TARTU UNIVERSITY
FACULTY OF SOCIAL SCIENCES

NARVA COLLEGE
STUDY PROGRAM "INFORMATION TECHNOLOGY SYSTEMS DEVELOPMENT"

Artur Rumjantsev

DEVELOPING A BACK-END PART OF A MOBILE APPLICATION FOR PARTICIPATION IN LOCAL EVENTS

Diploma thesis

Supervisor: Assistant A. Sääsk

NARVA 2018

Olen koostanud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, põhimõttelised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

……

Töö autori allkiri ja kuupäev

Non-exclusive licence to reproduce thesis

I, Artur Rumjantsev (date of birth: 28.10.1994),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright “developing an application for participation in local events”, supervised by assistant A. Sääsk,

2. I am aware of the fact that the author retains the right referred to in point 1.

3. This is to certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Narva, 08.05.2018

CONTENTS

TERMS AND NOTATIONS

Web Server - a computer system that processes and fulfils requests of web clients.

Front-end – the parts of a computer, piece of software, or website that are seen and directly used by the user. (Cambridge… 2018a)

Back-end – the part of a computer system, piece of software, etc., where data is stored or processed rather than the parts that are seen and directly used by the user. (Cambridge… 2018b)

Hypertext Transfer Protocol (HTTP) - a set of clearly defined methods of communication between a web client and a web server.

REST (Representational State Transfer) - an architectural style of interaction between components of a distributed application on a network. RESTful web application listens for specific URI to execute a certain procedure. In most cases it simple HTTP GET or POST method.

Application Programming Interface (API) - a set of rules that defines how a software program can request and receive information from other software. (Cambridge… 2018c)

Document Object Model (DOM) - an application programming interface that describes HTML documents as a tree structure.

IDE (Integrated Development Environment) - a set of software tools used by programmers to develop software.

GUI (Graphical User Interface) – interface elements are executed in the form of graphic images.

RSS (Rich Site Summary) – designed to describe news feeds, announcements of articles.

Parse - to analyse of relevant objects**.**

Domain name - the part of an email or website address on the internet that shows the name of the organization that the address belongs to.

Android - an operating system used mainly for mobile devices. (Cambridge… 2018d)

INTRODUCTION

One of the problems of the modern world is the abundance of information, so, sometimes, it is difficult to track the necessary and important in this excess flow. The area of interest to the author is the search for information about events taking place in his hometown. At the moment, there is no single environment in which you can find information about all such events. Therefore, the idea arose to write an application for mobile devices that would help to bring all such information in one place and provide it with a user-friendly interface. Due to the large amount of work, it was decided to divide it into two parts between two authors, one of which would deal with back-end-part of the application, and the other – front-end-part.

Thus, the purpose of this work is to develop the back-end-part of the event search application for the Android mobile platform, as well as a small administrative console to administer the database of event streams. The purpose of the back-end part of the application is to collect information about upcoming events from different sources and load this information into the database for quick access by the front-end part of the application. The purpose of the administrative console is to give the database administrator a convenient tool to add and edit information flows, based on which the event database is formed.

Another author takes over the development of the front-end part of the application. With the help of the front-end part of the application, users can easily find a suitable event based on their preferences.

To achieve the goal of the thesis the author must perform the following tasks:

- Develop a script to retrieve information about events from sites that provide or publish such information;
- Create a database to store information about events in it;
- Create a secure client-server connection between frontend and backend parts of the application based on SSL;
- Create a RESTful API to quickly interact with the front-end part of the application;
- Create a Windows-based database management application;

It is planned that the application can be used not only in Estonia but also abroad. The author also considers it important that the developed application will provide a platform for the organizers of various events to disseminate information about the upcoming event among users of the application.

The work consists of an introduction; the first chapter, which describes the technologies used in the project; the second chapter, which describes the implementation of the project, as well as the interaction with a partner who develops front-end-part of the application; conclusions and a list of terms and abbreviations used. The full source code of the scripts and applications created within this work is presented in Appendix 1.

1 USED TECHNOLOGIES

This chapter describes what technologies are used in the project, as well as why the author chose to work with these technologies.

1.1 C# programming language

C# is a modern object-oriented programming language, which is currently the best choice for a developer when there is a need to write an application for Microsoft Windows platform. That is why the author chose C# for developing of the Windows-based database management console for the "Free Time Organizer" application. In addition, the author of this thesis learns to program in C# and the experience of writing a program in this programming language is very important to him.

1.2 PHP programming language

PHP is a popular scripting language. This programming language is very popular in writing web applications. This language is very easy to learn. PHP is supported by the hosting on which the server and database with events, so this programming language was chosen to write the API. (The PHP… 2018a)

PHP programming language was chosen, because it is very simple to learn and it is in the author's opinion the easiest language for writing a back-end and creating a RESTful API.

1.3 IDE Microsoft Visual Studio 2015

Visual Studio is popular with developers. It allows you to write in C# and other languages. Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, forms designer for building GUI applications, web designer, class designer, and database schema designer. (Microsoft 2018)

1.4 NuGet

NuGet is a free and open-source package manager designed for the Microsoft development platform. NuGet is distributed as a Visual Studio extension (Brockschmidt; Myers 2018). This package manager needs to be installed for the latest working version of the library MySqlClient to work with the database and Json.NET to work with JSON format.

1.5 Domain

Since the Android application, the back-end of which is developed in this thesis, is called "Free Time Organizer", the logical domain name to register for it was www.fto.ee. The domain name and the hosting for the domain were purchased from the service provider Zone Media OÜ. (Zone… 2018)

1.6 Web server

Virtual machine specification:

- Disk space - 128GB;
- Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz;
- Unlimited data transfer
- SSL certificates;
- HTTPS support;
- Cron jobs - 2;
- PHP5 support;
- A SQL database server

The hosting also provides free SSL certificates from “Let's Encrypt”.

1.7 MySQL

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. (Oracle… 2018)

1.8 PhpMyAdmin

phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the Web. phpMyAdmin supports a wide range of operations on MySQL and InnoDB. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc.) can be performed via the user interface, while user still have the ability to directly execute any SQL statement. (phpMyAdmin… 2018)

The author chose phpMyAdmin for this project because it has very good documentation and it was already pre-installed on the hosting, which was bought for the application.

1.9 Slim

It is an ideal tool for creating APIs that use, re-use any data or provide access to them. Slim supports dependency injection. Slim provides a fast and powerful router that maps route callbacks to specific HTTP request methods and URIs. It supports parameters and pattern matching. (Slim… 2018)

To install this framework, you need to install the composer package manager and enter to the project folder through the terminal and register as in Listing 1.

Listing 1. *Installing Slim via composer.*

```
$ composer require slim/slim "^3.0"
```

1.10 SoapUI

The most advanced REST testing tool in the world. SoapUI is an open-source web service testing application for service-oriented architectures and representational state transfers (REST). Its functionality covers web service inspection, invoking, development, simulation and mocking, functional testing, load and compliance testing. (Smartbear 2018)

The author chose SoapUI for this project because it has very good documentation and it has free version of program.

1.11 SSL

SSL certificates create an encrypted connection and establish trust. SSL certificates create a foundation of trust by establishing a secure connection. To ensure visitors their connection is secure, browsers provide visual cues, such as a lock icon or a green bar.

SSL certificates have a key pair: a public and a private key. These keys work together to establish an encrypted connection. The certificate also contains what is called the "subject," which is the identity of the certificate/website owner.

