

TARTU UNIVERSITY
FACULTY OF SOCIAL SCIENCES

NARVA COLLEGE
STUDY PROGRAM “INFORMATION TECHNOLOGY SYSTEMS
DEVELOPMENT“

Aleksandr Džikajev
**DEVELOPING A FRONT-END PART OF AN APPLICATION FOR PARTICI-
PATION IN LOCAL EVENTS**

Diploma thesis

Supervisor: Assistant Andre Säask

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 license to reproduce thesis

I, Aleksandr Dzikajev (date of birth: 29.08.1996),

1. herewith grant the University of Tartu a free permit (non-exclusive license) 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 a front-end part of an application for participation in local events”, supervised by assistant Andre Säask.
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 license does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Narva, 23.05.2018

CONTENTS

| | |
|---------------------------------------|----|
| INTRODUCTION | 7 |
| 1 USED TECHNOLOGIES | 9 |
| 1.1 Android operating system | 9 |
| 1.2 Android Studio | 9 |
| 1.3 Android application..... | 9 |
| 1.4 Java programming language | 9 |
| 1.5 XML | 10 |
| 1.6 Gradle | 10 |
| 1.7 Android Emulator..... | 10 |
| 1.8 VCS | 10 |
| 1.9 REST | 11 |
| 1.10 JSON | 11 |
| 1.11 Soapui..... | 11 |
| 1.12 XP..... | 12 |
| 2 COMPETING APPLICATIONS | 13 |
| 2.1 Eventbrite | 13 |
| 2.2 Eventtus – Events App | 14 |
| 2.3 Rukkus Event Tickets | 17 |
| 2.4 All Events in city..... | 19 |
| 2.5 Concerts, Theater – Ticketea..... | 21 |
| 2.6 Conclusion..... | 22 |
| 3 APPLICATION ARCHITECTURE | 24 |
| 3.1 Design of an application..... | 24 |
| 3.2 GUI..... | 25 |
| 3.3 Activities in application | 26 |

| | | |
|-------------------------------|---------------------------------------|----|
| 3.3.1 | Welcome activity | 29 |
| 3.3.2 | Home activity..... | 29 |
| 3.3.3 | Event details activity..... | 30 |
| 3.3.4 | Report activity..... | 32 |
| 3.3.5 | Genre setting activity | 33 |
| 3.3.6 | City setting activity | 34 |
| 3.3.7 | Date setting activity | 35 |
| 3.4 | Connection to back-end..... | 36 |
| 3.4.1 | Events..... | 36 |
| 3.4.2 | Reports | 36 |
| 3.4.3 | Genres | 36 |
| 3.4.4 | URL links testing | 37 |
| 3.5 | Event filters | 37 |
| 3.6 | JSON parsing | 37 |
| 3.7 | Functional requirements..... | 38 |
| 3.8 | Non-functional requirements | 39 |
| 3.9 | Publishing in Google Play market..... | 39 |
| 3.10 | Application testing | 40 |
| 3.10.1 | Manual testing..... | 40 |
| 3.10.2 | Testing with Google Play tools..... | 41 |
| CONCLUSION..... | | 43 |
| RESÜMEE..... | | 45 |
| REFERENCES | | 47 |
| APPENDICES | | 48 |
| Appendix 1. Source code | | 48 |

TERMS AND NOTATIONS

XML – Extensible Markup Language.

GUI – Graphical User Interface.

API – Application Program Interface. In computer programming, an application programming interface (API) is a set of subroutine definitions, protocols, and tools for building application software. In general terms, it is a set of clearly defined methods of communication between various software components. (API 2018a)

REST (Representational State Transfer) - an architectural style of interaction between components of a distributed application on a network.

URL – Uniform Resource Locator. Reference to a web resource which specifies its location on a computer network.

RSS – Rich Site Summary.

IDE – Integrated Development Environment.

GPS – Global Positioning System.

JSON – JavaScript Object Notation.

Google Play – The market to download applications on the android platform.

Back-end – 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. (Back-end 2018b)

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

APK – Application Package Kit

INTRODUCTION

In the modern world, time is one of the most valuable and important resources, and therefore a large number of services such as banking, shopping or even rent payment go online or become mobile applications to allow people to access these services from anywhere in the world, where people can assess the Internet.

In this paper the author plans to focus on granting people access to such a service as information about educational, entertainment, sport and other such events, which occur in their city and country. At the moment there is no single environment in which it would be possible to find structured and sorted information about all local Estonian events. Therefore, the author decided to create an application, which would contain all possible information about all events taking place in Estonia, thereby allowing the user to save time in searching events by appropriate time, event type or genre and the location of the event.

In view of the large amount of work, the project was divided between two authors, one of which would deal with the front-end part of the project, and another author would work on the back-end part of the project. The author of this paper took the development of the front-end part.

Thus, the aim of the thesis is to develop a mobile application for the Android platform, which would provide its users with a customizable flow of information about local events. The application would receive the data stream from the back-end part of the application using REST technology, which will be provided by the second author.

Based on the expected functionality the main functions of the application should be:

- displaying a scrollable event flow, where each event is a separate block;
- the ability to see more detailed information of each event by tapping on it;
- the ability to sort and filter events by type, time and location;
- the ability to go to the webpage of the event, if it exists;
- the opportunity to proceed to buying a ticket for a chosen event, if such an option exists;
- the ability to add an event to the calendar of the mobile device;
- the opportunity to share the event with friends through social networks or other communication channels;

- the opportunity to send a complain about an event;

To fulfil the purpose of the thesis, the author must perform the following tasks:

- Develop the design of the application, taking into account the requirements of the convenience of the user interface such as ease of use and possibility to skip registration;
- Together with another author create a data exchange model between the back-end and front-end parts of the application;
- Implement all the functional requirements in the application;
- Upload the application to the Google Play;
- Manually test the application.

There are hopes that the application will be used not only in Estonia, but also abroad, when we add corresponding news feeds to the application.

The paper consists of an introduction, two chapters, a conclusion, a list of terms, a list of used literature and an appendix with the source code.

The first chapter describes technologies used by the author in the project, the second chapter contains a brief description of already existing applications with similar functionality, and the third chapter describes the process of development of the project and the cooperation with another author, who develops the back-end part of the application.

1 USED TECHNOLOGIES

This chapter describes the technologies, which the author uses in the project and explanations why the author chose those technologies.

1.1 Android operating system

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets. (Android... 2018a)

Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013. As of May 2017, it has over two billion monthly active users, the largest installed base of any operating system, and as of 2017, the Google Play store features over 3.5 million apps. (Android... 2018a)

Author chose Android because it is currently the most popular smartphone operating system. Also, Java is the author's general programming language

1.2 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. (Google 2018)

Author chose this IDE, because he thinks that it is the most convenient IDE for Android application development.

1.3 Android application

An Android app is a software application running on the Android platform. Because the Android platform is built for mobile devices, a typical Android app is designed for a smartphone or a tablet PC running on the Android OS. (Techopedia 2018)

1.4 Java programming language

Java is a modern, strongly typed, class-based object-oriented programming language. Java is default language for the development of Android applications in Android Studio IDE. Also, the experience of writing an application in this language is very important to the author and he is sure that Java will be useful to him in the future.

1.5 XML

Extensible Markup Language, abbreviated XML, describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879]. By construction, XML documents are conforming SGML documents.

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure. (Extensible Markup Language... 2008)

1.6 Gradle

Gradle is an open-source build automation system that builds upon the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language (DSL) instead of the XML form used by Apache Maven for declaring the project configuration. Gradle uses a directed acyclic graph ("DAG") to determine the order in which tasks can be run. (Gradle 2018b)

The author chose Gradle because it is the default project builder in the Android Studio.

1.7 Android Emulator

The Android Emulator is a virtual device which simulates an Android OS based device, it allows a programmer to run Android applications on a computer and use computer resources. It comes with predefined configurations for popular devices.

There is a lot of emulators from other developers, but the author only uses Android Studio default emulator, because it is free, easy to use and author already knows how to use it.

1.8 VCS

A component of software configuration management, version control, also known as revision control or source control, is the management of changes to documents, computer programs, large web sites, and other collections of information. Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or

simply “revision”. For example, an initial set of files is “revision 1”. When the first change is made, the resulting set is “revision 2”, and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged. (Version control 2018c)

1.9 REST

REST technology is very popular among Android applications because, REST is a simple way to manage interactions between independent systems. REST allows to interact with clients as diverse as mobile phones, back-end servers and other websites.

REST technology perfectly suits for the application’s needs, since the front-end part should receive data quickly from the back-end and not parse information from sites directly, which would be unacceptably slow.

1.10 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, which 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)

1.11 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)

1.12 XP

It has been decided to choose extreme programming development method. XP is the ability to develop, when requirements for the project are not completely clear and difficult to imagine.

