern systems are improved all the time in order to decrease a significant influence of high people traffic through sensors). As well as horizontal, vertical ATS counters are required to be recalibrated from time to time.



*Illustration 4: Vertical visitor counter in supermarket. Source: [3]*

### 2.2.2 Thermal sensors based ATS

Sensors that are using thermal radiation of humans for counting visitors are called thermal sensors. Thermal radiation based sensors are “perceiving” and counting every human as an object comparing to infrared sensors which are counting in fact only beam interrupts and “ignore” objects. Usually thermal ATS are implemented as embedded. That means that such sensors are embedded in other ATS types such as infrared and used as a secondary supporting sensor which helps to increase the general attendance tracking system accuracy. Illustration 5 shows a typical human thermal map.



*Illustration 5: How thermal sensor "sees" a visitor. Source: [4]*

Thermal ATS sensors are easy to mount, they have high counting accuracy and they are able to work in a complete darkness. However, thermal sensors are weather dependent. In cold countries thermal ATS are not recommended to be mounted at building entrances because in winter periods of time sensor counting accuracy is decreasing significantly due to frequent temperature changes because doors are often being opened and closed. Illustration 6 demonstrates an example of the considered sensor.



*Illustration 6: Thermal sensor example. Source: [3]*

**Advantages.** Thermal sensors based ATS have quite high counting accuracy up to 100% if properly mounted and some conditions are satisfied. Such sensors are simple for mounting almost as horizontal visitors counters and can work even in a complete darkness.

**Disadvantages.** Thermal sensors are more expensive than IR sensors but high price is compensated with high accuracy. Mounting of thermal based ATS is quite simple but at the same time they are more technologically advanced and, hence, more complicated implemented and more expensive in production. Another significant disadvantage is weather dependency so such sensors are accurate enough for using in Northern Countries including Finland. Accuracy is decreasing in low temperatures and thus it is not recommended to mount such sensors at building entrance in winter.

### 2.2.3 ATS with Artificial Intelligence

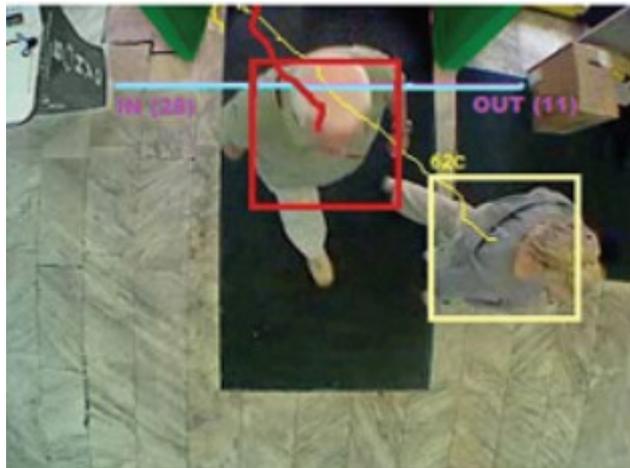
ATS systems that rely on artificial intelligence are using several IR sensors mounted close to entrance floor (not higher than a human ankle level) in order to create a kind of a sensor grid. When some object is appearing in the grid area it is compared with a pattern which is already pre-set in the ATS software system. If an object matches with a pattern a visitor is counted.

**Advantages.** Accuracy for such ATS is also quite high (around 98%). Those system are able to distinguish people from other objects passing through sensors and also able to count visitors outside of buildings. Artificial Intelligence ATS are temperature and lighting independent and, hence, are able to work in a complete darkness and/or very cold areas.

**Disadvantages.** Attendance tracking systems with artificial intelligence elements require big area for installing and not so easy for mounting, implementation and production. Also those system are ones of the most expensive.

## 2.2.4 Computer vision (camera based sensors)

Nowadays computer vision is the most modern technology for ATS which became widespread in the last few years. Computer vision based ATS sensors (illustration 9) are the most high-quality and advanced (with some limitations) for the current moment of time. These sensors are able to count people which are appearing in their camera scope as it is shown on the picture below (illustration 7).

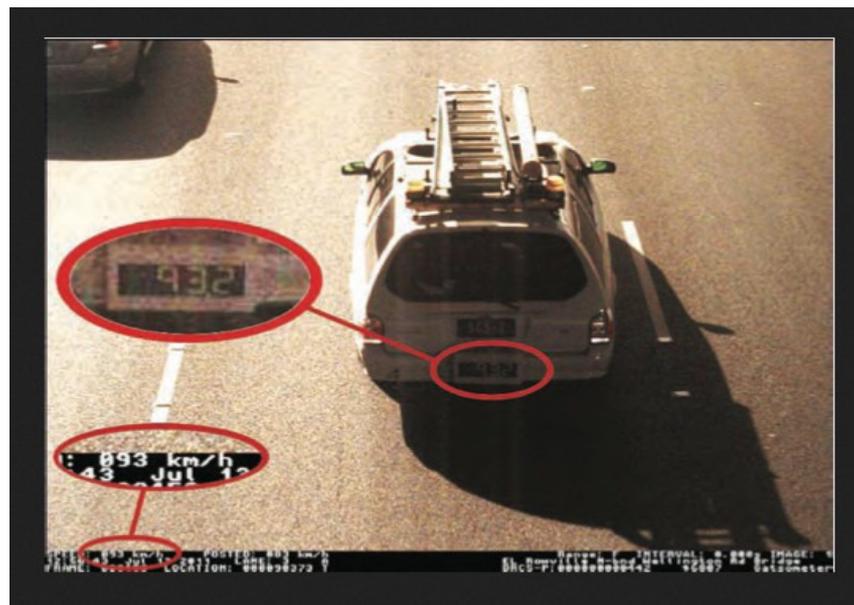


*Illustration 7: How camera based ATS sensor is "seeing" visitors. Source: [5]*

Computer vision based ATS sensors are very precise but they are sensible to lighting. So in order to achieve high accuracy the lighting must be stable without significant changes. In darkness and outside of the buildings counting accuracy might decrease significantly because of constant changes in street lighting conditions - changes of day and night, cloudy and sunny weather, and so on). However, this disadvantage is relatively easy to overcome by using auxiliary technologies (e.g. IR sensors together with cameras).

Camera based sensors are also applied when it is important to know more information about guests than just the fact of their visit. Those sensors are able to count people standing in a queue and distinguish them by age, gender and other categories ("sorting" by categories is also true for walking people). Software for such kind of ATS require high skilled programming and big testing work. It is usually tested using videos containing people wearing winter and summer clothes, people passing one by one and in groups walking close to each other, videos with bad quality simulating dirty or partially covered camera.

ATS based on cameras can be mounted on roads (near or above) and high-ways for counting the number of cars and detecting rule breakers. Nowadays such cameras which are interoperating with a specially designed software system can identify a car registration number and, hence, a person to be fined for breaking highway code. Especially over-speeding, illustration 8.



*Illustration 8: Car identification via its number. Source: [6]*

**Advantages.** Computer vision based ATS are extremely accurate (100% in good lighting conditions). They are able to “sort” people by age, gender, height and other categories and also applicable for the purposes of the road traffic control and, hence, are useful reducing road police expenses. ATS with computer vision can be relatively easy embedded into an existing compatible building video control systems, for instance security cameras. In this case there is no need in any additional hardware which simplifies mounting process and making it shorter and cheaper.

**Disadvantages.** Counting accuracy for the currently considered type of ATS is lighting dependable- So it is impossible to use camera based ATS in specially darkened entertainment places such as nightclubs, discos and laser-tag playgrounds.



*Illustration. 9: Camera based ATS sensor examples. Source: [3]*

### **2.3 The most known ATS producers**

In the following we will consider briefly the most known ATS producers and distributors presented on the European market. ATS producer and sales leader for the European market nowadays is British company “Footfall”. Its counting sensors register more than 6 000 000 000 visitors ever year. Other significant manufacturers are “BrickStream”, Irisys.

An Eastern European and post-USSR market is represented by a company “Watcom Group” which is registering 600 000 000 visitors per year and “ANTitheft” (Russian: АНТИвор) working in Europe as “Rstat” and dealing with retail counting, analysis and security (retail thefts prevention).