1. Browser connects to a web server (website) secured with SSL (https). Browser requests that the server identify itself.
2. Server sends a copy of its SSL Certificate, including the server's public key.

3. Browser checks the certificate root against a list of trusted CAs and that the certificate is unexpired, unrevoked, and that its common name is valid for the website that it is connecting to. If the browser trusts the certificate, it creates, encrypts, and sends back a symmetric session key using the server's public key.
4. Server decrypts the symmetric session key using its private key and sends back an acknowledgement encrypted with the session key to start the encrypted session.
5. Server and Browser now encrypt all transmitted data with the session key.

(DigiCert 2018)

1.12 JSON

JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

(JSON 2018)

1.13 Webalizer

The Webalizer is a fast, free web server log file analysis program. It produces highly detailed, easily configurable usage reports in HTML format, for viewing with a standard web browser. (Barrett 2014)

2 APPLICATION DEVELOPING

This chapter describes application development: event selection, project layers, database tables, daemon usage, and other practical aspects.

2.1 Development method

Together with another author, we chose that the development method for this work should be extreme programming, because this the method of choice when the requirements for the project are not clear and difficult to imagine.

The main features of extreme programming are following (Jeffries 2011):

- At the beginning of the project there is only an approximate plan, which, after each iteration, becomes more detailed and the understanding of the whole project increases;
- High frequency of releases. The new version of the product has minor changes compared to the previous one, but the release time is minimal;
- Refactoring. Improve code quality without reducing functionality;
- Collective responsibility. Despite the fact that each team member performs their part of the work, the whole team is responsible for the code as a whole;

The authors met once a week to discuss the details of the project and possible future problems. We decided what would be the design that would be acceptable for our mobile application. Such meetings did not last long, on average it was about 2 hours for this weekly discussion.

2.2 Sources of information about events

Information about events can be extracted from various websites and news feeds. The main source of such information in Estonia is Piletilevi. It provides a free for all RSS feeds, which are easy to parse. Piletilevi RSS feeds are categorized by genre, i.e. each genre has its own feed (AS Piletilevi 2018). Another important news source for us was hometown website narva.ee.

For the flexibility and completeness of our application, it is important, that it could work with any kind of news feed, which is currently available, so our back-end administrative console should accept links to news feeds in all existing popular data exchange formats like JSON, XML and HTML. Currently administrative console only accepts JSON links, but, in the future, the author will add XML and HTML support, too.

2.3 Parse events

For parsing web pages the author used a very useful script called “simple html dom”, which is written on PHP (Chen; Schlick; Carroll 2014). It parses web pages for the information and returns a structured array, which is easy to process further. One of the problems in using web sites for finding information about events was that not all web sites contain all the information the script expects to find about the events. All the information it finds is loaded into the database, when there is no information for particular data fields, a null value is loaded into the database instead.

Currently, for the sake of the thesis functionality presentation, the author only added two sources of event information to the project: Piletilevi and Narva town web site. In the future, many more sources will be added.

2.3.1 Piletilevi web site

Piletilevi is the main source of event information is the current version of the application.

The major problem with parsing the Piletilevi website was with events, which lasted for several days, while taking place in different locations each day or even having different types of events in different days. In this case, HTML tags for those events could be different, which made it difficult to associate. To solve this problem, the script has to check the similarity of the HTML tags. The script takes a link from RSS feed, parses the HTML-file it points to and loads the parsed data into an array, and then into the database. After each pass, the script clears the array of data and then starts over.

By Piletilevi internal classification, all single-day events are called “concerts” while all multi-day events are called “shows”. The script structure for parsing Piletilevi data flow is shown on **Figure 1**.

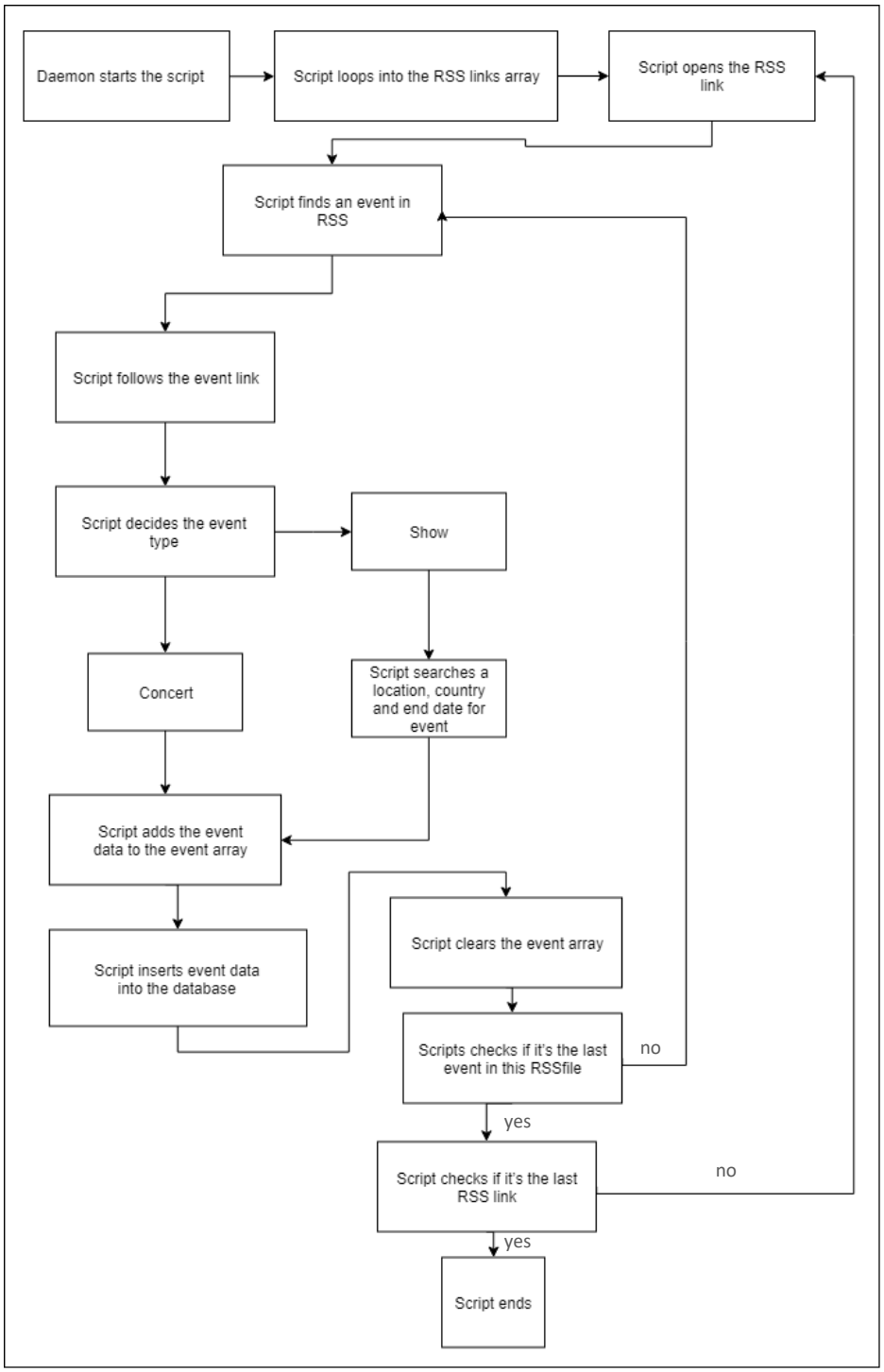

Figure 1. *Logic of script for Piletilevi. (Source: author)*

2.3.2 Narva town web site

For parsing the web site of his hometown, Narva, the author has created a separate script, which only works for this one particular web site. Narva town's site contains information about events start dates, end dates, locations and some additional information. This web site was suitable for parsing, as all information was presented in tables.