Below are listed main features of extreme programming (Jeffries 2011):

- At the start of the project, there is no precise plan and after each release the understanding of tasks increases;
- Short iterations, high frequency of releases. Working product after each iteration;
- Keep the code clean. It gives good scalability of the project. Refactoring;
- Each member of a team is responsible for the final product, even if they work on different parts of the product;

Weekly meetings with the second author were scheduled, where future problems and plans were discussed. Meetings usually lasted up to 2 hours. Also, there were smaller meetings, where were discussed design features and suggestions.

2 COMPETING APPLICATIONS

This chapter speaks about alternative applications, which already exist for the Android platform at the moment.

There are several applications on the Google Play market, which could be considered as alternatives to the author's project. In next subchapters, the author briefly describes these applications and explains why they are not ideal and why there is still room for a better solution.

2.1 Eventbrite¹

Eventbrite is the most popular application in its area with more than 5 million downloads. The application has an attractive user-friendly design. It requires registration or login through Facebook account. It allows a user to buy tickets or go to the source site of the event.

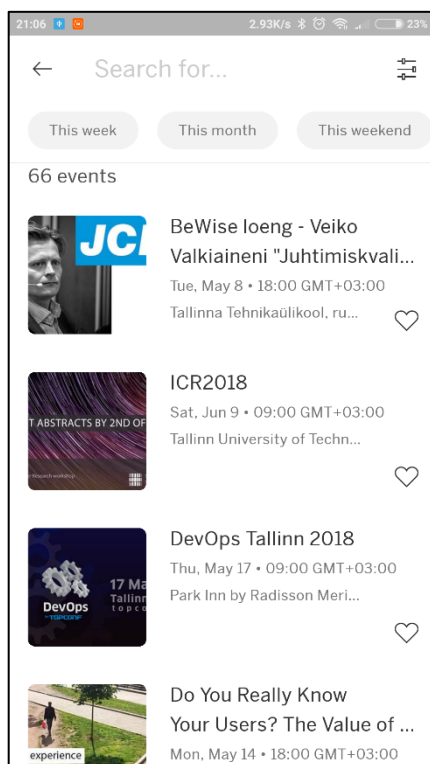


Figure 1. *Eventbrite events.* (Source: author)

¹ <https://play.google.com/store/apps/details?id=com.eventbrite.attendee>

Eventbrite is the one of two applications that showed events taking place in Estonia, but only limited to Tallinn. The application is working well in the United States, but in Europe and especially in Estonia it is much less useful. Most likely this is due to the fact that all the events the application takes from Facebook.

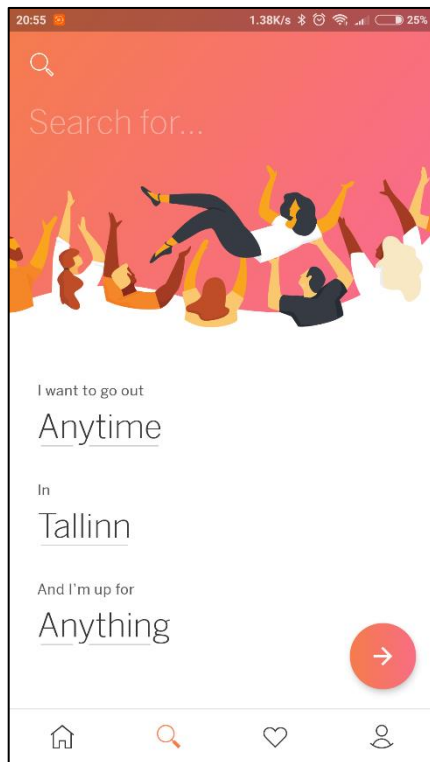


Figure 2. Eventbrite settings. (Source: author)

2.2 Eventtus – Events App²

Eventtus has similar functionality to the application of this thesis, but it only has sorting by genre, location and name of the event. The author thinks, that the design of the application is not entirely user-friendly and it is easy for its user to get confused. Application requires login through Facebook or registration. Unfortunately, the application does not find any events in Estonia.

² <https://play.google.com/store/apps/details?id=com.smartizer.eventtus.android>



Figure 3. *Eventtus login.* (Source: author)

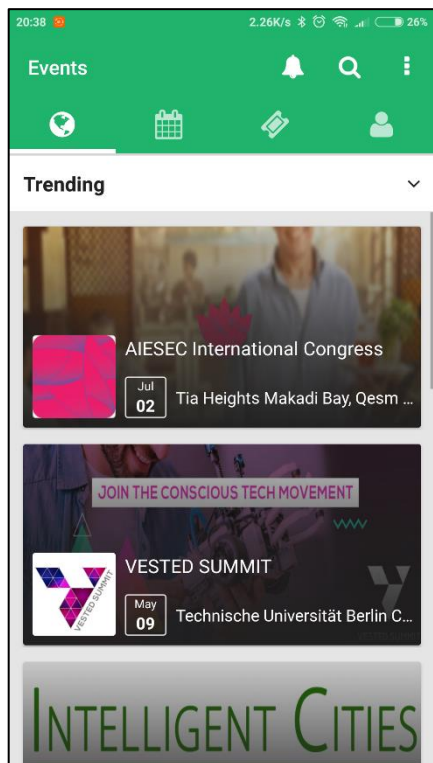


Figure 4. *Eventtus event list.* (Source: author)

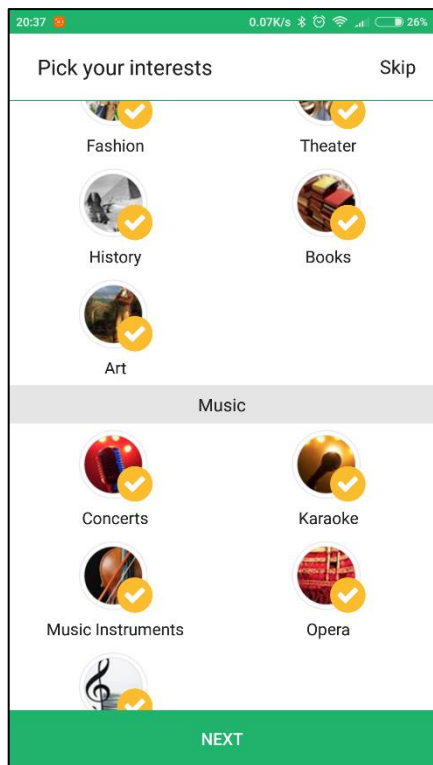


Figure 5. Eventtus event genres. (Source: author)

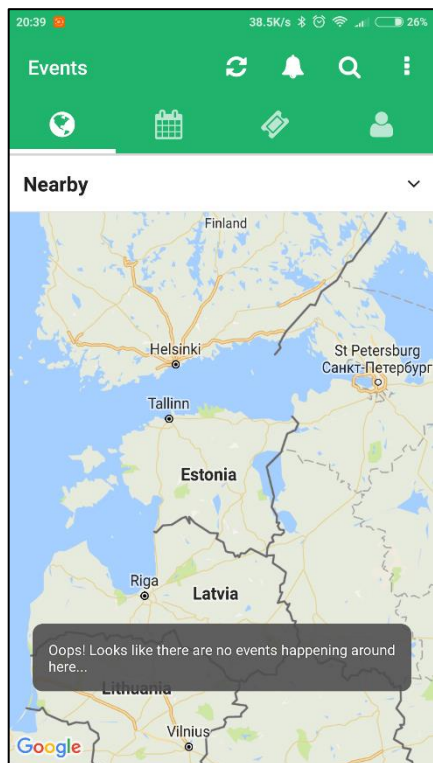


Figure 6. Eventtus location settings. (Source: author)

2.3 Rukkus Event Tickets³

The application has only 3 genres: concerts, sports and theater. There is a possibility to set the location manually or with the help of GPS. The application requires login through Facebook or registration. Unfortunately, it does not find any events in Estonia.