### **3 TRIPWIRE DEVELOPMENT PROCESS**

This chapter covers the development process of ATS “TRIPWIRE”. It is observing what kind of needs are required to be satisfied by the project implementation, setting goals needed to be achieved and includes information about system architecture as well as risks analysis and management.

#### **3.1. The need/problem behind the project**

The project partner is Demola New Factory. The partner needs an equipment and an appropriate software in order to count visitors coming to its Tampere location. The software is supposed to be used by Demola staff (the target user) for the purposes of registering people coming in and coming out, keeping and analyzing statistics, events planning and resource management.

Also, the project must contain an open API which could be used by 3<sup>rd</sup> party developers who would like to develop their own client software in order to customize ATS for their own needs and purposes.

The value for the end user is an advanced tool which saves time and potentially financial resources for keeping statistics and resource management. The commercial value of the ATS TRIPWIRE is flexibility for the end customers which is supplied by an open API for developing the 3<sup>rd</sup> party solutions.

Both: end-user and commercial values are measured by registering customers feedback, as well as collecting and analyzing user experience data. Project requirements had been derived during meetings with Demola facilitators and based on source [7]. Also supportive materials from sources [8], [9] are used for better analysis of the project needs and goals.

#### **3.2. Project requirements**

In the following table (table 1) there is a list defining which requirements must be completed for the project and their current status information (completed, partially completed, not implemented).

*Table 1: A list of project requirements*

<b>##</b>	<b>Requirement description</b>	<b>Status</b>
1	Setup IR based ATS sensor at the New Factory door.	Completed (equipment have been pre-adjusted by the customer).
2	Functionality for counting visitors	Completed (the current hardware supports registering incoming and leaving visitors each time they are crossing the beam and the related software is also implemented on a server side).
3	Implement log parsing (register every coming or leaving visitor) in order to determine number of visitors in a particular time frame).	Completed.
4	Visualization and graph data representation including hourly, daily, monthly histograms.	Completed (implemented on a client side for an "official" client).
5	REST/RESTful API which provides data and implements a functionality for the 3rd party (client) software.	Completed.
6	Make user access levels: two levels - administrators and users.	Completed.
7	Events planning and resource management functionality. Comparing planned number of people with number of visitors came in fact (in order to optimize future events	Completed.

	organization).	
8	Track visitors mood (make functionality for registering entries from happy/not happy buttons).	Partially completed (the needed software is implemented but the mood board (a device with 3 or 4 buttons looking like smiles) for it is still missing, additional testing of this software could be required when the moodboard is installed).

### 3.3 Risk analysis and management

Almost every software project is vulnerable for risks. At least basic risk management helps to complete a project on time and avoid unexpected circumstances. Table 2 is a result of a brief analysis for risks which is current project could be vulnerable for.

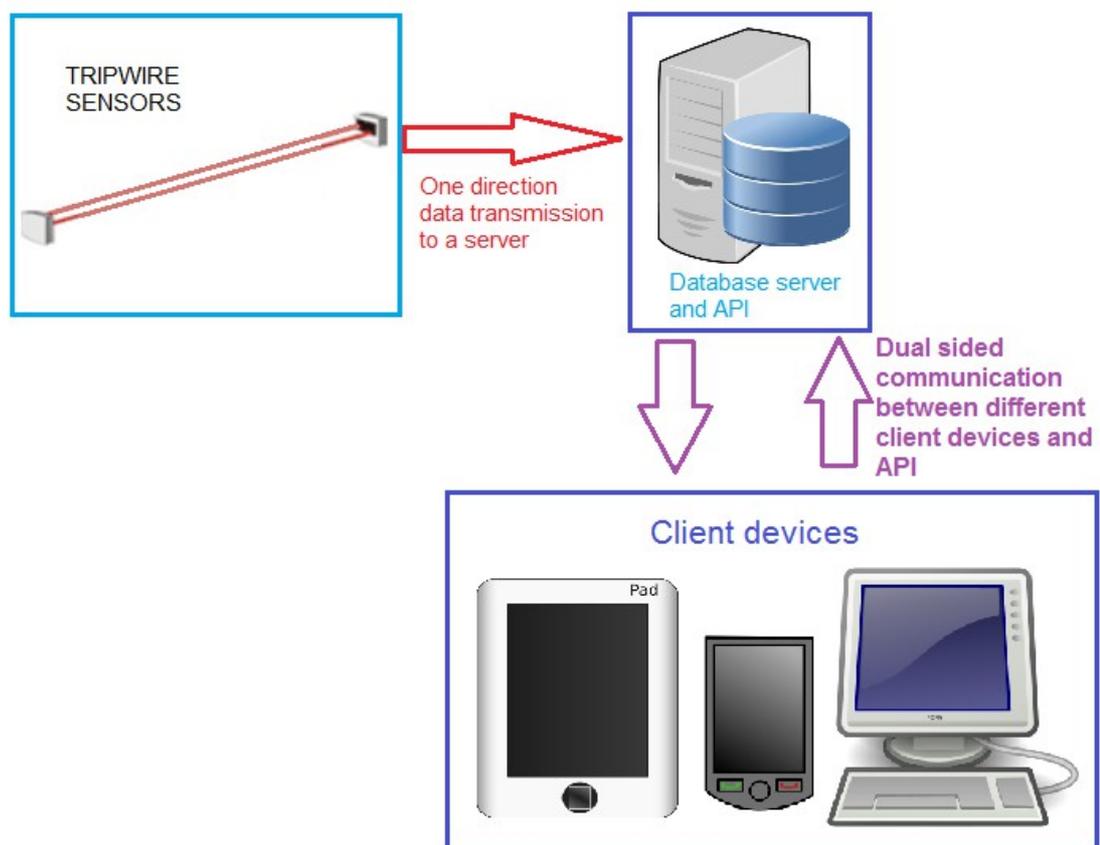
*Table 2: Risk management*

##	Risk description	Prevention activities	Probability
1	Software and hardware issues	Making reserve copies  Having additional hardware devices such as reserve external hard disks.	Low
2	Wrong functionality development	Careful program features planning and careful testing.	Medium
3	Low performance	Minimizing number of the SQL queries.	High

4	Bad design and UI	Testing and collecting usage data on the step of maintenance.	Medium
---	-------------------	---	--------

### 3.4 Planning architecture

The system consists of three levels of implementation: hardware, server software and client software. Hardware level is covered in a quite general manner because programming of embedded systems which was done here is not a topic for consideration in the scope of this thesis. Server and client side software will be considered below in a more detailed way.



*Illustration 10: Tripwire working scheme*

In illustration 10 (above) we can see how the ATS is working in general. The visitor ATS sensors - TRIPWIRE sensors drawn in a light blue

frame - are sending information about a visitor to database using a relevant API function. In a database it will look like a record which is containing a visitor's movement direction (the systems registers if a visitor is coming or leaving) and a visitors pass time stamp as it is demonstrated on the illustration below (illustration 11).

<b>Id</b>	<b>Time</b> Incoming or outcoming time	<b>type</b>
866	2013-05-20 16:20:34	Incoming
867	2013-05-20 16:20:35	Incoming
868	2013-05-20 16:20:36	Incoming
869	2013-05-20 16:20:37	Outcoming
870	2013-05-20 16:20:38	Outcoming

*Illustration 11: Example of a visitor registration in a DB*

Seconds are the latter part of the time stamp, which is recorded when a guest crosses a sensor beam and it is used in a database but do not really important for statistics analysis for the current client software because the smallest important period of time which could be used for calculations is one minute.

The server is containing all the functionality which is needed for getting and working with statistical data. Also it contains functionality for providing user access control by levels for some parts of the data - such as user and events management and a simple login/password authentication.

Communication with a client is happening via HTTP requests. It means that a client is retrieving some kind of information and than gets it as a response or (a negative response) based on a current user policy and parameters sent in the information request. Some activities in the ATS database are available for a server administrators only in order to prevent data corruption by less experienced users. User access control is implemented as a part of the API functionality.