As there is not much information on this site and as it has a very simple structure, there were no problems with parsing this information and adding it to the database.

2.4 Daemon

Due to the fact that the hosting offers only 2 cron jobs, it was decided to make a simple daemon. A daemon is script or program, which always runs in the background and performs the tasks that have been set.

The pcntl_fork() (The PHP… 2018b) function creates a child process (**Figure 2**). The child process is then written to the $pid variable and the parent process ends. The copy that is separated from the parent process continues to run without a break.

The daemon once in 10800 seconds (three hours) runs the script, which adds new events to the database.

To start the daemon, enter the command in the terminal as shown in **Figure 3**.

```php
<?php
  // Fork proccess
  $pid = pcntl_fork();
  if ($pid == -1) {
    // Error
    die('could not fork'.PHP_EOL);
  } else if ($pid) {
    // Parent process die. We are now child process.
    die('die parent process'.PHP_EOL);
  } else {
    // New proccess, start main cycle
    while(true) {
      $ch = curl_init();
      curl_setopt($ch, CURLOPT_URL, "https://fto.ee/api/dom/run.php");
      curl_setopt($ch, CURLOPT_HEADER, false);
      curl_exec($ch);
      curl_close($ch);
      sleep(10800);
    }
  }
```

Figure 2. *Daemon code. (Source: author)*

```
virt71844:sn-69-33.tll07.zoneas.eu:~/domeenid/www.fto.ee/htdocs/api/dom> php deamon.php
die parent process
virt71844:sn-69-33.tll07.zoneas.eu:~/domeenid/www.fto.ee/htdocs/api/dom> ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
virt71844  61551  0.0  0.0 278728 54776 ?        SN   22:45   0:00 /opt/zone/bin/php71-cgi
virt71844  64922  0.0  0.0  86312  3580 ?        SN   22:56   0:00 sshd: virt71844@pts/5
virt71844  64923  0.1  0.0  81700  4072 pts/5    SNs  22:56   0:00 -bash
virt71844  65151  0.1  0.0 423076 15408 pts/5    SN   22:56   0:00 php deamon.php
virt71844  65176  0.0  0.0  76040  1904 pts/5    RN+  22:57   0:00 ps aux
```

Figure 3. *Starting daemon. (Source: author)*

2.5 Back-end

The backend of the project consists of a script written in a PHP language and a database, parser of events from different sources. The main problem is SQL injection - an attack on the database, which will allow to execute some action, which was not planned by the developer of the back-end-part. RESTful APIs use functions such as prepare() and bindParam() to prevent SQL injection (Listing 2). In bindParam() functions, all variables will be displayed as brackets in SQL query.

Listing 2. *Example part of RESTful API code.*

```
$app-
>post('/reports/create', function (Request $request, Resp
onse $response) {
$data = $request->getParsedBody();
$sth = $this->db->prepare("INSERT INTO reports
(event_id,report) VALUES (:event_id,:report)");
$sth->bindParam(':event_id', $data["event_id"]);
$sth->bindParam(':report', $data["report"]);
$sth->execute();
$sth->fetchAll();
return $this->response->withStatus(201);
});
```

Listing 3. *Example of SQL queries in RESTful API code.*

```
SELECT city FROM events WHERE city IS NOT NULL AND
city!='' GROUP BY city;
INSERT INTO genres (genre, description) VALUES (:genre,
:description);
UPDATE events SET title = :title , date=:date, end_date =
:end_date, time = :time, location = :location, city =
:city, country=:country, genre=:genre, info=:info,
link=:link, img=:img, price=:price, des=:des WHERE id =
:id;
```

2.6 Layers of project

The project can be conditionally divided into 4 layers (**Figure 4**), such as:

1. Back-end layer;
2. Communication layer;
3. Front-end layer;
4. Consumer layer;

Back-end layer contains a database that stores all the information about events and a Slim framework that protects the project from SQL injections. In front-end layer is Android application or desktop application for editing the database. In the communication layer response for JSON and request for RESTful API via URL. Consumer layer have an end user like administrator or customer.

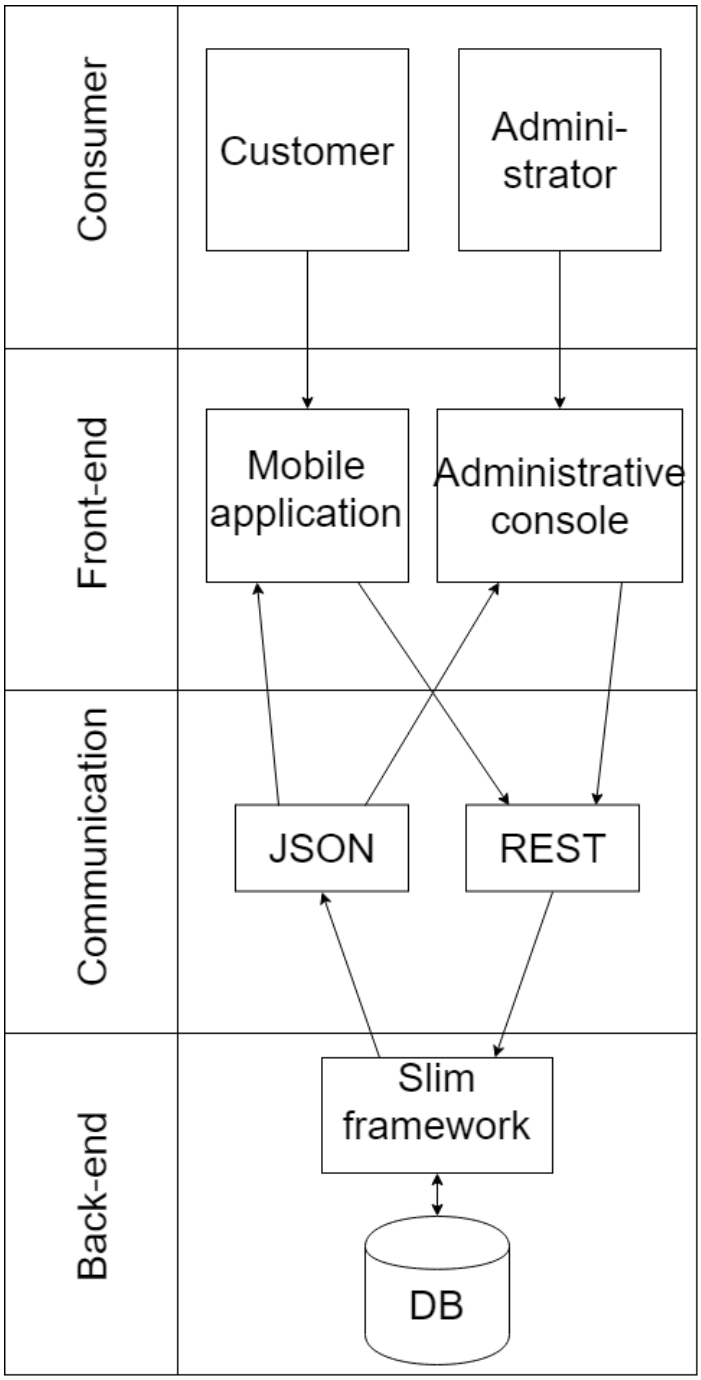

Figure 4. *Layers in project. (Source: author)*

2.7 Database

The database consists of 5 tables. Table 1 contains information about events. Database in first normal form. Events must always have a start date, but they do not always have an end date for the event. Therefore, in the database, “end_date” can be null, or if the event is one day, then “date” and “end_date” will be filled with the same date.