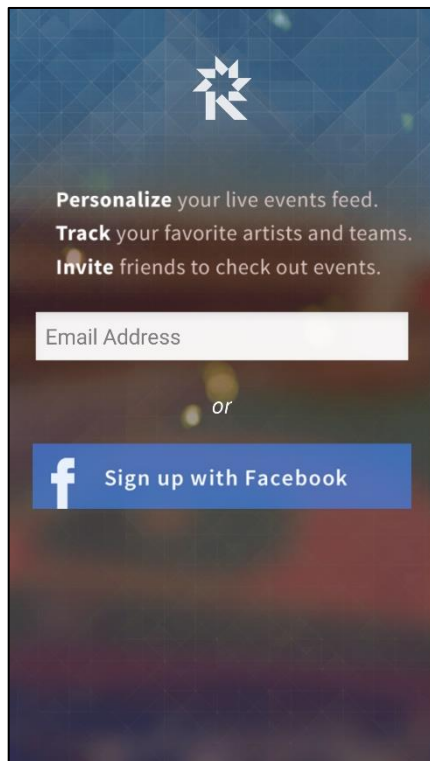


Figure 7. *Rukkus login. (Source: author)*

³ <https://play.google.com/store/apps/details?id=com.rukkus.app>

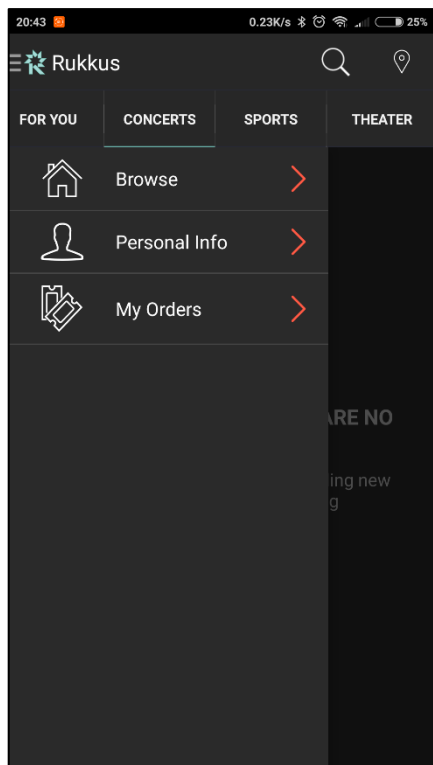


Figure 8. *Rukkus menu.* (Source: author)

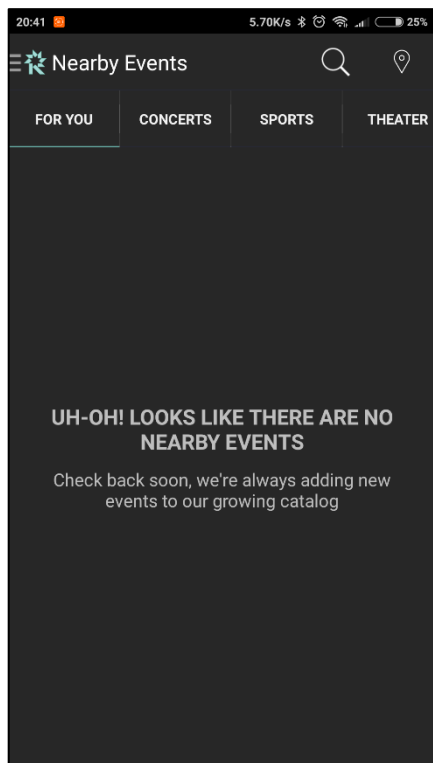


Figure 9. *Rukkus events.* (Source: author)

2.4 All Events in city⁴

In the “All Events in city” application users can bypass registration. There is filtering by dates, cities and genres, but not all events’ pages can provide a source website or a buying tickets service. There is no information about the price of the tickets, when tickets have prices. The application does not contain all the cities of Estonia, but it is nevertheless one of the main competitors for the author’s project.

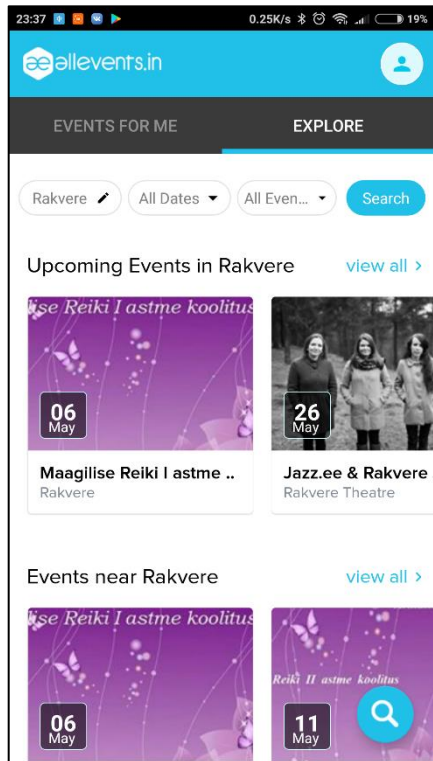


Figure 10. All Events in city event list. (Source: author)

⁴ <https://play.google.com/store/apps/details?id=com.amitech.allevnts&rdid=com.amitech.allevnts>

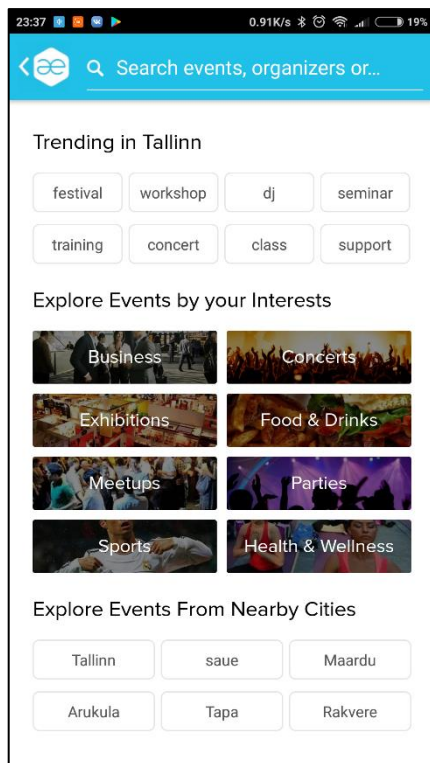


Figure 11. All Events in city filter settings. (Source: author)

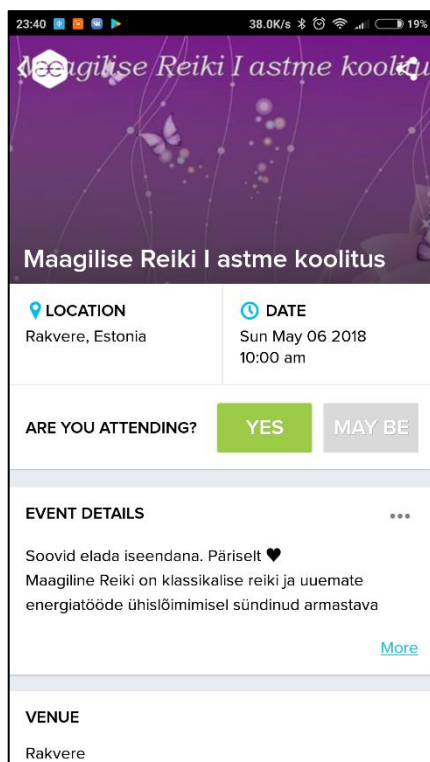


Figure 12. All Events in city event details. (Source: author)

2.5 Concerts, Theater – Ticketea⁵

There is a registration in the “Ticketea” application, but user can skip it. It has a fairly convenient navigation. The application does not show anything in Estonia. It has a big amount of event categories. It also has date filter settings.

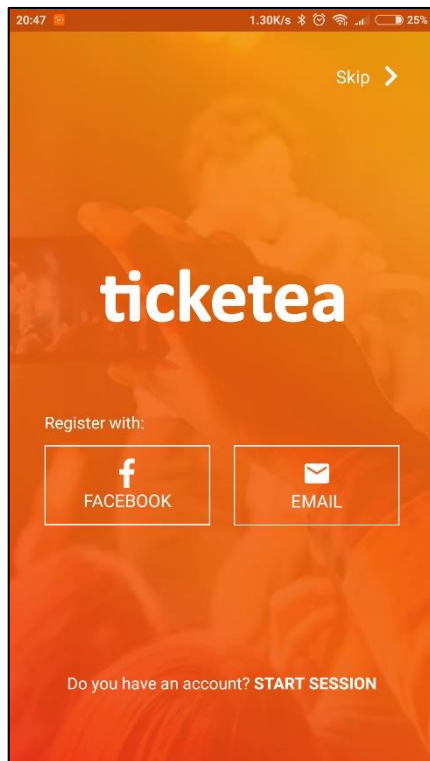


Figure 13. Ticketea login. (Source: author)

⁵ <https://play.google.com/store/apps/details?id=com.ticketea.geminis>

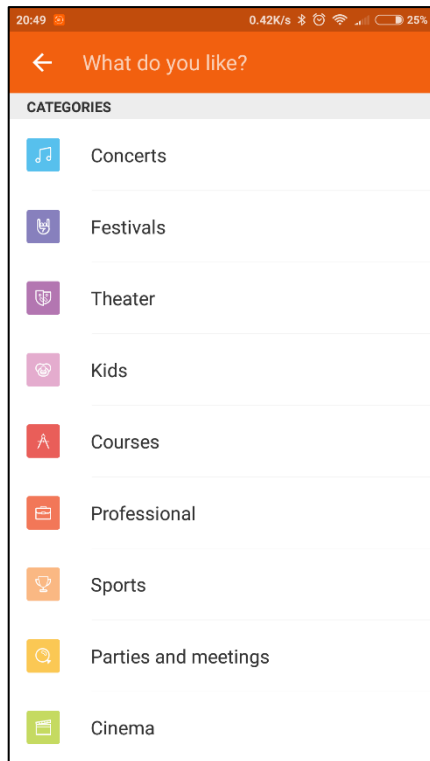


Figure 14. Ticketea filter settings. (Source: author)

2.6 Conclusion

As shown by table below (Table 1) the main competitor to author's application is "All Events in city". It has decent database which contains a few Estonian cities, but just like other competing applications they try to cover the whole world using Facebook events, which is not very effective in the case of Estonia, in author's opinion. Author believes, that this strategy alone will not provide a sufficient cover of all events and will cause a lack of information flows.