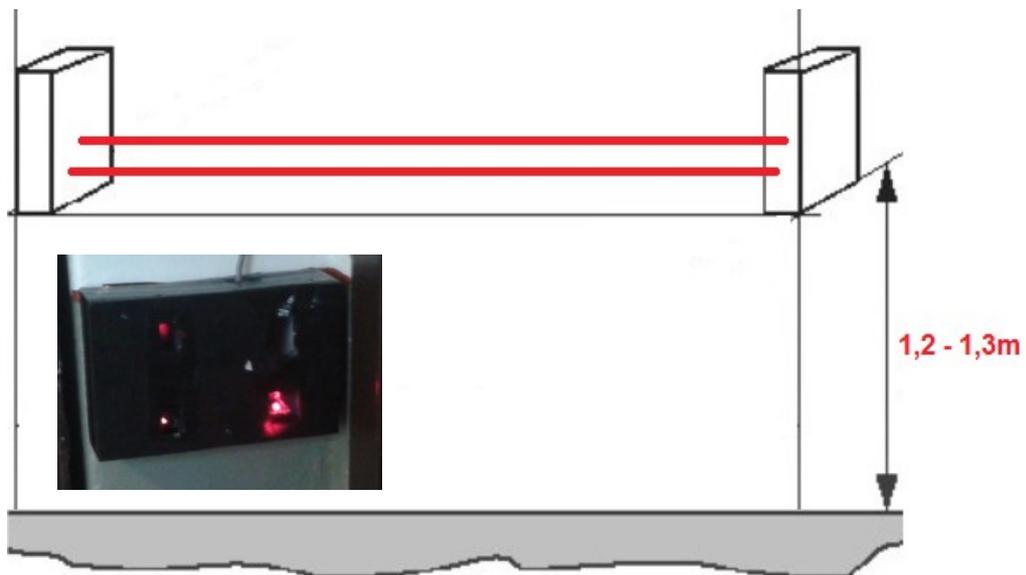
The current client software is the front-end part of the considered attendance tracking system which is required both for working with the statistical data and ATS management.

Client software is separated from the server-side implementation and can be adopted to the needs of the particular customer or implemented for any platforms including mobile environment. Later in this chapter one of the typical clients for TRIPWIRE will be considered and we assume it as the “official” one and all the rest as the 3<sup>rd</sup> party "adopted" developments.

The server-side software is also adjusted for working with data registered from happy/not happy mood board. This device is not installed together with TRIPWIRE, however, it can be easily mounted later because all the software functionality is already adjusted.

### 3.4.1. Hardware details

In general the system everything works as it shown on the illustration 12 below.



*Illustration 12: Tripwire main sensor*

The main hardware component of the system is a simple IR sensor based on two lasers (illustration 12 - inner photo with two beam emitters placed in black plastic box). Two lasers are needed for determining if a current visitor is coming or leaving. The lasers are installed in a plastic box glued to the wall near entrance approximately 1.2 – 1.3 meters above the floor. On the other side of the wall there are two laser beam

acceptors. Communication with a database server happens each time when a system detects a visitor and makes a record about him. In other words, a record in a DB is made every time when laser beams are interrupted.

### 3.4.2 Server side: database implementation aspects

Server-side application is consisting of two related to each other parts. The first part is a database for storing records about visitors, users and events happening in Demola and the second part is an API for handling of the data and implementation of the user accessibility divided by levels.

The database is implemented on the base of MySQL database management system (DBMS), which was selected after referencing with the source [10]. DBMS systems such as Oracle and MS-SQL Server were also considered as possible implementations for TRIPWIRE database server. However it decided not to use them because for this project MySQL Serve is the best choice because it is lightweight, more simple to use and open source, hence, allowed to use without legal restrictions. The database itself is consisting of 5 tables and built like it is shown on the illustration 13 below. Three are required to be connected with the foreign keys as shown in the illustration 13 using green lines.

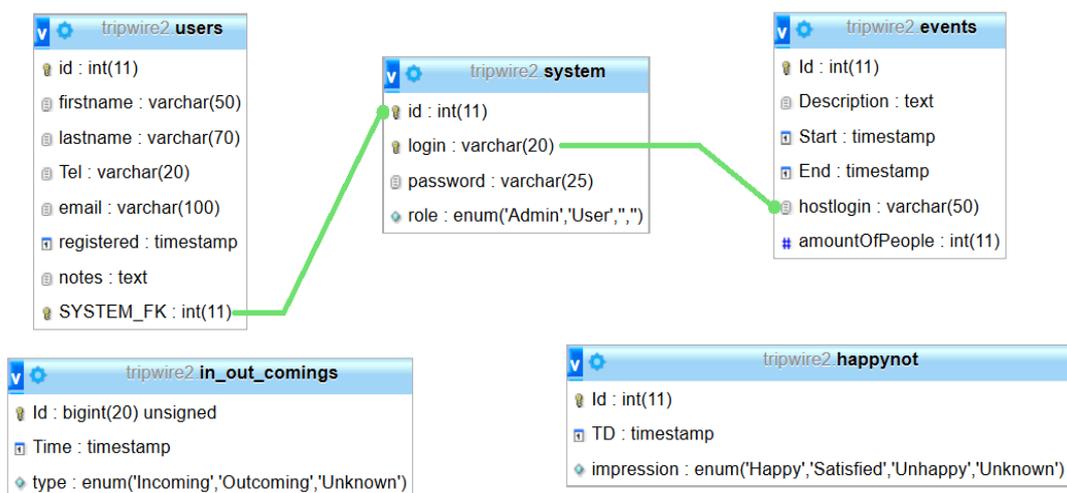


Illustration 13: Tripwire database scheme

A brief database model description is provided below.

Table *in\_out\_comings* is the most used table in the database because it is storing an information about visitors with distinguishing them on “Incoming” and “Outcoming”. If information coming from a sensor is corrupted (due to wrong laser calibration, signal noises, etc.) the record value for *type* is assigned to *Unknown*. Statistics is collected mostly by using this table.

Table *happynot* is supposed to work almost the same way as the previously considered one and it is used for registering user feedback - the users are offered to click the relevant mood board buttons. For the case of connection problems, noises and other issues an entry the record value for *impression* is assigned to *Unknown*.

*Unknown* value is also useful for analyzing how well in general the system equipment is working. Too many of such records might tell about equipment malfunctions and/or badly mounted or calibrated lasers.

The *System* table is containing information required for a user authentication and managing user access levels. The data from this table is also used for verifying the correctness of user login and password and user access rights based on his / her system roles (*Admin / User*).

Table *Users* is required for user management and containing additional data about people who will be using the system and connected with the table *System* via foreign key. User management is available only for administrators (*Admin* status).

*Events* table is containing information required for tracking a number of guests coming to some event and comparing them with the planned quantity of people (or signed-up) for a particular event. Such information is used for improving events organization in the future.

### **3.4.3 Server and API implementation aspects**

Current API is implemented "on top" of the current database in order to supply needed functionality for the system and avoid potentially unsafe use of the database (connecting directly from the client to DB is unlikely

due to security and information loss risks). Also this API provides a framework containing ready-made tools which can be called from a client software without reimplementing them in a client software solution.

Initially the API functionality was planned to be coded using Java. However, it was determined that API implemented on Java might be not good for TRIPWIRE because this system is not planned to be as big as large server applications (e.g. server software of corporate or ERP information systems). Consequently, it was decided to use PHP (Source[11]).

Comparing to Java, PHP fits well for small and medium-sized server applications and has more simple syntax. The overall “philosophy” of PHP could be formulated as: “Less code, more results”. For this project PHP fits very well because it allows to concentrate on designing data structures without making their implementation complicated. API RESTfulness is adjusted through relevant Apache settings registered in the *.htaccess* server file.

It is important to notice that the class diagram class data types should not be interpreted as the same concept existing in such languages as Java and C++. Here the text coming after “:” at the end of each attribute should be considered as the characteristic of a sort of information which is usually stored in and read from this variable during program run time. This is so because PHP has a dynamic object typing (types of an object may change while the program is executed).

The current API software has high degree of modularity. It consists of classes and each class has its own functionality. Two objects of diagram *Index.php* and *config.php* are not formally implemented as classes. However, they are included because their behavior is still class-like.

*Index.php* is the main file which accepts requests from clients, incoming parameters and returns information regarding to incoming parameters and their values. The main file accepts two types of parameters: *actions* and *action parameters (arguments)*.