When parsing events, events that already exist in the table will not be added to the table, because the events table “link” received a unique index, so records with the same link will be ignored.

Table 1. *Table of events*

Name	**Index**	**Type**	**Null**	**Advanced**
id	Primary key	int(11)	No	Auto increment
title		varchar(255)	No	
date		date	No	
end_date		date	Yes	
time		time	Yes	
location		varchar(255)	Yes	
city		varchar(255)	Yes	
country		varchar(255)	Yes	
genre	Foreign key	tinyint(3)	No	
info		varchar(255)	Yes	
link		varchar(512)	No	Unique
img		text	Yes	
price		varchar(255)	Yes	
des		mediumtext	Yes	

As Table 2 shows if add another genre with the same name to the table with genres, genre will be ignored.

Table 2. *Table of genres.*

Name	**Index**	**Type**	**Null**	**Advanced**
id	Primary key	int(11)	No	Auto increment
genre		varchar(255)	No	Unique
description		varchar(255)	Yes	

In the table “logs” (Table 3) contains information about the device of the user. Values are added to this table “logs” when the mobile or administrative console is launched. The IP address is converted as an integer by using the PHP “ip2long()” function in back-end part, as shown in Listing 4.

Listing 4. *Example of using ip2long() function.*

```
ip2long('192.168.0.1');
Output: 3232235521
```

Table 3. *Table of logs.*

Name	**Index**	**Type**	**Null**	**Advanced**
id	Primary key	int(11)	No	Auto increment
ip		int(11)	No	
proxy		varchar(255)	No	
time		timestamp	No	
country		varchar(255)	No	
region		varchar(255)	No	
city		varchar(255)	No	

agent		text	No	

Table “reports” (Table 4) contains records with complaints about the event. These entries are added from the mobile application by customer.

Table 4. *Table of reports.*

Name	**Index**	**Type**	**Null**	**Default**	**Advanced**
id	Primary key	int(11)	No		Auto increment
event_id	Foreign key	int(11)	No		
report		varchar(255)	No		
time		timestamp	No	current_timestamp()	
old		tinyint(1)	No	0	

The table with warnings (Table 5) includes edited events that will no longer be displayed in the administrative console as events that have an error but may reappear in program if the field “edited” is “0”.

Table 5. *Table of warns.*

Name	**Index**	**Type**	**Null**	**Default**	**Advanced**
id	Primary key	int(11)	No		Auto increment
event_id	Foreign key	int(11)	No		
timestamp		timestamp	No	current_timestamp()	ON UPDATE CURRENT_TIMESTAMP()

edited		tinyint(1)	Yes		

2.8 Tables relationships

There are only three relationships between tables (Figure 5):

1. Table “events” and “reports” - many to many;
2. Table “events” and “warns” - one to one;
3. Table “events” and “genres” - many to one;

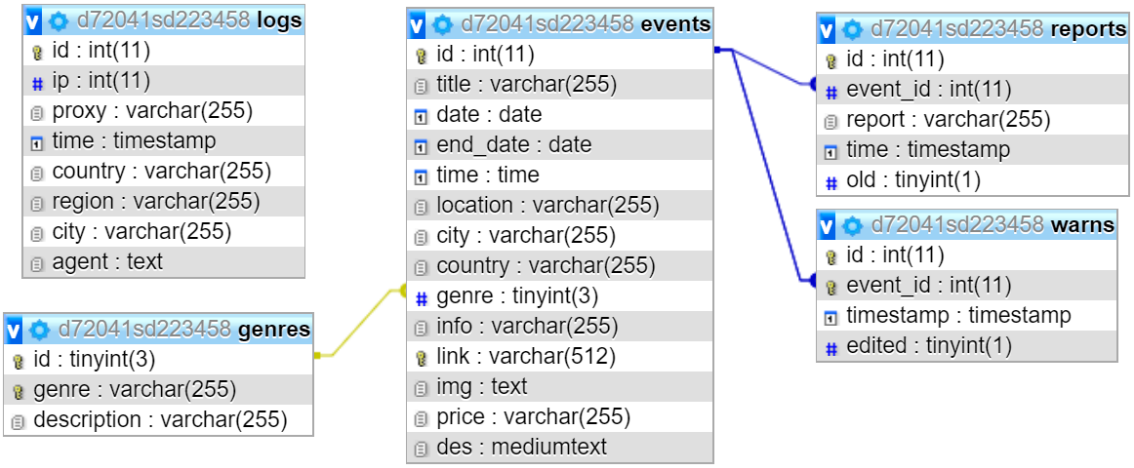

Figure 5. *Tables relationships from phpMyAdmin. (Source: author)*

2.9 Routes

Routes in the RESTful API are represented by URIs. The index route for the API returns all the available information for the API (Listing 5). Routes should be unique.

Mobile application can use main routes:

Events:

- GET /api/v1/events - can be used to get events;
- GET /api/v1/events/{event id} - can be used to get event by event id;
- GET /api/v1/events/find/{date}[/{city}[/{genres}]] - can be used to find events by date, city and genre (Listing 5);
- GET /api/v1/events/city - can be used to download a list of current cities, where the events will take place;
- GET /api/v1/events/random - can be used to receive a random event;

Reports:

- POST /api/v1/reports/create - can be used to send complaints to the event;

Logs:

- GET /api/v1/ip - can be used to collect information about the user's, such as name of device, location;

Listing 5. *Example of using route GET /api/v1/events/find/{date}[/{city}[/{genres}]]*

```
https://fto.ee/api/v1/events/find/today
https://fto.ee/api/v1/events/find/2018-06-19/tallinn
https://fto.ee/api/v1/events/find/tomorrow/all/0_5
```

Administrative console can use main routes:

Events:

- GET /api/v1/events - can be used to get events;
- GET /api/v1/events/{id} - can be used to get event by event id;
- GET /api/v1/events/city - can be used to download a list of current cities, where the events will take place;
- GET /api/v1/events/warnings - can be used to find an event with an error;
- POST /api/v1/events/create - can be used to create a new event and load it into the database;
- PUT /api/v1/events/update - can be used to update an event;
- DELETE /api/v1/events/delete - can be used to delete an existing event;

Online:

- GET /api/v1/online - can be used to get number of runs of Android application;
- GET /api/v1/online/day - can be used to get the number of runs of the Android app in 24 hours;

Reports:

- GET /api/v1/reports/{0/1} - can be used to receive new or old complaints about events;
- PUT /api/v1/reports/update - can be used to change the status of a complaint;
- DELETE /api/v1/reports/delete - can be used to remove a complaint;

Logs:

- GET /api/v1/ip - can be used to collect information about the user's, such as technique, location;

Genres:

- GET /api/v1/genres - can be used to get a current list of genres;
- POST /api/v1/genres/add - can be used to add a new genre to the database;

Files:

- POST /api/v1/upload - can be used to upload a file to a web server;
- POST /api/v1/deamon - can use to uploading new source to the web server;

Each route has been tested in SoapUI for the correct output, the method that is used and test load. Test load is modelling usage of a software program by simulating multiple users accessing into the program concurrently, approximately 150 connections per second. This test needed for show the error in back-end part.