The author picked several good features from the competing application and implemented in his own solution to make it more suitable and reliable for Estonia.

Table 1. *Applications and comparing their functionality.*

| Application | Filter by genre | Filter by city | Filter by date | Requires registration | Proceed to buying tickets | Detailed information | Add event to calendar | Share event details | Events in Estonia | Events not only in Tallinn |
|-----------------------|-----------------|----------------|----------------|-----------------------|---------------------------|----------------------|-----------------------|---------------------|-------------------|----------------------------|
| Eventbrite | X | X | X | X | X | X | X | X | X | |
| Eventtus – Events App | X | X | | X | X | X | X | X | | |
| Rukkus Event Tickets | X | X | | X | | | | | | |
| All Events in city | X | X | X | | X | X | X | X | X | X |
| Ticketea | X | X | X | | X | X | | X | | |
| Author's solution | X | X | X | | X | X | X | X | X | X |

3 APPLICATION ARCHITECTURE

This chapter speaks about events, describes the structure of the project, the design and the process of developing the Android application.

3.1 Design of an application

It has been decided to choose 5 colors for GUI (Figure 15):

- #524449
- #89b7aa
- #ebe970
- #e79652
- #e15a58

All the colors the author selected using online Adobe Color CC⁶. According to literature and article the author read colors combine well and reflect the essence of the application. (Babich 2017)

The red color #E15A58 is applied for top toolbar. Brown or dark violet #524449 color stands for background of the application. And other colors are reserved for buttons, notifications and widgets, which will be added to the application in the future.



Figure 15. Colors of the application. (Source: author)

⁶ <https://color.adobe.com>

3.2 GUI

The first screen encountered by the user when the application is opened is the welcome screen which lasts a few seconds. After that, the user is moved to the main application screen which contains a scrollable view with event blocks (the author will call them cards hereafter). Each card contains brief information about one particular event and its picture (Figure 18).

Making a swipe from left to right or tapping on the corresponding icon in the upper left corner a user will open the side menu, where the user can find such menu items as:

- filtering by genre;
- filtering by city;
- filtering by date.

(Figure 19)

Also, menu has the item called “liked events” which the author plans to implement later.

There is also a “Random” menu item. By tapping on it, the user will get only one card with a random event for today on the main screen. Making a swipe from top to bottom the user can manually update the list of cards with events from the server.

By clicking on the menu item called “Genre”, the user will open the screen where he can select genres. They can be toggled on or off, depending on what the user would like to see. By pressing the back button, the settings are saved (Figure 25).

When user clicks on the menu item called “City”, the user will open screen with the city settings, where the user can enter only the city, which is in the database. When the user taps OK the city is added to the list and the user can enable or disable sorting for this city (Figure 26).

When the user taps on the menu item called “Date”, the date filter settings screen opens. It contains such options as: all time, today, tomorrow, this week, this month, or choose date using the drop-down calendar widget (Figure 27).

When the user presses back button, the data is saved and the user is transferred to the main screen with event cards filtered by the user’s choices.

Users can tap on the event card and then they will be transferred to the screen with detailed information about the event. On the screen with detailed information, the user can see all the information about the event such as:

- Event title;
- Time;
- Location;
- Price of the ticket, if any;
- Description of the event.

(Figure 20, Figure 21)

Users can go to the source page of the event by clicking on the floating button. There they can buy tickets to the selected event. Also, there are three menu buttons on top right of the screen:

- First button allows to add an event to the user's mobile device calendar (Figure 22);
- Second button opens a screen where the user can enter text and send a complain report about this event (Figure 24);
- Third button gives an opportunity to share the event with friends through social networks or other communication channels (Figure 23).

3.3 Activities in application

Android application consists of Java classes (Figure 16) and XML layouts (Figure 17).

Each Java class has its own purpose. Some of the classes are used to maintain the functionality of the application screens, others are used as data objects, as adapters, which have their own functions such as data processing or as background services. Visual parts of the application are not created by Java classes, but by XML layouts instead. XML layouts contain information such as button positions, text height, color, background color etc. Also, there are XML files, which store dimensions, texts and color variables.

This subchapter describes Java classes and XML layouts and how are they related.

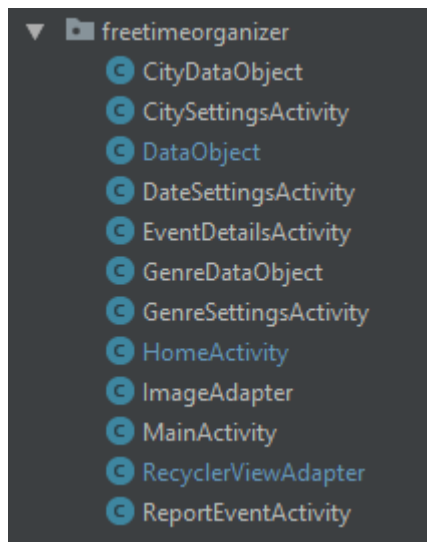


Figure 16. List of classes. (Source: author)

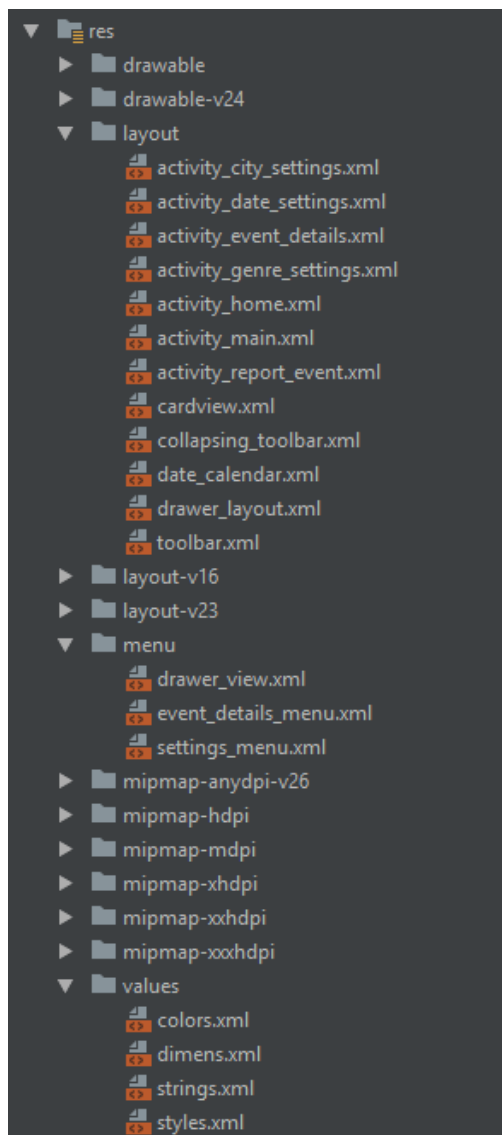


Figure 17. List of XML resources. (Source: author)

Table 2. *Relationship between activities and layouts.*

| Activity | Layout |
|-----------------------------|-----------------------------|
| MainActivity.class | activity_main.xml |
| HomeActivity.class | activity_home.xml |
| CitySettingsActivity.class | activity_city_settings.xml |
| DateSettingsActivity.class | activity_date_settigns.xml |
| GenreSettingsActivity.class | activity_genre_settings.xml |
| EventDetailsActivity.class | activity_event_details.xml |
| ReportEventActivity.class | activity_report_event.xml |

Table 3. *Adapters and data objects.*

| Class | Description |
|---------------------------|---|
| DataObject.class | Model of event data |
| CityDataObject.class | Model of data used in city filter settings |
| GenreDataObject.class | Model of data used in genre filter settings |
| ImageAdapter.class | Image adapter is used to draw pictures on the Home-Activity screen |
| RecyclerViewAdapter.class | Adapter which takes every single row of DataObject ArrayList and assigns it to each card on HomeActivity screen |

User interface of an Android application is divided into screens which are called activities. At the moment the application of this thesis contains 7 Activities:

3.3.1 Welcome activity

Welcome screen, opens every time on application start, which lasts a few seconds, allowing user to see which application is being opened.