The API structure is looking like it is shown on a class diagram provided below (illustration 14).

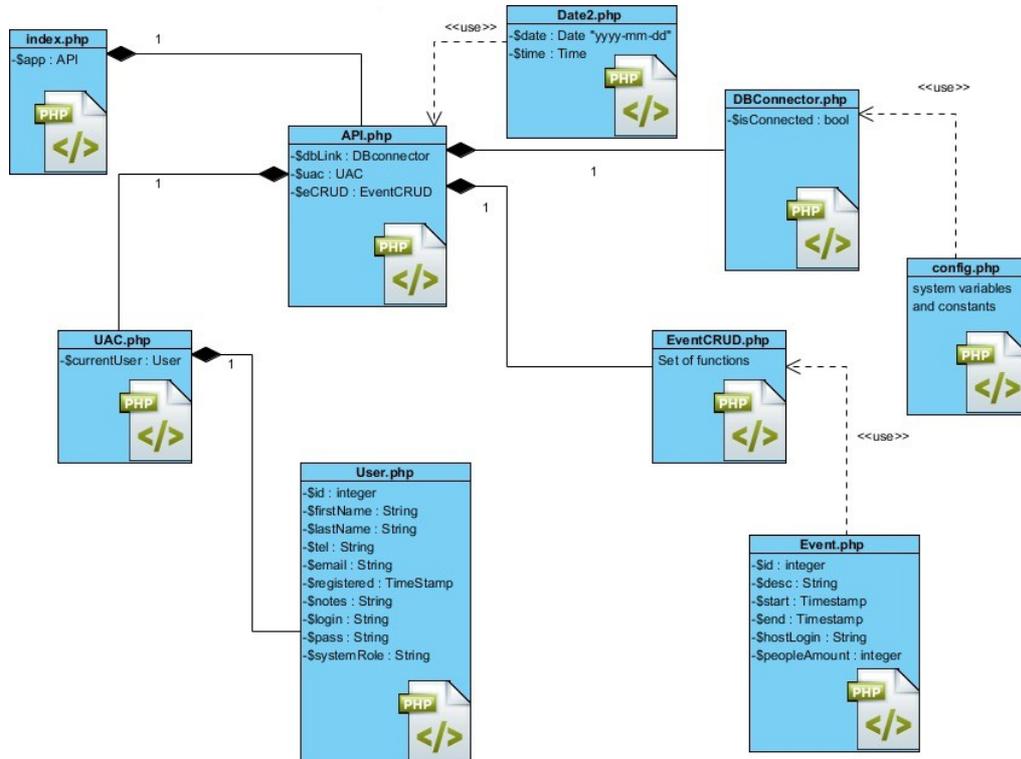


Illustration 14: API class diagram

An action is passed via GET method (`$_GET['action']` - e.g. <http://tripwire/test> where test is an 'action' value) – in other words what kind of information the client needs; action arguments (all other parameters required for fetching the needed information) are passed via POST method.

Also, together with action arguments two other parameters are transmitted each time when a request is sent to a server: `$_POST['login111']` which is responsible for passing username and `$_POST['pass111']` which is responsible for passing password. Those parameters are obligatory for passing authentication.

Currently the API software supports the following information requests listed below in table 3. In case if some parameter is requiring a special value to be assigned such value or its description is provided in round brackets next to the actual name of parameter.

Table 3: A list of information requests supported by the current API version

##	Request description and a relevant action (\$_GET['action']) parameter.	POST parameters and their possible values.	Server response description.
1	Record incoming or leaving visitor.  \$_GET['action'] == 'makeEntry'	\$_POST['type'] (Incoming, Outcoming, Unknown.)	No response.
2	Record a guest impression (mood board button pressed).  \$_GET['action'] == 'happy'	\$_POST['impression'] (Happy, Satisfied, Unhappy, Unknown)	No response.
3	Test if server is working. \$_GET['action'] == "test"	\$_POST['testString']	The relevant message and output string ('testString') is printed.
4	Add user to a system.  \$_GET['action'] == "addUser"	\$_POST['firstname'] \$_POST['lastname'] \$_POST['tel'] \$_POST['email'] \$_POST['notes'] \$_POST['login'] \$_POST['pass']  \$_POST['role']	If a user was added successfully the relevant message is printed. If not the reason is provided.

		(Admin, User).	
5	Get user info by his / her login  <i>\$_GET['action'] == "getUser"</i>	<i>\$_POST['login']</i>	If a user was deleted successfully an information about successful user erasing is sent to output. Otherwise an error message is sent.
6	Delete user with a specified login.  <i>\$_GET['action'] == "deleteUser"</i>	<i>\$_POST['loginToDelete']</i>	If a user was deleted successfully an information about successful user erasing is sent to output. Otherwise an error message is sent.
7	Update user information using a specified login.  <i>\$_GET['action'] == "updateUser"</i>	<i>\$_POST['loginToUpdate']</i> <i>\$_POST['fieldToUpdate']</i> (relevant table field in DB)  <i>\$_POST['value']</i> (new value)	If a user was updated successfully the relevant message is printed. If not the reason is provided.

8	<p>Get statistics for the whole period system installed.</p> <hr/> <p><i>\$_GET['action'] == "getOverallStat"</i></p>	<p>No parameters required.</p>	<p>If successful statistics is printed.</p>
9	<p>Get statistics for in between specified dates and times.</p> <hr/> <p><i>\$_GET['action'] == "ovDate"</i></p>	<p><i>\$_POST['date1']</i>  <i>\$_POST['time1']</i>  <i>\$_POST['date2']</i>  <i>\$_POST['time2']</i></p>	<p>If successful statistics is printed</p>
10	<p>Add a new event.</p> <hr/> <p><i>\$_GET['action'] == "addEvent"</i></p>	<p><i>\$_POST['desc']</i> (description)  <i>\$_POST['startDate']</i>  <i>\$_POST['startTime']</i>  <i>\$_POST['endDate']</i>;  <i>\$_POST['endTime']</i>;  <i>\$_POST['hostLogin']</i>;    <i>\$_POST['peopleAmount']</i> (integer value only).</p>	<p>If successful the confirmation message is printed.</p>
11	<p>Get event info by id.</p> <hr/> <p><i>\$_GET['action'] == "getEvent"</i></p>	<p><i>\$_POST['id']</i></p>	<p>If successful full event info is printed.</p>
12	<p>Delete event and its info using id.</p> <hr/>	<p><i>\$_POST['id']</i></p>	<p>A message of error is printed out if operation</p>

	<code>\$_GET['action'] == "delEvent"</code>		failed.
13	Update event info by id.  <code>\$_GET['action'] == "updateEvent"</code>	<code>\$_POST['id'];</code>  <code>\$_POST['field'];</code> (relevant database table field in DB to be updated);  <code>\$_POST['value']</code> (new value)	If not successful an error message is printed.
14	Get all registered users' logins.  <code>\$_GET['action'] == "getAU"</code>	No parameters.	If successful all users login values are printed.
15	Get all registered events' logins.  <code>\$_GET['action'] == "getAE"</code>	No parameters.	If successful all event values are printed.

The last 2 actions in table 3 (14 and 15) are auxiliary and usually they are used together with other actions.

A brief information about the rest of the classes is provided below. *API.php* contains a class which accepts the input parameters from *Index.php* and generates relevant output. It also encapsulates some implementation details. *Config.php* contains database connection credentials and system messages. Classes *EventCRUD.php* and *UAC.php* (*User Activity Control*) are typical CRUD-like (create, read, update, delete) classes for operating with objects *Event.php* and *User.php* respectively. *Date2.php* is a small class – implemented as a wrapper for working with MySQL timestamps.

### 3.5 An example of client software implementation

In the following, the development of an example client software is shown. The client is developed with Java programming language using three 3<sup>rd</sup> party frameworks: Java Swing for coding graphical user interface, JFreeChart (source [12]) for coding statistical data visualization and Apache Http Components (source [13]) framework for providing interoperation with server using HTTP requests.