2.10 Administrative console

An administrative console is developed to manage events in the database. In this application, user can view events, edit, delete, and add. To enter in program user IP address should be in the whitelist on hosting, then the application makes a request to GET /api/v1/ip and back-end adds the data to the table “logs”: IP address of the computer from which was made launch, time of entry, version of program and computer name. All of the functions associated with the database, use paths from the back-end part. The only one where native functions are used to communicate with back-end part is where checks the user when logging in to the application using MySqlClient. For handling JSON format used Json.NET in the program.

2.11 Functional requirements

Functional requirements will determine the functions that the application will perform. They correspond to the user's request and the task being solved.

Log in to the app:

- Description: The user enters the Windows-based program.
- Prerequisites: The user already has an account to log in and his IP address is in the whitelist.
- User response:
    - The user starts the program.
    - The user enters his username and password.

    - The user presses the “Enter” button.

- Result: To the user opens main windows of program.
- Alternative result:
    - If the user does not fill in the username or password fields, an error is displayed. If incorrect data is entered, an error is displayed.
    - The user can exit the program by pressing the “Exit” button.

Log out from the app:

- Description: The user exits the program.
- Prerequisites: The user has already entered the program under his name and password.
- User response:
    - The user clicks on the cross in the upper right corner.
- Result: The user closed the program.

Add new event:

- Description: The user adds a new event to the database.
- Prerequisites: The user is logged in.
- User response:
    - The user clicks the “New event” button in the “Main” tab.
    - In the window that appears, the user enters data.
    - The user clicks on the “Add” button.
- Result: The event is added to the database and is displayed in the mobile application.

Edit event:

- Description: The user modifies the data in the selected event.
- Prerequisites: The user is logged in and selected “Search” tab.
- User response:
    - Writes a keyword in the text box
    - Selects the appropriate event and clicks on it.

    - The user edits the data.
    - The user clicks "Edit".
    - In the window that appears, the user confirms his action.
    - The user clicks "Yes".
- Result: Data changed in the database.

Delete event:

- Description: The user deletes the event from the database.
- Prerequisites: The user is logged in and selected event.
- User response:
    - The user right-clicks on the “Edit” button and clicks “Delete this event”.
    - The user confirms the deletion.
- Result: The event was removed from the database.

Force update events:

- Description: Add new events to the database.
- Prerequisites: The user is logged in and selected “Main” tab.
- User response:
    - The user clicks on the “Force update events” button.
    - Confirms execution by pressing the “OK“ button.
- Result: New events from sources are added to the database.

Search by keyword:

- Description: The user searches for an event that has a keyword.
- Prerequisites: The user is logged in.
- User response:
    - ○ The user selects the “Search” tab.
    - The user types a keyword into a text box.
- Result: A list of events that contain the keyword.

Watch new and old reports:

- Description: The user is looking at old or new reports.
- Prerequisites: The user is logged in and selected the “Reports” tab.
- User response:
    - The user selects “New” or “Old” report.
- Result: Depending on what the user selects, they are shown old or new reports.

Show events where is problems:

- Description: Problematic events are shown to the user.
- Prerequisites: The user is logged in.
- User response:
    - The user selects the “Main " tab.
- Result**:** Problematic events will appear in the right part of the program.
- Alternative result:
    - If there are no problematic events in the database, they will not appear.

Edit JSON keys:

- Description: The user changes the old key to the new JSON key.
- Prerequisites: The user is logged in and selected “New sources” tab.
- User response:
    - In the upper text box, the user types a link to the source that outputs the JSON format.
    - The user clicks on the “Check JSON” button.
    - In the left pane, select the desired key and right click.
    - In the menu that appears, the user clicks on “Rename key”.
    - The user enters a new name and presses the “Rename” button.
- Result: The selected key changes to the new name.
- Alternative result:

- If the user selects “Delete key” after the right click, it is deleted from the text field.

All functional requirements were manually tested by author.

2.12 Non-functional requirement

Non-functional requirements are application limitations and they define characteristics.

- Windows application cannot run multiple copies in one computer.
- The application must run on Windows 7 and higher.
- The application should not be installable, but work after downloading from the Internet.
- An unauthorized user must not log in to the application.
- The application must have a fast response interface.

To check non-functional requirements the author manually tested them. For the first requirement, the first copy of the program was launched, and then the second. After that appears a warning that program already running. For the second requirement, author run the program on Windows 10, then in Windows properties changed compatibility mode to Windows 7 and program was launched, and then tested on genuine Windows 7. For the third requirement, the program needed all the necessary libraries in the application root folder. For the fourth requirement, if the user entered wrong credentials then the warning should appear. For the last requirement, every operation needs to complete within 10 seconds period to meet the fast response interface requirements (Nielsen 2009).

2.13 GUI

With the first encounter, the user who launched the program, this is the login window (Figure 6). If the user entered the correct login and password to user opens main window (**Figure 7**). In the main menu user can view a diagram there can see multiple launched mobile application per hour in 24 hours. The X-axis shows hours from 0 a.m. to 12 p.m. The Y-axis shows the maximum launched mobile applications per hour.

The main window has all the important information for the user, such as a list of events that have errors. There are buttons to add a new event and view the statistics generated by Webalizer. The "Force update events" button adds new events from sources.

In the “Search” tab (**Figure 8**), the user can enter keyword into text box and search automatically, but he can choose from which date to end date keyword will be searched.

In the “Reports” tab (**Figure 9**), the user can view new and old reports, open an event for which a complaint was received or send a complaint to the old ones. It is possible to delete the old report, need to right-click to display the menu and click on “Delete report”.

In the “New sources” (**Figure 10**) tab, the user can insert a link to the source that outputs JSON format. The large text box displays the JSON text that the link outputs. If the keys do not match with database table “events” (Table 1), user can change them by clicking on any highlighted key. To send JSON to the database, click on the “Insert JSON into DB” button. If user want to save new source, he adds a link to daemon by clicking on the “Save and send to updater” button. In the window that created a new event (**Figure 11**) in the field for cities loaded all cities which are already exists in database. For genres are also loaded actual.

