Bachelor Project

Music Voting Server

Mykhailo Svyrydovych

Elvin Mammadov

Parisa Ostadzadeh

Inhaltsverzeichnis

[1. Introduction 2](#_Toc64570032)

[2. Motivation and objectives 3](#_Toc64570033)

[3. Preparation 3](#_Toc64570034)

[3.1 Gantt Chart 3](#_Toc64570035)

[3.2 Requirements 3](#_Toc64570036)

[3.3 Risks 4](#_Toc64570037)

[3.4 Technologies 4](#_Toc64570038)

[3.4.1 Frameworks and libraries 4](#_Toc64570039)

[3.4.2 Task tracking 5](#_Toc64570040)

[3.4.3 Version Control 6](#_Toc64570041)

[4. Use case diagram 7](#_Toc64570042)

[5. Client-server architecture and REST API 7](#_Toc64570043)

# Introduction

The topic of this project is development of a Web Application that can be ran on a Raspberry PI[[1]](#endnote-1) microcomputer. The application can be installed on an offline server (without Internet connection) and run entirely inside a local Wi-Fi network. The use case is to provide means for song voting that forms a live playlist. The application also plays music and allows songs uploads. It is useful during parties where participants would like to vote for their favorite songs.

Users‘ Devices

Speakers

Admin laptop

Web Application (backend)

Linux (Raspbian)

Raspberry PI

Laptop

Android

iOS

General project scheme

# Motivation and objectives

The main objective was to practice working on a software project (that has more complex architecture than university labs) and coordinating as a team of several developers. We had to plan the project schedule using Gantt Chart. It is quite popular tool in Project Management when there is a need to show an activity over time. In our case it helped to make sure that our tasks and activities are inside the project timeframe. We used it to plan our work. The second tool that we used was YouTrack[[2]](#endnote-2) - browser-based software. It is JIRA-like project management tool that is used for task planning and bug tracking.

The second goal was to get knew knowledge and deepen existing in programming languages, frameworks and tools for Web- and mobile development. As these two areas of software development are quite popular nowadays.

Finally, our goal was to have a working prototype that can be open for building additional functionality and third-party integrations on top. The main features were planned with look into keeping up to a general time frame for a project. But we would like to continue working on it in future as well. The main features included: upload songs, played pre-installed and uploaded songs, show song list to users, accept votes, allow remote administration.

# Preparation

## Gantt Chart

Gantt chart is a bar chart that is used to illustrate a schedule of a project. This kind of a chart includes tasks to be performed (on the vertical axis) and time intervals for the tasks (on the horizontal axis).[[3]](#endnote-3) For our project we used an Excel Gantt chart template as it is simple and easy to use. We needed the chart to plan the tasks for team members for the whole project according to given time frame. We did not include there every possible single task, but rather main general tasks to complete the project. We understood at that point that some additional tasks will appear as we progress and planned the time accordingly. We split tasks into five phases. Task were assigned to a Task Owner, but it did not mean that the person should do the whole task alone, but to control and be responsible for task completion. Our Gantt chart is included at the end of this report.

## Requirements

As a part of the planning process, we had a team meeting where we brainstormed the possible projects features and derived requirements.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

It was a draft list of what the project prototype should be able to do by the end of the planned project schedule.

## Risks

Risk management is a crucial part in any long running project. As we planned to do the project for at least 4 months, we decided to discuss all possible issues that could happen and influence the project. We created an xml table with risks descriptions, likelihood, impact, severity, owner, mitigation actions and contingent actions. The full table can be found in the annex.

## Technologies

It is really important to choose the right technologies at the beginning and not to switch it in the middle of the project. The main points in choosing the technologies for the project were:

1. Fitting to the project requirements (Web-App that can be run on a Raspberry PI microcomputer, mobile apps for the client)

2. Popularity on the job market (Our interest is to learn and work with modern tools that are required by potential employers)

3. Good documentation, tutorials etc. (To be able to learn it fast)

### 3.4.1 Frameworks and libraries

Any web application can be generally divided into two parts: backend (running on the server) and frontend(running on the user device). Nowadays all web-applications are developed using frameworks and libraries. Frameworks usage increases the development speed and makes it possible to invest more time into implementing concrete features rather than inventing a bicycle and do some low-layer programming. For the backend we decided to choose some framework based on one of the following languages: Java, C++, JavaScript (as team members had some knowledge in these three). The respective frameworks to look at were: Spring Boot, Treefrog, Express.js. Discussing them in a team according to the 3 technology requirements mentioned above, we decided to develop the backend with Java *Spring Boot*. Research showed that there were some other successful web-application projects that were run on the Raspberry PI without problems, Java is a very popular language through the Hamburg companies and, finally, there are a really nice official documentation and non-official tutorials. The dominating language for the browser frontend is JavaScript. And we decided to use React framework as it has very useful official documentation with tutorial and it is the most popular frontend framework today.

For the mobile clients the choice was to write native apps for Android and iOS in Java and Swift respectively or to choose some multiplatform solution. As we did not have enough manpower, we decided to search for the latter one. The choice here was between Flutter and React native. We decided to use Flutter.  
//Elvin – write about Flutter technology and why did we choose it for the project

### 3.4.2 Task tracking

JetBrains YouTrack is a web-based issue tracking and project management platform.[[4]](#endnote-4) It designed for agile teams. It allows to create, assign and keep track of the development tasks. Task template can be easily customized with various options. Next important feature is the Kanban board that represents tasks and their states. The aim of the Kanban board is to make the general project workflow and task progress clear and available to all participants.[[5]](#endnote-5)

Finally, YouTrack allows integration with GitHub. This connection helps to track git commits right inside YouTrack(e.g. task key can be mentioned in a commit and it triggers an action at YouTrack).  
  