Java has been chosen especially because of a well-designed framework Java Swing (Sources [14] - [16]) which contains a set of components which could be utilized for a quick GUI development. On the other hand, Java has worse performance in comparison with C++. However, C++ has less superior GUI tools in addition to more complicated syntax and more complicated memory management. The only exception for C++ is a QT framework which is quite similar to Swing by its features. Unfortunately, this framework still contains the mentioned C++ problems related to syntax, memory management and other issues.

The class relation diagram of a client application is provided on the illustration 15 below. The application consists of 10 classes. Near arrows there are name of variables used in code via which one class is related to another.

*App.java* is the main class of the application which is responsible for the main program output, events and users management. The communication with the server is happening via the *Communicator.java* object which accepts and parses server responses (uses HTTP).

*Visualizator.java* class is responsible for generation of statistical data graphs for a particular time period (periods) and providing this information to *visFrame.java* and *ImgViewer.java* classes which are responsible for displaying graphs and exporting them as a picture if necessary.

Classes *userAdder.java*, *UserEditor.java*, *EventAdder.java*, *EventEditor.java* are representing auxiliary dialog boxes which are responsible for adding and editing users and events respectively.

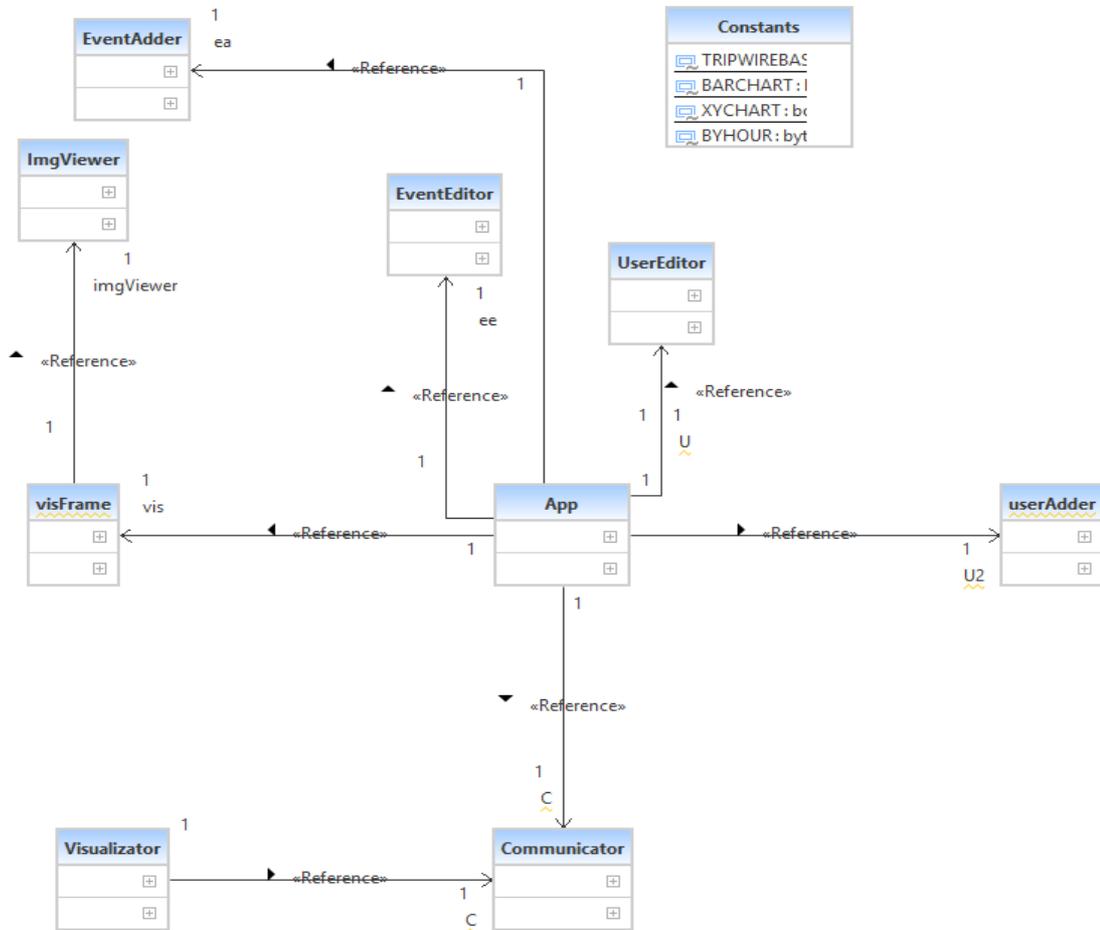


Illustration 15: TRIPWIRE client class relations

Deletion of users and events information does not require any special object and built in into *App.java* as methods. Comparing the planned number of visitors to some event with number of people came in fact is also in-built into *App.java*.

The following illustrations 16-18 are a number of sample screenshots illustrating the work of the client. Illustration 16 is showing how the statistical data might be visualized. Illustration 17 is an example of how the overall statistical data is retrieved as a simple text report and illustration 18 is demonstrating us how the full information about the events is requested and the planned number of guests for an each event is compared with the number of people attended it in fact

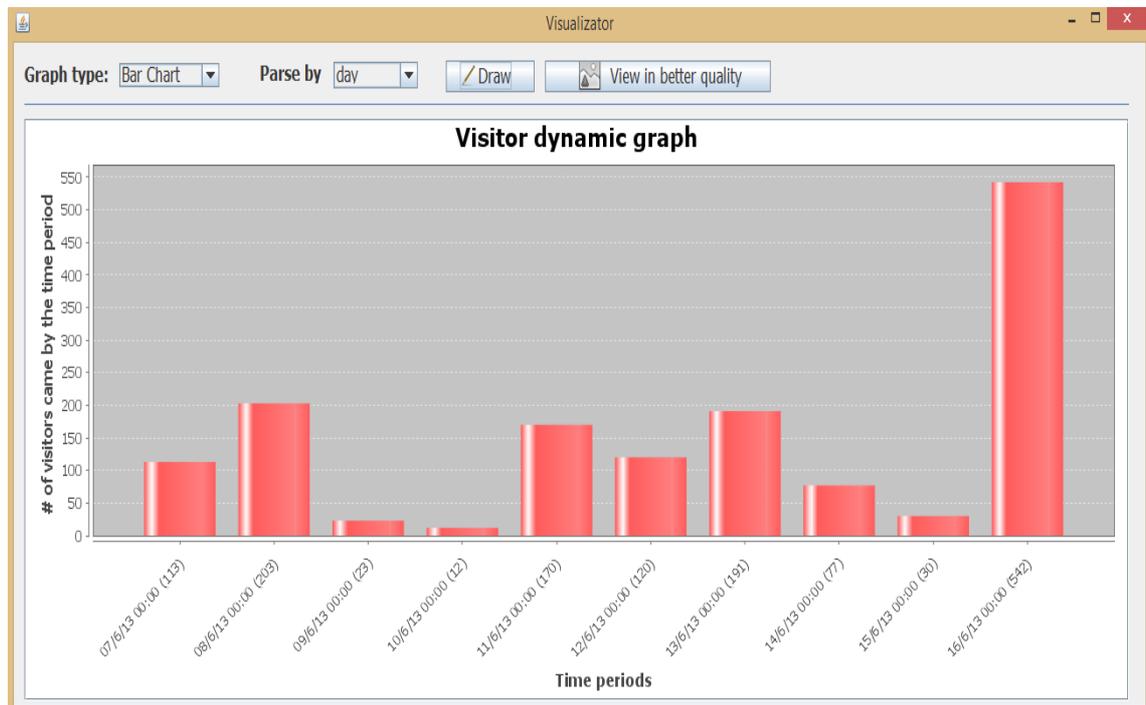


Illustration 16: Statistical data visualization example

The 'TRIPWIRE client' window shows a login section with 'Login: admin' and 'Password: \*\*\*'. Below are tabs for 'Statistics', 'Users', and 'Events'. The 'Visitor statistics request' section has 'From' and 'To' date pickers. The 'Statistics output' section displays the following data:

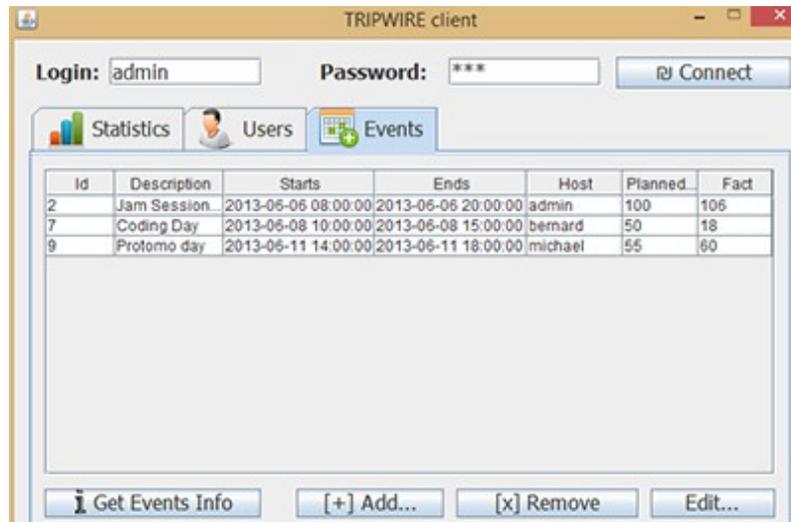
```

Since system installation it registered ( from: 2013-05-20
16:20:34 to: 2013-06-19 18:57:11 )
Ins: 2193
Outs: 1129
Unknown: 2
Maximum of visitors was on 2013-06-15: 542
Minimum of visitors was on 2013-06-16: 2
Average amount of visitors: 129.0000
Amount of happy people: 6
Amount of satisfied people: 7
Amount of unhappy people: 1
Amount of unknown happynot entries: 2

```

Buttons at the bottom include 'Get Statistics', 'Copy to clipboard', and 'Visualize...'. A checkbox 'Get overall stat.' is checked.

Illustration 17: Retrieving overall statistics



*Illustration 18: Retrieving events info list and comparing the number of planned visitors with the number of attending guests.*

## 4 TESTING PROCEDURE

This chapter covers used testing methods and testing environment and describing in details the testing process itself. During this chapter we will see how the software testing is utilized for estimating program performance and fixing bugs if any of them are appearing. For better understanding how Apache-based server is working source [17] was used partially.

### 4.1 Determining testing methods and setting testing environment

There are different testing methods including automated testing using such tools as Junit. However, it was decided to use the “manual” method based on testing according to so-called *user stories* (explained in section 4.2).

For testing purposes the *Denwer* server emulation environment has been used. It is not a specialized testing software but it contains all the required software that allows to expand a server on a local PC or laptop.

*Denwer* package contains local MySQL database server, PHP running environment, and Apache-based web server. The server side software is installed at *Denwer* as a “normal” web application and uses the “normal” web-address as if it was situated in the internet (in our case it is <http://tripwire/>)

Testing environment specifications are presented in table 4.

*Table 4: Testing environment specifications*

<b>Platform</b>	Laptop ASUS X502C, Intel Core i5 1.8GHz, OS Windows 8.1, 4GB of RAM, 500GB HDD
<b>Denwer version</b>	v 3.0
<b>Server configurations</b>	Apache v2.2, MySQL v5.4, PHP v5.3

Test results which are obtained via testing on a local computer might differ from the results obtained on a real server because the data needs to be transferred from a server to a client via internet connection. But due to low amount of transmitted information (not more than several kilobytes per request) the server performance and the performance while working on the local machine should not differ significantly from each other.

## **4.2 User stories based testing**

The testing procedure is based on user stories. It considers the most typical ways of using the application using. The purpose is to model the typical user activity according to provided description (e.g. registering a new event entry), measure the result (succeed/failed) and performance (how much time it takes to complete an action – might be measured several times and calculated the average).

Such method of testing is actually more close to inspection, however, it does not explicitly include the code inspection process. Each user story testing is supplied by a screenshot. During the process of testing the real test data collected at Demola New Factory in Tampere during May and June 2013 has been used. The main advantage of this testing method is an ability to become the "real" user and feel exactly what the the user is actually expecting from a product.

There is a difference between incoming and leaving visitors in statistics data because TRIPWIRE has been installed at the main entrance only. Some people could have used auxiliary exits which are not under ATS sensors control. In other words, such difference have not been caused by a programming error or hardware failure. For entering most of the guests are using main doors so the incoming visitors numbers are quite precise. Statistics is also might be affected when e.g. big enough boxes are carried through sensors but that is not happening often.

User stories and their testing results are provided in tables 5 - 15. Some user stories have a supporting screenshots (or a link to them) provided under user story specification.

Table 5: User story 1 details

##	User story description	Status (result)	Time (seconds)	Comments
1	A user opens the main application window enters his/her credentials and presses “Connect”	Success	2.5 – 3.0	If user login or password is wrong it might up to 5 seconds.

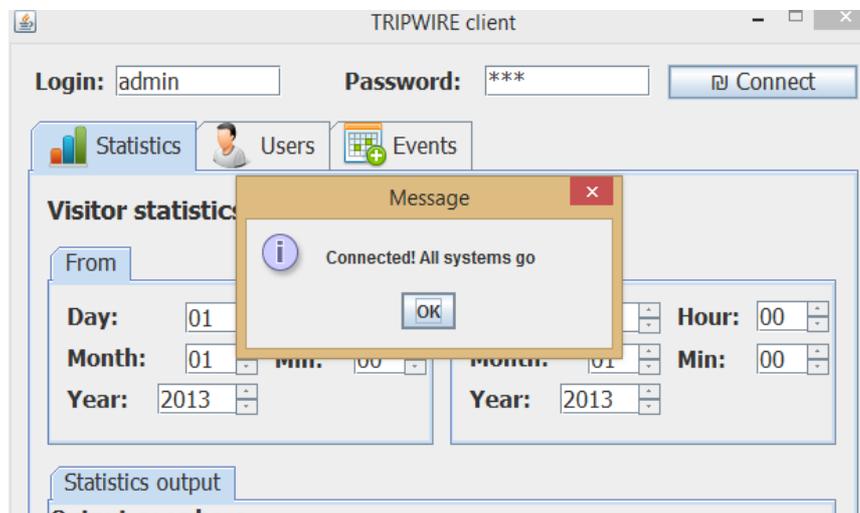
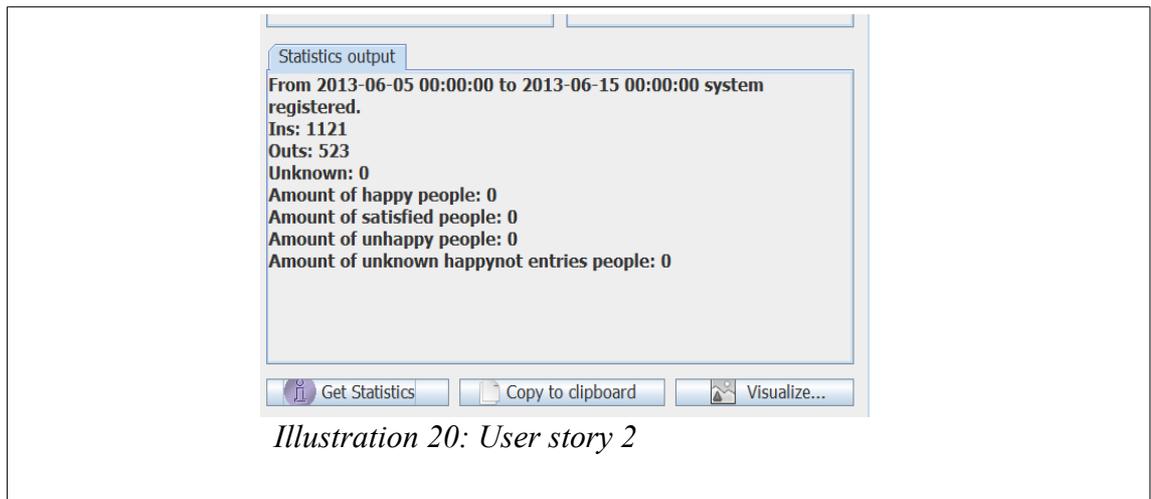