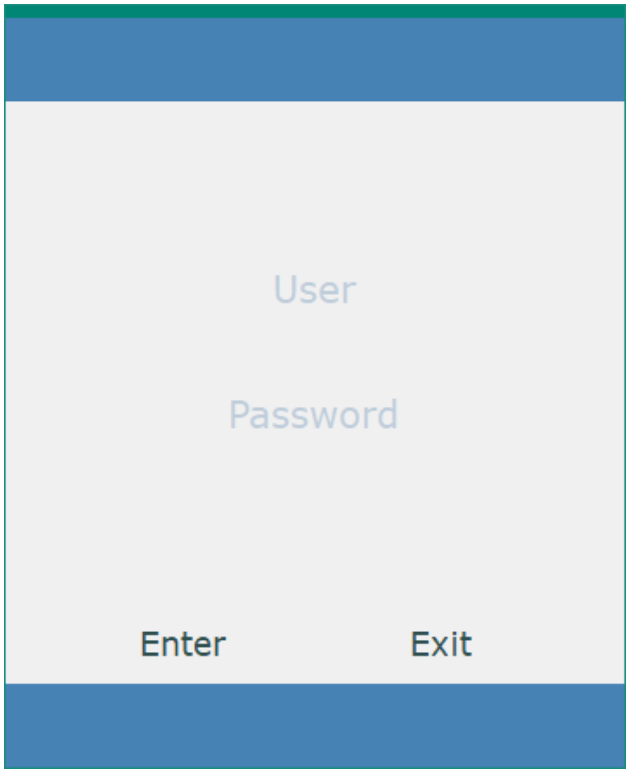

Figure 6*. Log in window. (Source: author)*

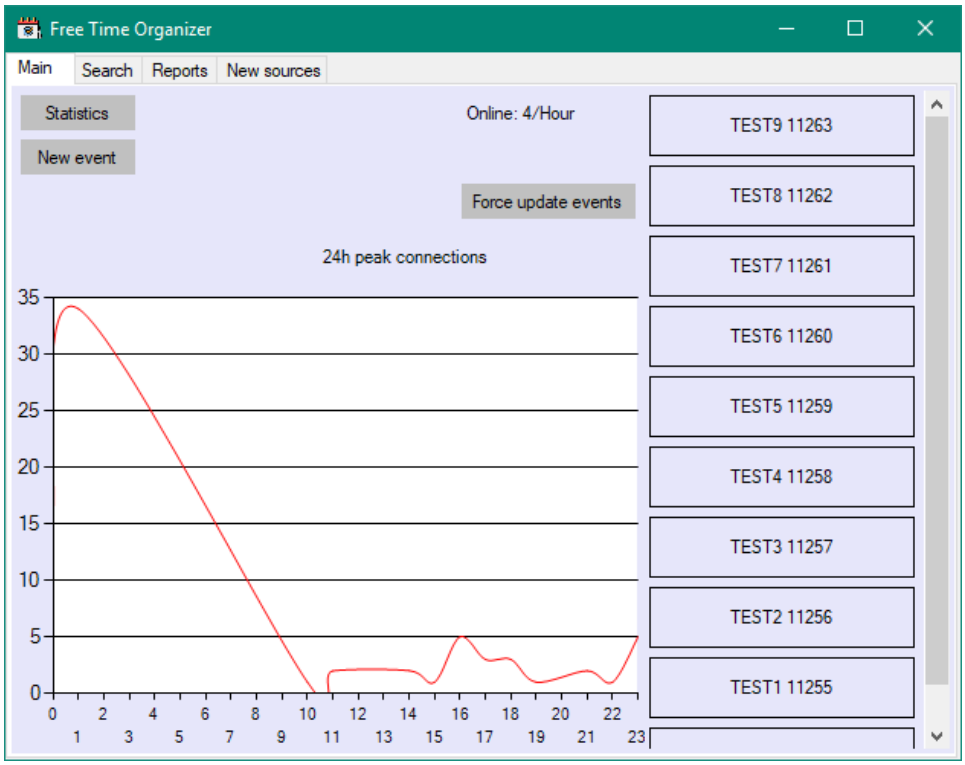

Figure 7. *Main window. (Source: author)*

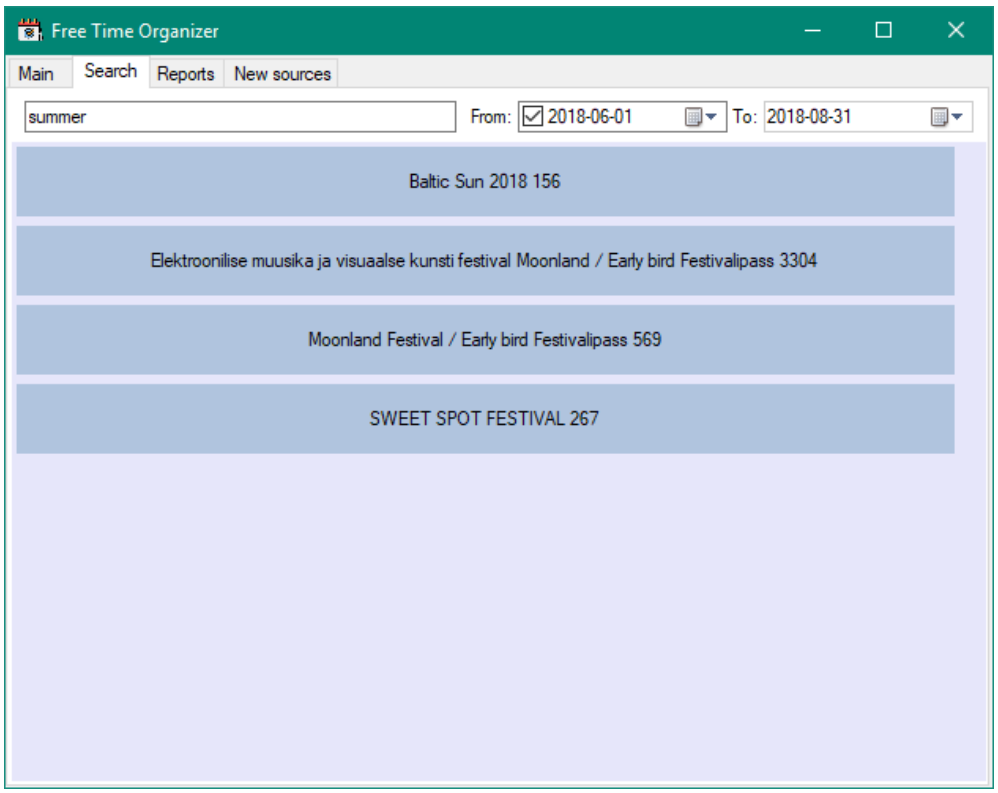

Figure 8. *Search tab. (Source: author)*

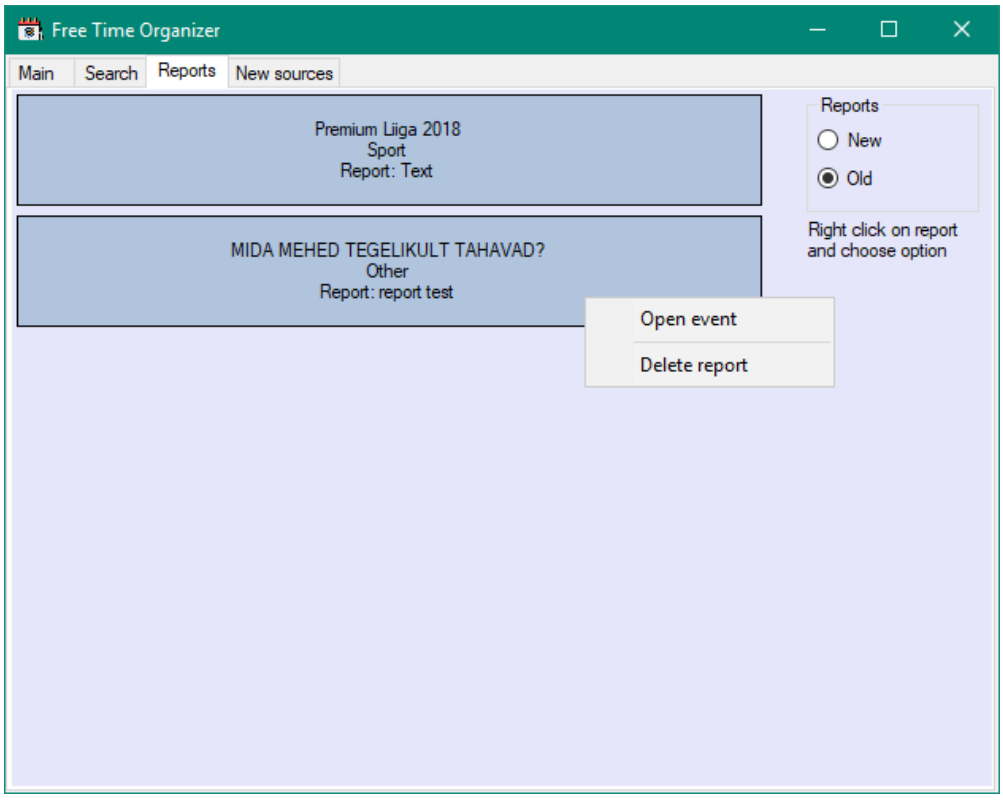

Figure 9. *Reports tab. (Source: author)*

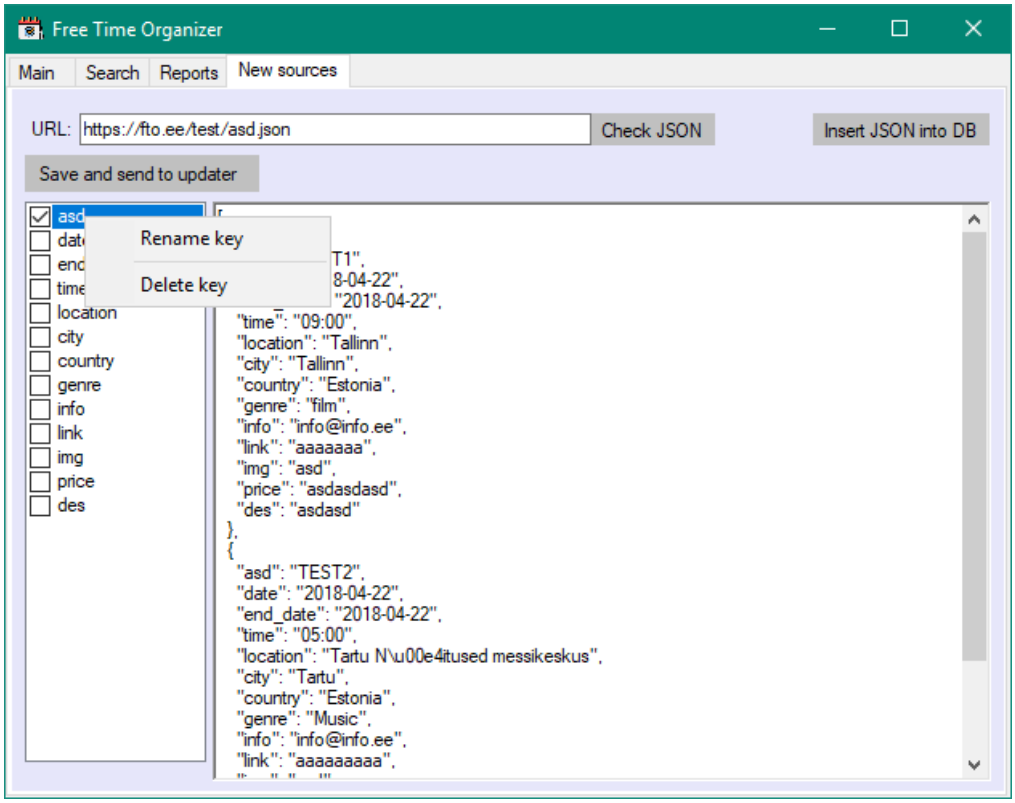