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

YouTrack Dashboard

Ein Bild, das Text, Screenshot, Monitor, Bildschirm enthält.

Automatisch generierte Beschreibung

YouTrack Kanban Board

### 3.4.3 Version Control

Version control or source control is a practice that helps to track and manage changes of a source code. Especially it is useful in software teams, when several developers are working on the same project. It helps them to work faster and smarter. Version control systems keeps track of any change in some kind of a database and allows to revert a change and jump back to a specific point in the history. It protects the codebase from human errors and makes bug fixing easier. [[6]](#endnote-6)

#### Benefits of version control systems

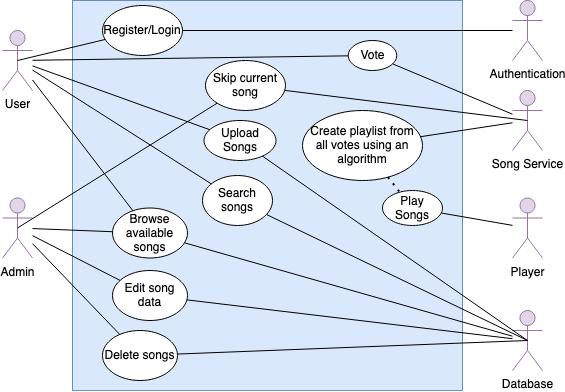
* Getting long-term history of every file
* Branching and merging. Allows to keep multiple streams of work separated for a while with a possibility to merge.
* Being able to trace each change mode to the program and track it with task tracking tools[[7]](#endnote-7)

#### Different VCS types

Version Control Systems can be divided into two groups: distributed and centralized. It describes how the remote architecture is built. A centralized VCS has a single point of failure – central storage. When it is lost, the whole project is lost. A distributed VCS provides a full copy of the source code for each instance. Examples of centralized systems: SVN, CVS. Examples of distributed systems: Git, Mercurial.[[8]](#endnote-8) We decided to use Git because it is the most popular distributed VCS.

The two most popular Git hosting platforms are GitHub and GitLab. We chose GitHub as it allows to make a repository public in future (GitLab supports only private repositories).

# Use case diagram



# Client-server architecture and REST API

A server is a remote computer that accepts requests from clients, processes them and sends back the response data over HTTP/HTTPS protocol.

A client is a computer or a device that sends requests to the server and receives a response.[[9]](#endnote-9)

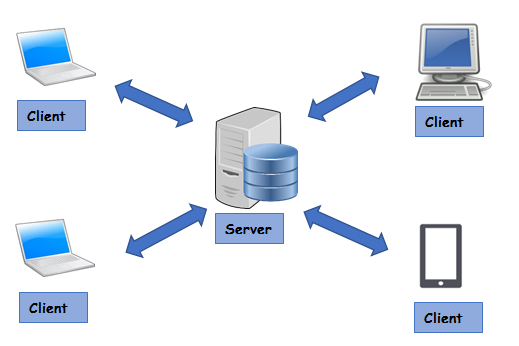


Figure. Client-server architecture[[10]](#endnote-10)

REST is a web architectural structure first described in a scientific paper called ﻿“Representational State Transfer” (REST). In web architecture clients use Application Programming Interfaces (APIs) to communicate with web server. These APIs are been used to provide a set of functions to allow communication between computer programs and make data exchange. Nowadays REST API architecture pattern are widely used.[[11]](#endnote-11)

Generally, REST API architecture should conform following principles[[12]](#endnote-12):

* Uniform interface
* Client–server
* Stateless
* Cacheable
* Layered system
* Code on demand (optional)

## Project REST API overview

|  |  |  |
| --- | --- | --- |
| Http method | Endpoint | Description |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. https://www.raspberrypi.org [↑](#endnote-ref-1)
2. https://www.jetbrains.com/youtrack/ [↑](#endnote-ref-2)
3. https://en.wikipedia.org/wiki/Gantt\_chart [↑](#endnote-ref-3)
4. https://www.jetbrains.com/help/youtrack/standalone/YouTrack-Documentation.html [↑](#endnote-ref-4)
5. https://en.wikipedia.org/wiki/Kanban\_(development)#Kanban\_boards [↑](#endnote-ref-5)
6. https://www.atlassian.com/git/tutorials/what-is-version-control [↑](#endnote-ref-6)
7. https://www.atlassian.com/git/tutorials/what-is-version-control [↑](#endnote-ref-7)
8. https://bitbucket.org/product/version-control-software [↑](#endnote-ref-8)
9. https://www.educative.io/courses/learn-rest-soap-api-test-automation-java/JEK9Mv7RLA9 [↑](#endnote-ref-9)
10. https://www.omnisci.com/technical-glossary/client-server [↑](#endnote-ref-10)
11. Marc Masse - REST API design rulebook [↑](#endnote-ref-11)
12. https://restfulapi.net/rest-architectural-constraints/ [↑](#endnote-ref-12)