3.3.2 Home activity

Main screen of the application, where the list of event cards is located with brief information about events and their pictures. Also, there is a side menu, called a drawer, which slides from left edge of the screen when the screen is swiped from left to right.

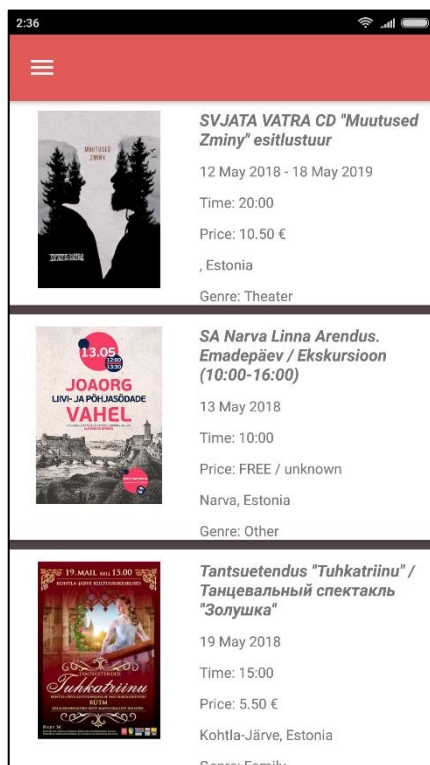


Figure 18. Home activity. (Source: author)

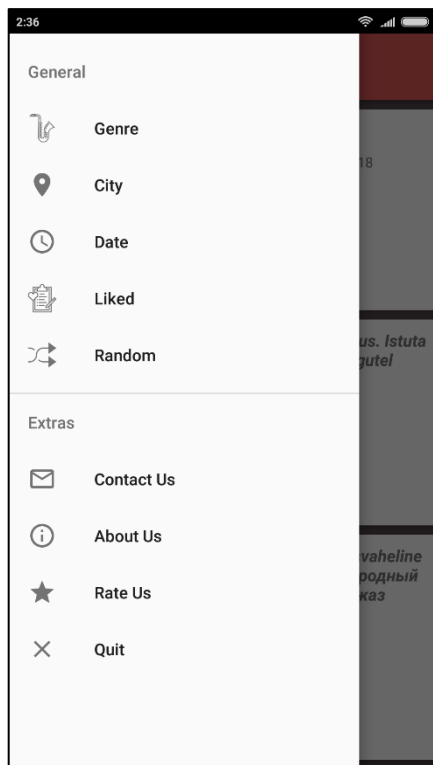


Figure 19. *Drawer menu. (Source: author)*

3.3.3 Event details activity

Detailed information about event, which card was tapped. Also, there are 5 buttons in this view:

- 4 buttons at the top of screen. They belong to the toolbar menu:
 - Back button – closes “event details layout”;
 - Share button – allows users to share this event via social networks (Figure 23);
 - “Report” button – allows users to send a complain about the selected event. Also leads users to a new activity called ReportEventActivity (Figure 24);
 - Add button – adds the event to the device calendar (Figure 22);
- A floating button with the sphere icon on it – opens the event’s web page;

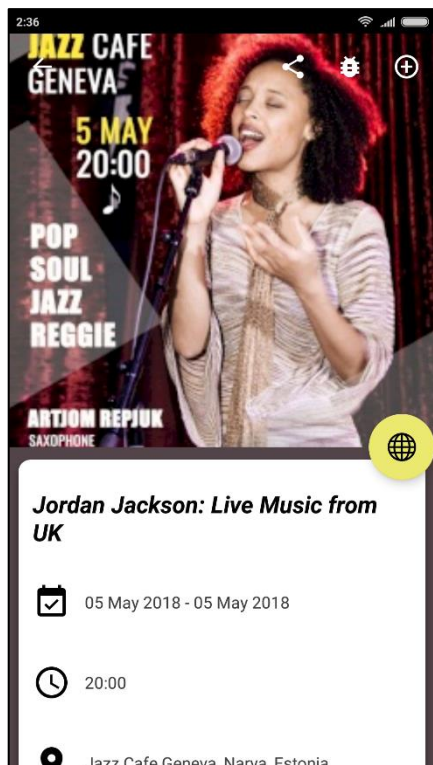


Figure 20. Event details activity, 1 part. (Source: author)

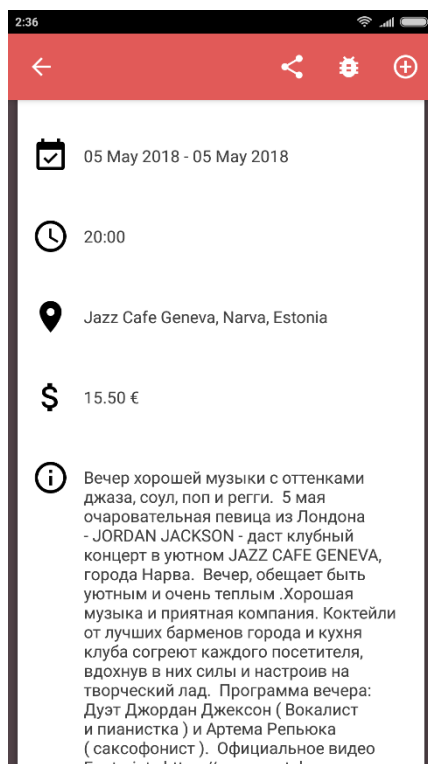


Figure 21. Event Details activity, second part. (Source: author)

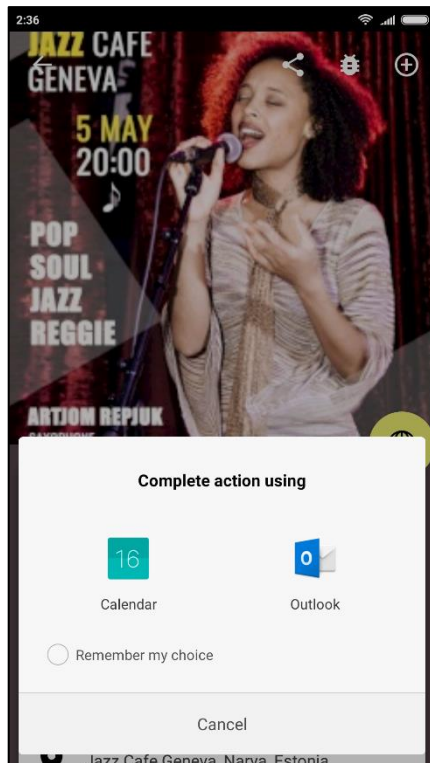


Figure 22. Add to calendar action. (Source: author)

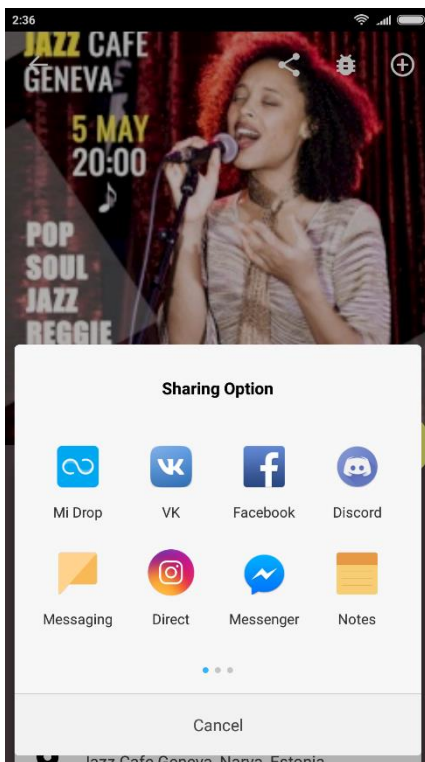


Figure 23. Sharing option. (Source: author)

3.3.4 Report activity

“Report an event” screen. User can report only the chosen event.

Sends a POST request to “https://fto.ee/api/v1/reports/create” URL, which sends the user’s message and the event unique identification number as a JSON array.