Figure 10 *New sources tab. (Source: author)*

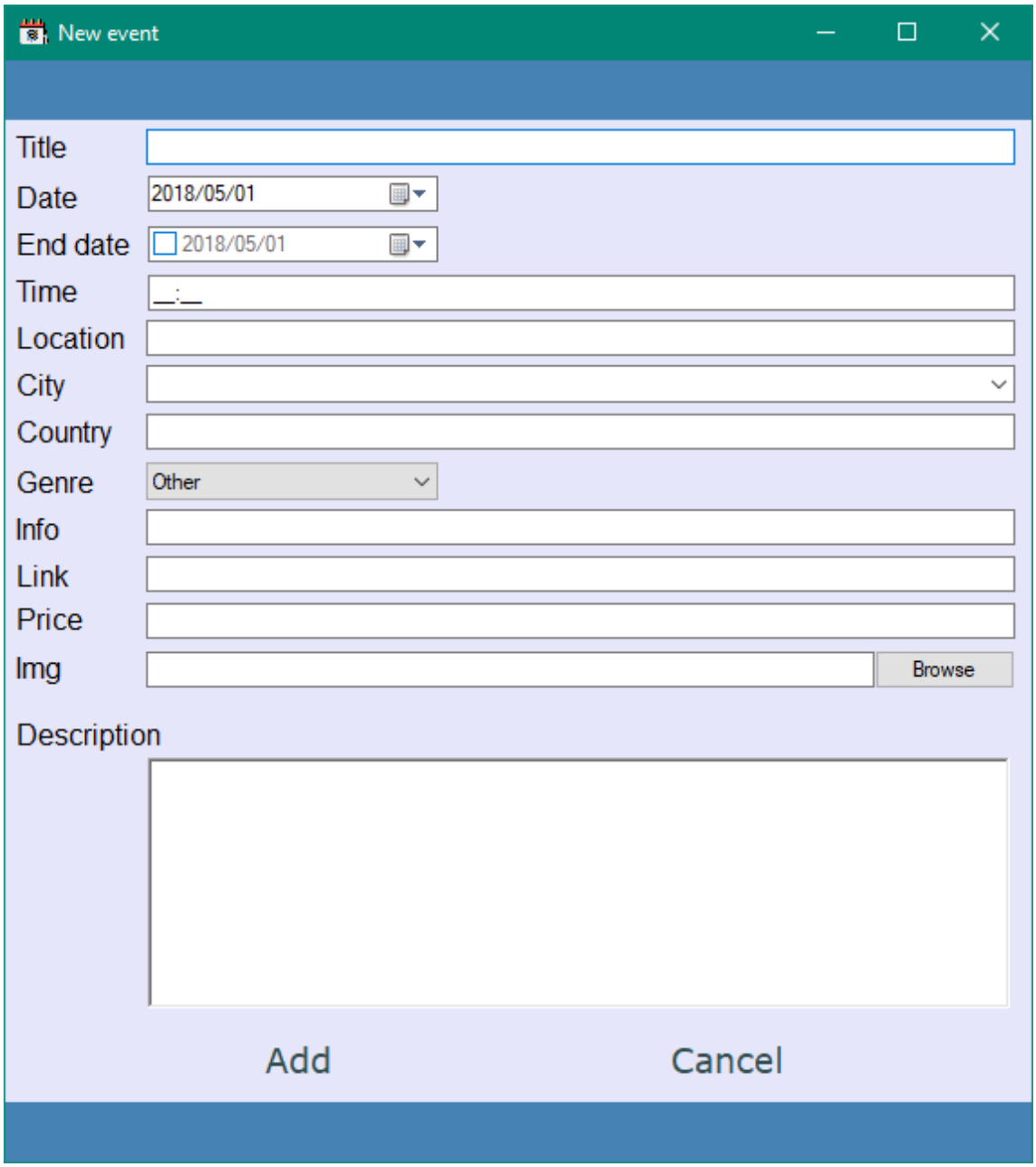

Figure 11. *New event window. (Source: author)*

CONCLUSION

The main task of this thesis was the development of back-end-part for Android-application "Free Time Organizer", which can help users find the information about local events of interest for any of the cities available in the application database.