Illustration 19: User Story 1

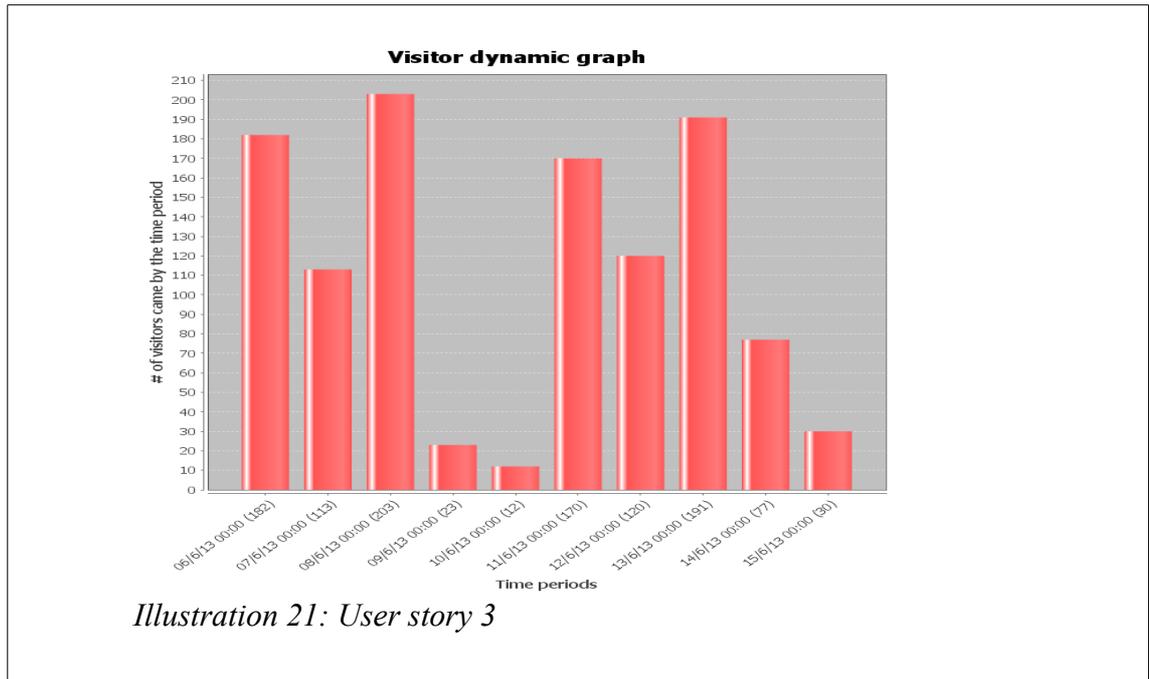
Table 6: User story 2 details

##	User story description	Status (result)	Time (seconds)	Comments
2	A user needs pure text statistics for from 05.06.2013 00:00 to 15.06.2013 00:00	Success	0.1	



*Table 7: User story 3 details*

##	User story description	Status (result)	Time (seconds)	Comments
3	A user needs to visualize the statistics from user story #2 in a graph which shows a number of visitors per day	Success	15.1	<p>Visualizing data by time periods requires several queries to DB to be executed. That might take time.</p> <p>The record 06/6/13 00:00 (182) means that from midnight 5<sup>th</sup> to midnight 6<sup>th</sup> 182 persons visited the location</p>



*Table 8: User story 4 details*

##	User description	story	Status (result)	Time (seconds)	Comments
4	User needs to visualize the statistics from 05.06.2013 00:00 to 09.06.2013 00:00 in a graph which shows number of visitors per day		Success	96	Visualizing data by time periods requires several queries to DB to be executed. That might take time.

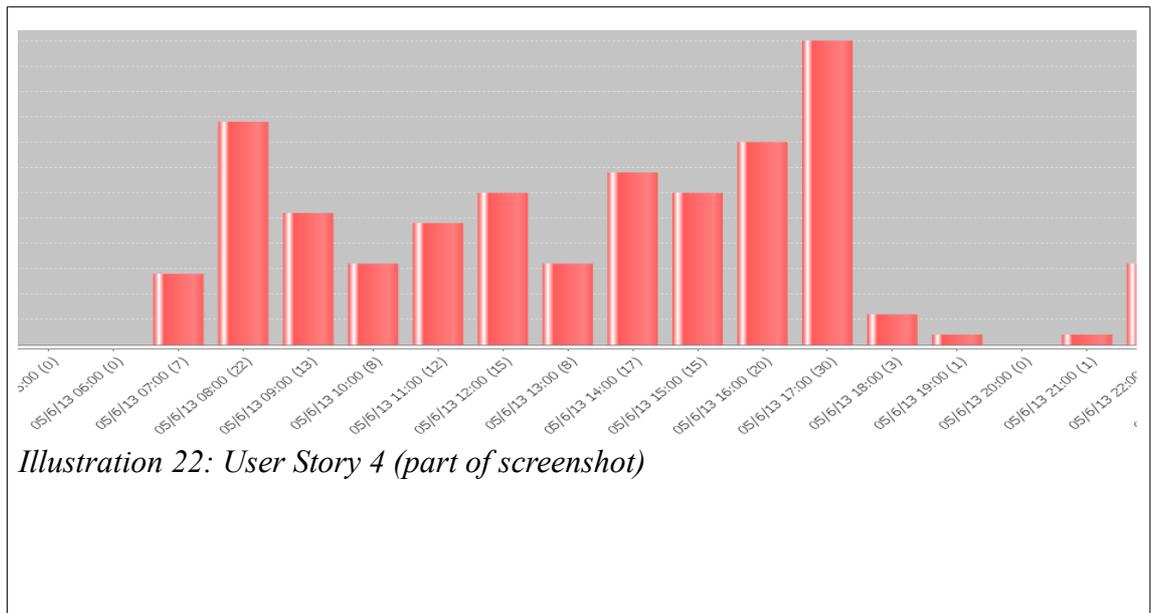


Table 9: User story 5 details

##	User story description	Status (result)	Time (seconds)	Comments
5	A user requests an overall statistical data	Success	< 1	
Screenshot: Illustration 17 page 32				

Table 10: User story 6 details

##	User story description	Status (result)	Time (seconds)	Comments
6	A user needs to get a list of current events and and compare a planned number people and the real number of guests for each of them.	Success.	6.7 – 7.0	<p>Screenshot: Illustration 14, top right, page #27.</p> <p>If a number of events is significant (more than 50) it might take up to a minute or two to get and analyze all of them.</p>
Screenshot: Illustration 18 page 33				

Table 11: User story 7 details

##	User story description	Status (result)	Time (seconds)	Comments
7	Get current system users list, add new one (e.g. John Simpson) and save him.	Success.	15.8	Excluding time on typing new user data and credentials.

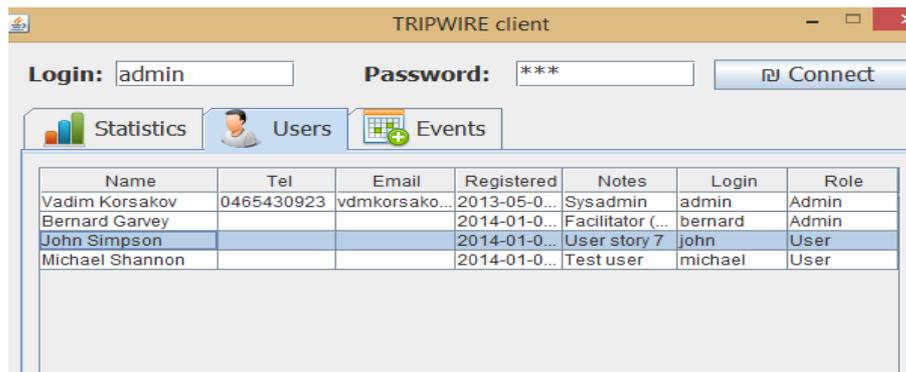


Illustration 23: User story 7

Table 12: User story 8 details

##	User story description	Status (result)	Time (seconds)	Comments
8	Enter missing credentials for John Simpson	Success.	10.4	Excluding time on typing new user data and credentials.