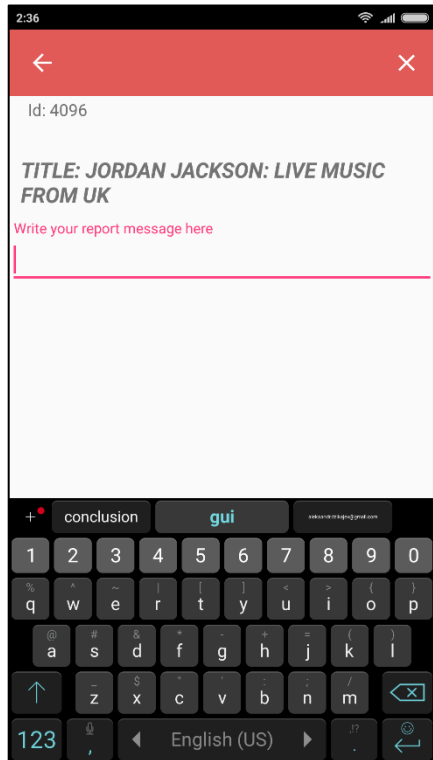


Figure 24. Report event activity. (Source: author)

3.3.5 Genre setting activity

The screen for filtering events by genre. Application gets a list of genres by making a GET request to “https://fto.ee/api/v1/genres” URL, which contains a JSON array of all genres, which are listed in the information for all parsed events. The user can choose one or several genres using corresponding toggles. The screen saves the selected list of genres when closed.

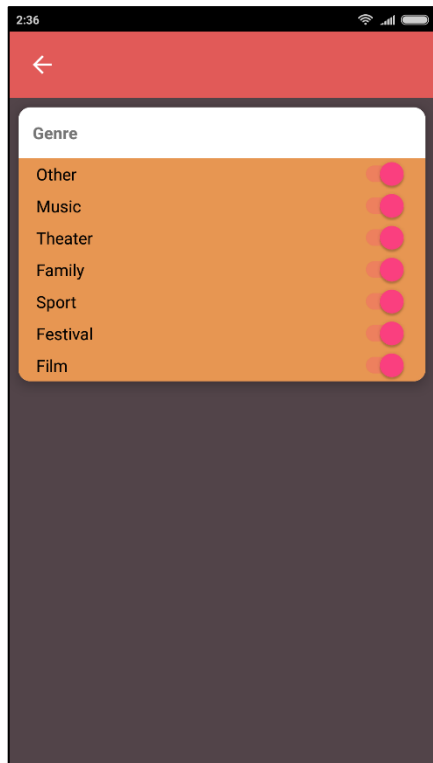


Figure 25. *Genre settings activity. (Source: author)*

3.3.6 City setting activity

The screen for filtering events by cities. Application gets a list of cities by making a GET request to “<https://fto.ee/api/v1/events/city>” URL, which contains a JSON array of all cities, which are listed in the information for all parsed events. The screen saves the selected list of cities when closed.

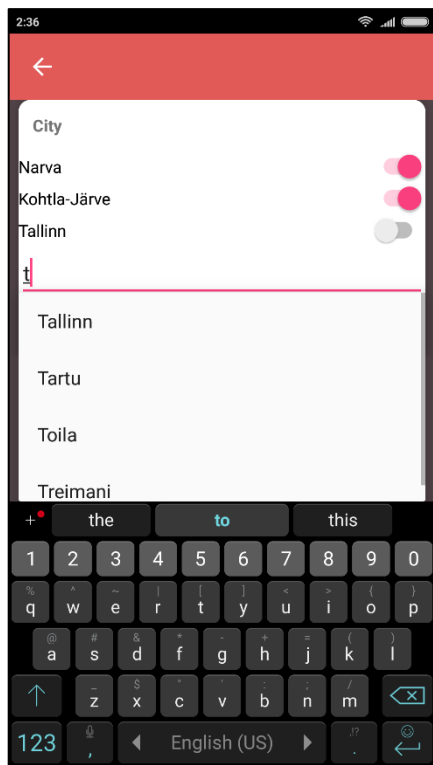


Figure 26. City settings activity. (Source: author)

3.3.7 Date setting activity

The screen for filtering events by dates. There is a possibility to filter events by:

- Anytime
- Today
- Tomorrow
- This week
- This month
- Custom date from calendar

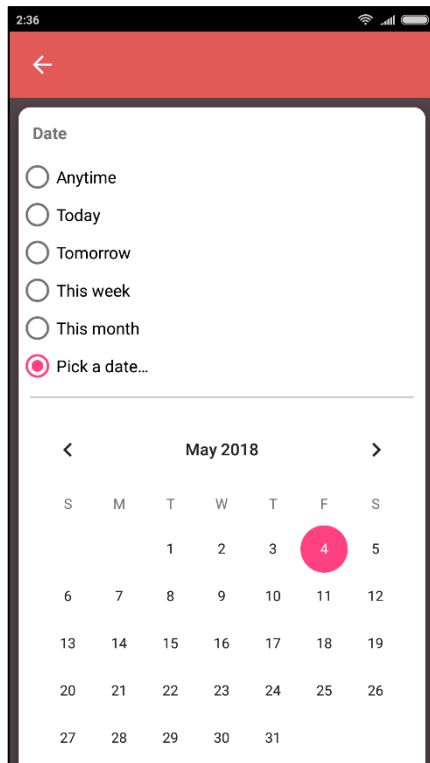


Figure 27. *Date settings activity.* (Source: author)

3.4 Connection to back-end

In this subchapter all URL links used in the application are listed. Each URL provides a JSON array with a big amount of data. The application parses them and converts the date to readable format for users.

3.4.1 Events

GET <https://fto.ee/api/v1/events> – can be used to get events;

GET <https://fto.ee/api/v1/events/find/{date}/{city}/{genres}> – used to find events by genre, city and date;

GET <https://fto.ee/api/v1/events/city> – used to load and provide actual list of cities;

GET <https://fto.ee/api/v1/events/random> – used to provide one random event for today;

3.4.2 Reports

POST <https://fto.ee/api/v1/reports/create> – used in sending reports;

3.4.3 Genres

GET <https://fto.ee/api/v1/genres> – used to load and provide actual list of genres;

3.4.4 URL links testing

Before using URL links in the application all links were tested manually using SoapUI to ensure, that they return right JSON. After successful tests and some changes URL links were implemented in the application.

3.5 Event filters

Event filtering is occurring by 3 criteria: genre, city and date by writing the final URL for the GET request, which looks like this:

“https://fto.ee/api/v1/events/find/{date}/{city}/{genres}”.

Each settings activity creates its own text variable, into which the key words of the query are written. After the activity closes, the string variables add up, forming the final request URL.

Listing 1. Example of used routes.

```
https://fto.ee/api/v1/events/find/all/Narva_Tallinn/1_4_6
https://fto.ee/api/v1/events/find/tomorrow/all/1
https://fto.ee/api/v1/events/find/all/Narva_Tallinn
https://fto.ee/api/v1/events/find/2018-05-14/Narva_Tallinn/1_4_6_0
```

3.6 JSON parsing

To obtain information and load it into the application author uses an Android library, called Volley⁷. Also, there are 2 methods, which are called one after another as soon as the final request URL was created. The first method (Figure 28) creates a request to URL and after the successful connection it calls the second method (Figure 29), which alternately record each element of the JSON array in the ArrayList of the DataObject object.

⁷ Volley is an HTTP library that makes networking for Android apps easier and most importantly, faster.

```

304     public void JSON_HTTP_CALL(String url) {
305         RequestOfJJsonArray = new JsonRequest(url,
306             new Response.Listener<JSONArray>() {
307                 @Override
308                 public void onResponse(JSONArray response) {
309                     ListOfdataAdapter.clear();
310                     EventDataHold.clear();
311                     ParseJJsonResponse(response);
312                 }
313             },
314             new Response.ErrorListener() {
315                 @Override
316                 public void onErrorResponse(VolleyError error) {
317                 }
318             }
319         ));
320         requestQueue = Volley.newRequestQueue(context: HomeActivity.this);
321         requestQueue.add(RequestOfJJsonArray);
322     }
323 }

```

Figure 28. *JSON_HTTP_CALL method. (Source: author)*

```

325     public void ParseJJsonResponse(JSONArray array) {
326         for (int i = 0; i < array.length(); i++) {
327             DataAdapter GetDataAdapter2 = new DataAdapter();
328             JSONObject json = null;
329             try {
330                 json = array.getJSONObject(i);
331                 GetDataAdapter2.setEventID(json.getString(Event_Id_JSON));
332                 GetDataAdapter2.setEventName(json.getString(Event_Name_JSON));
333                 GetDataAdapter2.setEventDate(json.getString(Event_Date_JSON));
334                 GetDataAdapter2.setEventEndDate(json.getString(Event_End_Date_JSON));
335                 GetDataAdapter2.setEventTime(json.getString(Event_Time_JSON));
336                 GetDataAdapter2.setEventLocation(json.getString(Event_Location_JSON));
337                 GetDataAdapter2.setEventCity(json.getString(Event_City_JSON));
338                 GetDataAdapter2.setEventCountry(json.getString(Event_Country_JSON));
339                 GetDataAdapter2.setEventGenre(json.getString(Event_Genre_JSON));
340                 GetDataAdapter2.setEventInfo(json.getString(Event_Info_JSON));
341                 GetDataAdapter2.setEventLink(json.getString(Event_Link_JSON));
342                 GetDataAdapter2.setEventImageURL(json.getString(Image_URL_JSON));
343                 GetDataAdapter2.setEventPrice(json.getString(Event_Price_JSON));
344                 GetDataAdapter2.setEventDescription(json.getString(Event_Description_JSON));
345             } catch (JSONException e) {
346                 e.printStackTrace();
347             }
348             ListOfdataAdapter.add(GetDataAdapter2);
349         }
350         recyclerViewadapter = new RecyclerViewAdapter(ListOfdataAdapter, context: this);
351         recyclerView.setAdapter(recyclerViewadapter);
352     }

```

Figure 29. *ParseJJsonResponse method. (Source: author)*

3.7 Functional requirements

Functional requirements describe the functions that the software must perform. Functional requirements can be considered as tasks for the developer.