The main task of the back-end-part of the application was to prepare the finished lists of events for the front-end-part of the application, with events filtered and sorted according to the user preferences by date, city or type. In addition, users have the option to send a complaint about the event, if they find the information inappropriate. Events can be loaded into the database from JSON, RSS or HTML data sources from the relevant information sites. Data exchange between front-end and back-end parts of the application is conveyed by REST queries and JSON replies, and protected with SSL encryption protocol.

The second part of the work was to create an administrative console for the database to add and edit information flows about events. With the console, the administrator of the database can view statistics, add new events, edit events, warn about problems in events, view user complaints, and add new sources in JSON format.

Thus, all the tasks set at the beginning of the work were completed, and the goal of the work was achieved. Back-end-part of the application was written in PHP programming language using the Slim framework. The database was created using MySql database system and maintained using phpMyAdmin tool. The administrative console for the database was written in C# programming language.

In the future, the author plans to add a web version of the administrative console using the progressive web application technology to be able to access administrative console from anywhere. In addition, the author wants to add many more dataflow sources and types to the database to be able to provide the application service for new cities and, possibly, even countries.

RESÜMEE

Selle lõputöö eesmärgiks oli arendada back-end-i osa sündmuste otsingu mobiilse rakenduse jaoks Android platvormile, samuti arendada väikse halduskonsooli sündmuste voogude lisamiseks andmebaasi. Back-end-i eesmärgiks oli koguda sündmuste informatsiooni erinevatest allikatest ja laadida need andmed andmebaasi, et teha neid hõlpsasti kättesaadavaks rakenduse front-end-i osa jaoks. Halduskonsooli eesmärk oli anda andmebaasi administraatorile mugava vahendi infovoogude toimetamiseks ja lisamiseks andmebaasi.

Rakenduse back-end-i peamine ülesanne oli valmistada front-end-i osa jaoks valmis nimekirju sündmustest, filtreeritutud ja järjestatud aja, asukoha või tüübi järgi vastavalt kasutajate eelistustele. Lisaks antakse kasutajale võimalus saata kaebuse sündmuse kohta, kui on põhjust arvata, et tegemist on kahtlase informatsiooniga. Sündmused laaditakse andmebaasi erinevate infovoogude kammimise läbi. Praegusel kujul oskavad back-endi skriptid lugeda infovoogusid JSON, RSS ja HTML formaadis. Front-end-i ja back-end-i osade omavaheline andmevahetus toimub REST ja JSON tehnoloogiate abil. Front-end pärib back-end-ilt info REST päringu kujul ja saab vastuse JSONi objekti näol. Andmevahetus on seejuures kaitstud SSL-i krüpteerimisprotokolli abil.

Teine osa tööst oli andmebaasi halduskonsooli arendamine, mille abil saab teha mitu erinevat toimingut ja vaadata andmebaasi kasutust. Selle halduskonsooli abil saab andmebaasi administraator lisada andmebaasi uusi andmevoogusid JSON-formaadis, parandada vigu andmetes, vaadata rakenduse päringute statistikat, saada teada probleemidest sündmustes ja vaadata kasutajate kaebusi sündmuste kohta.

Seega, on kõik töös püstitud ülesanded täidetud ja töö eesmärk on saavutatud. Back-end-i osa on kirjutatud PHP keeles kasutades Slim raamistikku. Andmebaas on loodud MySql andmebaasisüsteemis, andmebaasi administreerimine toimus phpMyAdmin tööriista abiga. Windowsi rakendus oli kirjutanud C# keeles.

Tulevikus plaanib autor lisada back-end-ile võime aru saada rohkemast andmevoogude hulgast, lisada andmebaasi palju uusi andmevoogusid ja lisada andmebaasi halduskonsoolile ka veebiliides.

REFERENCES

Cambridge University Press 2018a. *Front end.* Available at https://dictionary.cambridge.org/dictionary/english/front-end, accessed May 8, 2018.

Cambridge University Press 2018b. *Back end*. Available at https://dictionary.cambridge.org/dictionary/english/back-end, accessed May 8, 2018.

Cambridge University Press 2018c. *API.* Available at https://dictionary.cambridge.org/dictionary/english/api, accessed May 8, 2018.

Cambridge University Press 2018d. *Android*. Available at https://dictionary.cambridge.org/dictionary/english/android, accessed May 8, 2018.

The PHP Group 2018a. *What is PHP?*. Available at http://php.net/manual/en/intro-whatis.php, accessed April 30, 2018.

The PHP Group 2018b. *Pcntl_fork.* Available at http://php.net/pcntl_fork, accessed April 30, 2018.

Microsoft 2018. *Editing Code in Visual Studio IDE*. Available at https://www.visualstudio.com/vs/features/ide/, accessed April 30, 2018.

Smartbear 2018. *SoapUI*. Available at https://www.soapui.org/docs/functional-testing/getting-started.html, accessed May 20, 2018.

Brockschmidt Kraig, Myers Alfred 2018. *NuGet*. Available at https://docs.microsoft.com/en-us/nuget/what-is-nuget, accessed May 20, 2018.

Oracle Corporation 2018. *What is MySQL?*. Available at https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html, accessed April 30, 2018.

Slim Framework Team 2018. *Slim*. Available at https://www.slimframework.com/, accessed April 30, 2018.

phpMyAdmin contributors 2018. *About*. Available at https://www.phpmyadmin.net/, accessed April 30, 2018.

DigiCert 2018. *What is an SSL certificate?* Available at https://www.digicert.com/ssl/, accessed April 30, 2018.

JSON 2018. *Introducing JSON*. Available at https://json.org/, accessed April 30, 2018.

Barrett B. 2014. *The Webalizer*. Available at http://www.webalizer.org/, accessed April 30, 2018.

AS Piletilevi 2018. *Piletilevi RSS*. Available at https://www.piletilevi.ee/est/uldinfo/rss/, accessed April 30, 2018.

Zone Media LLC 2018. *Compare Web Hosting plans*. Available at https://www.zone.ee/en/web-hosting/compare/, accessed April 30, 2018.

Chen S.C., Schlick John, Carroll Rus 2014. *PHP Simple HTML DOM Parser*. Available at http://sourceforge.net/projects/simplehtmldom/, accessed May 1, 2018.

Jeffries E. Ronald 2011. *What is Extreme Programming?*. Available at https://ronjeffries.com/xprog/what-is-extreme-programming/, accessed May 8, 2018.

Nielsen Jakob 2009. *Powers of 10: Time Scales in User Experience.* Available at https://www.nngroup.com/articles/powers-of-10-time-scales-in-ux/, accessed May 21, 2018.

APPENDICES

Appendix 1. Source code downloads

- Administrative console – https://fto.ee/fto.zip;
- Back-end – https://fto.ee/back-end.zip;

To enter in administrative console needs IP in whitelist on hosting.