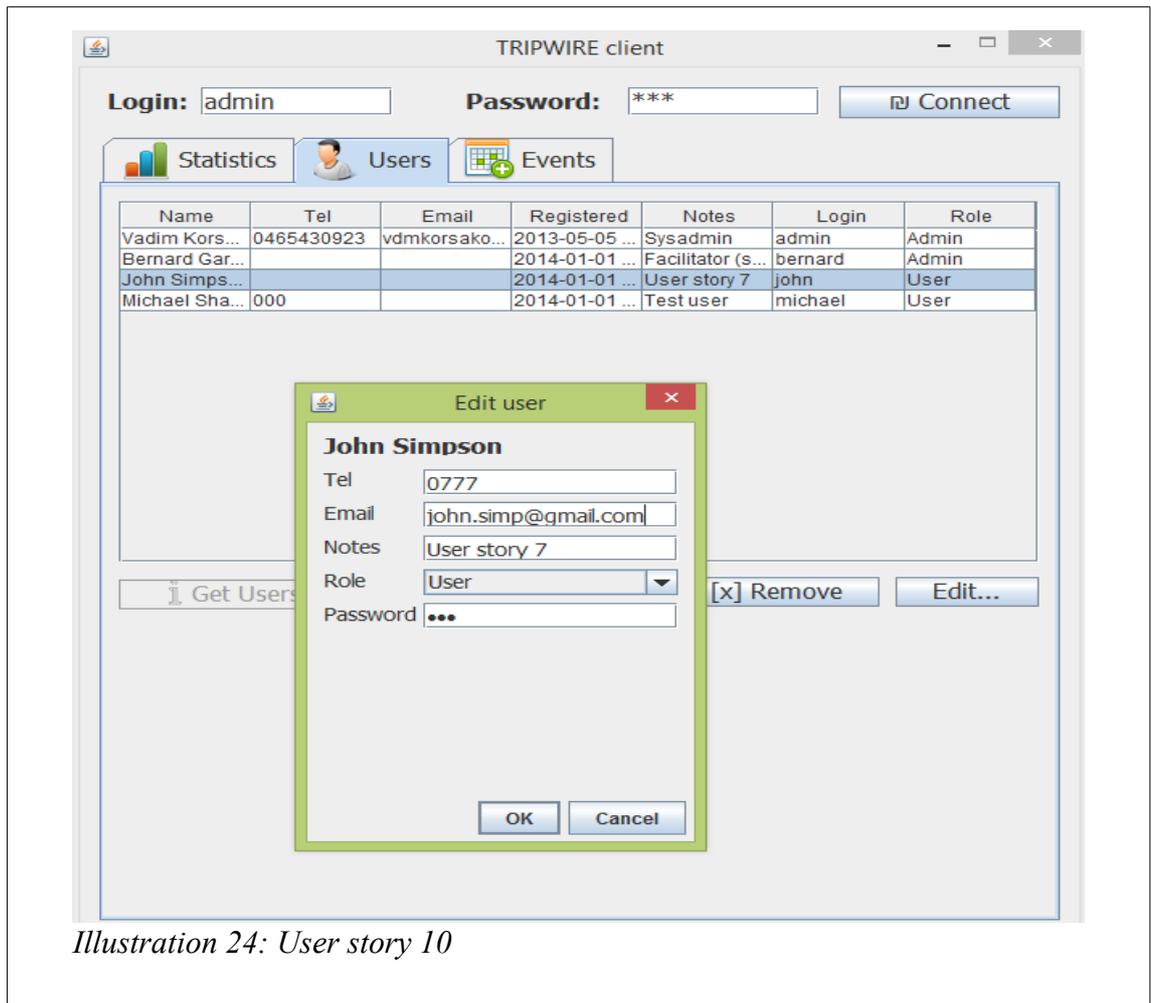


Table 13: User story 9 details

##	User story description	Status (result)	Time (seconds)	Comments
9	Visualize again data from user story #4 again and save it in *.png format	Success.	108	Including time spent on clicking relevant buttons and other UI controls.  No-screenshot activity.

Table 14: User story 10 details

##	User story description	Status (result)	Time (seconds)	Comments
10	Retrieve an overall statistics and visualize it by each day	Success	38.1	

**Visitor dynamic graph**

*Illustration 25: User Story 10*

Table 15: User story 11 details

##	User story description	Status (result)	Time (seconds)	Comments
11	Login as non-admin user (e.g. john and try to get users list). An error message should appear	Failed		Bug fixed (client side bug).

The server software has been tested using the same user stories via a normal browser in order to prevent server-side errors before client tests. No significant bugs or malfunctions were detected.

## 5 ACHIEVED RESULTS AND CONCLUSIONS

This chapter is dedicated to results, further maintenance and discussing future improvements.

The required functionality related to registering visitors has been fully implemented as it required by Demola including parsing statistical data, user access by roles (Admin, User) and events management. The main advantages of a system is scalability. The server side API is RESTful, also it could be used by classic OS and mobile platforms. Client software can be freely designed according to target user needs because the API is open. Additional functionality for the future moodboard is implemented.

The server-side PHP coding was performed using “Pure-PHP” concept. That means that no 3<sup>rd</sup> party PHP frameworks are used. That provides quite simple and easy to understand classes and data structures. In general, PHP frameworks (especially CodeIgniter) are very well solutions which are preventing so called “wheel invention”. But such frameworks are good for web sites and web sites-like applications.

Communicator.java is applicable to be reused as often as possible as a simple wrap class which simplifies HTTP requests functionality.

Despite the testing procedure that was performed in order to check the current software for bugs the system has still some features to be added or improved in the future.

Firstly, the current statistical data visualization is done on the client side but it would be better to make this function as a part of API. There are many users who requires such kind of functionality and it would be more convenient for them not to implement their own visualization each time. The visualization itself now is performed quite long. Making what is considered in #1 possible will increase the performance because visualizing on the server-side could be done by avoiding multiple queries to DB.

Another issue are user credentials which are transmitted anytime when server requests are performed. So the “Connect” button does not really connects the client to the server. By pressing this button an <http://tripwire/test> request is executed which checks if server is ready to

accept requests and send back answers. Could be fixed by session management implementation.

Another aspect relating to information security is password storing in TRIPWIRE database. For the future implementations it would be wise to use any of the presenting encryption algorithms in order to store user passwords securely. However, for this kind of system that is not that important.

Data from server is transmitted in a “self-made” format and client requires additional module to implement which will handle this data. In general, using JSON data format will simplify data handling on a client side because a lot of modern programming languages are having in-built JSON parsing tools or method or such tools can be easily obtained by using well-known 3<sup>rd</sup> party frameworks.

For now “manual” testing is acceptable but in future when system will become bigger automated testing only is appropriate because not all of the features are easily checked by a human-tester.

**BIBLIOGRAPHY**

- [1] General review of ATS <http://goo.gl/UfeBf> (Wikipedia, Russia)
- [2] Image source <http://goo.gl/cdOiwI>
- [3] Image Source <http://goo.gl/EU1Tfd>
- [4] Image source <http://goo.gl/5tQPDj>
- [5] Image source <http://goo.gl/xs1CJt>
- [6] Image Source <http://goo.gl/b60Z40>
- [7] Tripwire project announcement <http://blog.ossoil.com/wp-content/uploads/2013/04/tripwire-documentation-v2.pdf>
- [8] Daniel Drolet «Traffic counting can help stores track down lost customers», Ottawa Citizen June 2007 (<http://goo.gl/b3kHkw>)
- [9] Hollie Shaw «Converting retail browsers into buyers», National Post, July 2007 (<http://goo.gl/b3kHkw>)
- [10] Vikram Vaswani, "MySQL: The Complete Reference", McGraw Hill Professional, 2003
- [11] Ray Harris, "Murach's PHP and MySQL", Murach: Training & Reference, 2010.
- [12] JFreeChart framework home page <http://www.jfree.org/jfreechart/>
- [13] Apache httpcomponents home page <http://hc.apache.org/>
- [14] Java wiki-book [http://en.wikibooks.org/wiki/Java\\_Programming](http://en.wikibooks.org/wiki/Java_Programming)
- [15] Herbert Schildt, "Java The Complete Reference, 8th Edition"
- [16] Herbert Schildt, "Swing: A Beginner's Guide", McGraw Hill Professional, 2006.

[17] Ryan B. Bloom, "Apache Server 2.0: The Complete Reference" Osborne, 2002.

**APPENDIX A.**

An electronic media with tripwire source codes.