This subchapter lists functional requirements, which were created by analyzing strong and weak sides of the existing competing applications and adding to it the author's own vision.

- Users must have an ability to see a list of events;

- Users must have the ability to filter events by genre;
- Users must have the ability to filter events by city;
- Users must have the ability to filter events by date;
- Users must be able to click on the card with event;
- Users must be able to get detailed information about the event;
- Users must be able to open a source web page of a chosen event;
- Users must be able to the opportunity to proceed to buying a ticket for a chosen event, if such an option exists;
- Users must be able to add an event to the mobile device's calendar;
- Users must be able to share the event via messenger, email, SMS, etc.;
- Users must be able to complain about the event.

3.8 Non-functional requirements

Non-functional requirements are associated with such integration properties of the system as reliability, response time or the size of the system. They do not affect the functionality.

This subchapter represents list of non-functional requirements.

- The application must be placed on Google Plat market;
- The application must not require registration;
- The application must not require powerful mobile device;
- The application must run on Android version 5.0 and higher;
- The application must remember settings, when closed;
- The application must gather usage statistics;
- The application must be scalable for future development;
- The application must have simple design;
- The application must show welcome message every time it starts.

3.9 Publishing in Google Play market

To upload the application to Google Play there are some important steps which must be performed:

- Registration as a developer.

Registration as a developer is on a paid basis. For European countries it costs 25 euro.

- Signing the APK file.

APK file must be signed with the help of Android Studio. The IDE allows the developer to create his own passwords, which are called keys. Developers must use the same keys every time they sign the application installation file.

- Uploading and signature verification.

After the file is uploaded, Google Play services compare the signature of the new APK with the old APK, if one exists. After that step Google Play tools start testing the uploaded application.

After all listed above steps were completed, the application appeared in Google Play market⁸.

3.10 Application testing

3.10.1 Manual testing

For manual application testing it was decided to spread the application among volunteers and the author's acquaintances. Testers were told to do the following actions in the application:

- To open the application;
- Check if the user can see a list of cards with events;
- To enter the filter settings by genre and select any genre the user wants;
- To enter the filter settings of cities and select any city the user wants;
- To enter the filter settings by date and choose any option the user wants;
- To click on the card with genres and get detailed information about the event;
- To add an event to the mobile device's calendar;
- To share the event via messenger, email, SMS, etc.;
- To send a report about the event.

When all the actions listed above were made, users gave their feedback and suggestions about the application.

⁸ <https://play.google.com/apps/testing/fto.ee.swk.freetimeorganizer>

After design changes and bug fixes were made, the author repeated the manual test with same people one more time to be sure that all problems were solved.

3.10.2 Testing with Google Play tools

The automatic testing was made with the help of Google Play Pre-launch report. After the application is uploaded into the Google Play, the first step is to check if the application starts well and has no crashes. After all bugs were fixed the Google Play Pre-launch report does not show any crashes (Figure 30).

The performance of the application was also automatically tested, it shows, that application is decently fast and its performance depends on how old the device is. Information is on picture below (Figure 31).

The “Security” tab tells that no known vulnerabilities were detected for the application (Figure 32).

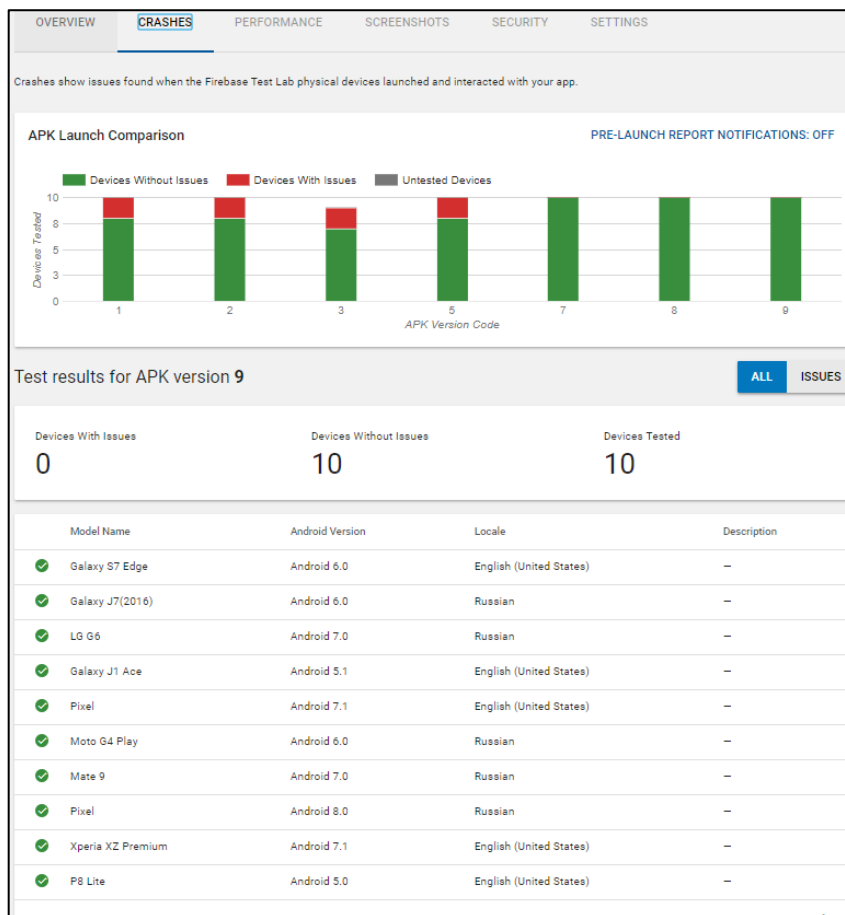


Figure 30. APK Launch Comparison, Crashes. (Source: author)

| OVERVIEW CRASHES PERFORMANCE SCREENSHOTS SECURITY SETTINGS | | | | | | | |
|---|---------------------|----------------------------|---|---|-----------------------------|---------------------------|---|
| | Device model | Avg. CPU (Percent) ? | Avg. network sent (Bytes/Sec) ? | Avg. network received (Bytes/Sec) ? | Avg. memory (Bytes) ? | Startup time (ms) ? | |
| ✓ | Galaxy S7 Edge ⓘ | 1.24% | 1.6K | 65K | 57M | 522 | ▼ |
| ✓ | Galaxy J7(2016) ⓘ | 1.16% | 1.2K | 57K | 32M | 373 | ▼ |
| ✓ | LG G6 ⓘ | 2.19% | 1.7K | 91K | 61M | 408 | ▼ |
| ✓ | Galaxy J1 Ace ⓘ | 3.00% | 917 | 49K | 16M | - | ▼ |
| ✓ | Pixel ⓘ | 6.34% | 3.7K | 165K | 113M | 230 | ▼ |
| ✓ | Moto G4 Play ⓘ | 9.15% | 2.9K | 141K | 55M | 615 | ▼ |
| ✓ | Mate 9 ⓘ | 4.02% | 3.2K | 112K | 100M | 179 | ▼ |
| ✓ | Pixel ⓘ | 4.48% | 3.2K | 126K | 56M | 235 | ▼ |
| ✓ | Xperia XZ Premium ⓘ | 3.64% | 2K | 165K | 118M | 226 | ▼ |
| ✓ | P8 Lite ⓘ | 4.30% | 2.2K | 115K | 46M | 620 | ▼ |

Figure 31. APK Launch Comparison, Performance. (Source: author)

| | | | | | |
|----------|---------|-------------|-------------|-----------------|----------|
| OVERVIEW | CRASHES | PERFORMANCE | SCREENSHOTS | SECURITY | SETTINGS |
|----------|---------|-------------|-------------|-----------------|----------|

The security section identifies issues that were found after checking your app's APK file for known vulnerabilities.



Security Scan Complete
 No known vulnerabilities were detected for APK 9.

Figure 32. APK Launch Comparison, Security. (Source: author)

CONCLUSION

The problem which this thesis seeks to solve is that there was no single environment in which it would be possible to find structured and sorted information about all local Estonian cultural, sport, educational, entertainment and other such events.

The aim of the thesis was the development of the Android application “Free Time Organizer”, which would help users to find the information about local events of interest for chosen date and type for any of the cities available in the application’s database.

To accomplish the aim of the work, the author performed following tasks:

- Developed the design of the application;
- Created a data exchange model between the back-end and front-end parts of the application;
- Implemented all functional requirements in the application;
- Placed the application to the Google Play store;
- Conduct manual testing of the application;
- Conduct automated testing with Google Play tools.

As these tasks were successfully completed and the goal of the thesis was achieved. The application was written in Java programming language, using Gradle as the project builder, XML as the markup language, Android Studio as the development environment and JSON as the data transfer format. Author used extreme programming methodology with very short iterations and with a working product with an added feature at the end of each iteration.

The author has implemented in his application almost all positive features from the already available competing applications. In addition, the author’s application is better targeted for its core market, Estonia, than the competing applications.

The capabilities of the created application fully cover the previously formulated functional requirements:

- Users have an ability to see a list of all events;
- Users can filter events by genre;
- Users can filter events by city;
- Users can filter events by date;

- Users can tap on a card with event;
- Users can get detailed information about the event;
- Users can open a source web page of a chosen event;
- Users have the opportunity to proceed to buying a ticket for a chosen event, if such an option exists;
- Users can add an event to the calendar of the mobile device;
- Users can share the event information via messenger, email, SMS, etc.;
- Users can complain about the event, if they find the information inappropriate.

The application receives data from the back-end part, which was developed by another author. The data structure and data exchange logic were determined in co-operation with another author.

The manual testing was held using volunteers, who agreed to give feedback about the application after use. Performance testing was held using Google Play tools.

The application is now available on Google Play market and every owner of an Android-based device can download and use the application. The link to the application download was given in part 3.9.

In the future, the author plans to add more functionality such as possibility to give feedbacks on recent events. Also, the author plans to find ways to monetize the application and to create iOS and PWA versions of the application to expand number of users. In addition, the author wants to expand the application service to new counties.

RESÜMEE

Probleem, mida lahendab see lõputöö, on tõsiasi et Eestis puudub ühine keskkond, milles igaüks saaks leida kogu informatsiooni kõikidest kultuuri-, spordi-, hariduse, meelelahutuse jms sündmuste kohta struktureeritud kujul, mis võimaldaks selles infokogus kiiresti leida enda jaoks huvitavaid kohalikke sündmusi.

Selle lõputöö eesmärk oli rakenduse “Free Time Organizer” arendamine Android platvormile, mille abil kasutajad saaksid leida informatsiooni neid huvitavatest kohalistest sündmustest valitud kuupäeva ja sündmuse tüübi kohta iga asukoha jaoks, mille kohta on andmeid rakenduse andmebaasis.

Lõputöö eesmärgi saavutamiseks töö autor lahendas järgmisi ülesandeid:

- Luua rakenduse disaini;
- Luua andmevahetuse mudeli rakenduse back-end-i ja front-end-i vahel;
- Juurutada rakendusse kõik rakendusele esitatud funktsionaalsed nõudmised;
- Laadida rakenduse üles Google Play keskkonda;
- Läbi viia manuaalse rakenduse testimise;
- Läbi viia automaatse rakenduse testimise Google Play tööriistade abil.

Kõik need ülesanded olid edukalt täidetud ja töö eesmärk oli sellega saavutatud. Rakendus oli kirjutatud Java programmeerimiskeeles, kasutades Gradle-i kui projekti ehitamise tööriista, XML-i kui kujunduse märgistuskeelt, Android Studio-t rakenduse arenduskeskkonnana ja JSON-it andmevahetusformaadina. Autor kasutas rakenduse arendamise käigus ekstreemprogrammeerimise meetodit, mis seisnes väga lühikestes arendustsüklites, kus iga arendustsükli lõpus oli töötav rakendus uue lisatud omadusega.

Autor üritas lisada oma rakendusse kõik kasulikud omadused ja funktsioonid, mida ta nägi teistes olemasolevates sarnastes rakendustes. Lisaks sellele, autor julgeb väita, et tema rakendus on oluliselt paremini sobitatud just Eesti turule, kui teised sarnased rakendused.

Loodud rakenduse võimalused täielikult katavad töös sõnastatud rakenduse funktsionaalseid nõudeid:

- Kasutajatel on võimalus näha nimekirja kõikide sündmustega;
- Kasutajad saavad filtreerida sündmusi tüübi järgi;

- Kasutajad saavad filtreerida sündmusi linna järgi;
- Kasutajad saavad filtreerida sündmusi kuupäeva järgi;
- Kasutajad saavad valida üksiku sündmuse;
- Kasutajad saavad näha üksikasjaliku informatsiooni valitud sündmuse kohta;
- Kasutajad saavad avada sündmuse veebilehe, kui see on olemas;
- Kasutajad saavad minna sündmuse pileti ostmise veebilehele, kui see on olemas;
- Kasutajad saavad lisada sündmuse mobiilse seadme kalendriprogrammi;
- Kasutajad saavad jagada infot sündmuse kohta sotsiaalvõrgustike, sõnumite jms kaudu;
- Kasutajad saavad saata kaebuse sündmuse kohta, kui sündmuse info tekitab kahtlusi.

Rakendus saab andmeid rakenduse back-end osast, mille on kirjutanud teine autor. Andmete struktuuri ja andmevahetuse loogika on mõlemad autorid koos välja töötanud.

Rakenduse manuaalne testimine oli läbi viidud kasutades vabatahtlikke, kes nõustusi anda tagasisidet rakenduse kasutamise kohta peale selle kasutamist. Rakenduse jõudluse testimine oli läbi viidud kasutades Google Play keskkonna tööriistu.

Rakendus on nüüd kättesaadav alla laadimiseks Google Play keskkonnas ja iga Android seadme kasutaja saab seda nüüd endale paigaldada. Viit rakenduse alla laadimiseks on toodud osas 3.9.

Tulevikus plaanib autor lisada rakendusse veel rohkem funktsionaalsust, näiteks, anda kasutajatele võimaluse kirjutada kommentaare hiljuti toimunud sündmuste kohta. Samuti, plaanib autor otsida võimalusi rakenduse abil raha teenida ning luua lisaks rakenduse iOS ja PWA versioone, selleks et suurendada rakenduse kasutajaskonda. Lõpuks tahab autor mingil hetkel alustada ka rakenduse laienemist ka teiste riikide turgudele.

REFERENCES

- Wikipedia 2018a. *Android (Operating System)*. Available at [https://en.wikipedia.org/wiki/Android_\(operating_system\)](https://en.wikipedia.org/wiki/Android_(operating_system)), accessed March 4, 2018.
- Wikipedia 2018b. *Gradle*. Available at <https://en.wikipedia.org/wiki/Gradle>, accessed March 4, 2018.
- Wikipedia 2018c. *Version control*. Available at https://en.wikipedia.org/wiki/Version_control, accessed March 4, 2018.
- Cambridge University Press 2018a. *API*. Available at <https://dictionary.cambridge.org/dictionary/english/api>, 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. *Front end*. Available at <https://dictionary.cambridge.org/dictionary/english/front-end>, accessed May 8, 2018.
- JSON 2018. *Introducing JSON*. Available at <https://json.org/>, accessed March 4, 2018.
- Google. *Meet Android Studio*. Available at <https://developer.android.com/studio/intro/index.html>, accessed March 22, 2018.
- W3C 2008. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Available at <https://www.w3.org/TR/xml/>, accessed March 4, 2018.
- Techopedia 2018. *Android App*. Available at <https://www.techopedia.com/definition/25099/android-app>, accessed March 4, 2018.
- Babich Nick 2017. *The Underestimated Power Of Color In Mobile App Design*. Available at <https://www.smashingmagazine.com/2017/01/underestimated-power-color-mobile-app-design/>, accessed May 22, 2018.
- Jeffries E. Ronald 2011. *What is Extreme Programming?* Available at <https://ronjeffries.com/xprog/what-is-extreme-programming/>, accessed May 8, 2018.
- Smartbear 2018. *SoapUI*. Available at <https://www.soapui.org/docs/functional-testing/getting-started.html>, accessed May 20, 2018.

APPENDICES

Appendix 1. Source code

- Android application – <https://github.com/sawkabasher/FreeTimeOrganizer>