Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Multiprojection control system

Bc. Ondřej Trojan

Supervisor: Ing. David Sedláček, Ph.D.
Field of study: Open Informatics
Subfield: Software Engineering
May 2020

Acknowledgements

I would like to express my gratitude to my supervisor Ing. David Sedláček, Ph.D. for the useful comments, remarks and engagement through the learning process of this master thesis. Furthermore, I would like to thank the testing participants, who have willingly shared their precious time during the process of testing. Lastly, I would like to thank my family and girlfriend who have supported me throughout the entire process of research and my whole study.

Declaration

Abstract

This work describes the design and development of a multi-projection control system built on top of the performed analysis and research. The use case of such a system is a research-oriented educational presentation with a targeted audience, specifically the interactive exhibition of Langweil Prague city model. This system is divided into three components: Web, App and Video player. While the web administrator console is for system configuration and maintenance, the mobile application serves as a convenient "remote" control and a daily operation tool. Last but not least, the device media player in the form of a mini PC is running a VLC player in the remote control interface. Before the design and implementation, we have performed a comparative analysis out of which we evaluated most suitable bundle of tools and frameworks. On top of the realised application, usability testing was executed from which factual conclusions and observations were obtained. That said, our demonstration system proved the feasibility of our design and suggested improvements and directions in which the next development should be targeted.

Keywords: Multi-display projection, Web development, Mobile app, VLC player, Java, Spring, Flutter

Supervisor: Ing. David Sedláček, Ph.D.
Department of Computer Graphic and Interaction,
Karlovo náměstí 13,
Praha 2

Abstrakt

Tato práce popisuje návrh a vývoj multiprojekčního řídicího systému postaveného na provedené analýze a výzkumu. Příkladem použití takového systému je výzkumně orientovaná vzdělávací prezentace s cíleným publikem, konkrétně interaktivní výstava modelu města Langweil v Praze. Práce je rozdělena do tří částí: Web, App a Video přehrávač. Zatímco konzole webového administrátora slouží pro konfiguraci a údržbu systému, mobilní aplikace slouží jako praktický nástroj pro dálkové ovládání a každodenní obsluhu. Poslední částí je přehrávač médií ve formě mini PC, na kterém běží VLC přehrávač v rozhraní dálkového ovládání. Před návrhem a implementací byla provedena srovnávací analýza, z níž byl vyhodnocen nejvhodnější soubor nástrojů a frameworků. Kromě realizované aplikace bylo provedeno testování použitelnosti, ze kterého byly získány věcné závěry a pozorování. Tento demonstrační systém prokázal proveditelnost našeho návrhu a ukázal, kterým směrem by měl být další vývoj cílen.

Klíčová slova: Vícedisplejová projekce, Webový vývoj, Mobilní aplikace, VLC přehrávač, Java, Spring, Flutter

Překlad názvu: Řídící systém pro multiprojekci

Contents

Figures

Tables

Chapter 1
Introduction

1.1 Motivation

As we are living in the age of widespread digitally-generated content, people are well accustomed to the visual presentations. Interactive exhibitions are known to be far more appealing to a broad audience and thus serve better as a teaching method. [1]. Therefore, new means of the utilization of such practise shall be introduced. This would unconventionally attract the audience and present the exposition. While dedicated solutions for multi-display projections are expensive, proprietary and limitary customizable, our solution shall be portable, light-weight and modifiable. The system will enable the creation and implementation of educational programs in a very innovative way. According to the available information, this is an original solution that can address both pupils and students with a non-traditional concept of the presentation and consequently increase the general interest of visitors in The City of Prague Museum.

1.2 Aim

The project aims to create a system that could challenge a professional and expensive dedicated solution for multi-display control. The primary focus of the thesis is to develop a system allowing both autonomous and guided presentation of Langweil model of Prague on several client devices with 1-4 connected displays. A secondary intention is to create a web-based console for maintenance of the exhibition's playlists and last but not least development of Mobile application for remote control.

1.3 Thesis structure

The elementary division of the thesis is to Design and Implementation. In the beginning, the introduction of the problematics is presented. It is succeeded by the analysis of the acceptable technologies, followed by the design of the application, its implementation and eventually evaluation with completed tests and observations.

1.4 Digital art

Digital art is a general term used for contemporary arts utilised for a set of artistic works and practices where the use of digital technology appears as a necessary part of the creative process. Since the 70s the various terms have been used for merging multimedia with computer art. The name of digital art is a subset of media art. Nowadays, the impact of digital technology has transformed activities such as painting, drawing, sculpture and music/sound art massively. It led to the development of new forms of art, such as net art, digital installation art together with virtual reality which have become recognised artistic practices. The term "digital artist" is used to describe an artist who makes use of digital technologies in the production of art.

1.5 Langweil model

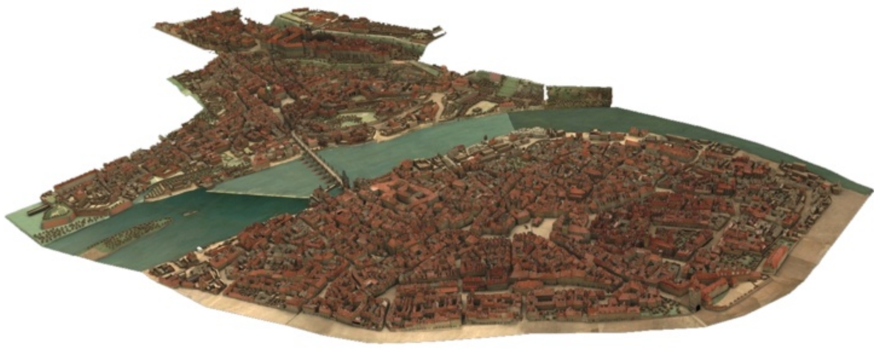


Figure 1.1: Langweil model full view

Lanweil model is a model of Prague made of paper, created by Antonín Langweil between 1826–1837. With its $20m^2$ and more than 2,000 buildings in the centre of Prague, it remained as a unique snapshot of an actual city appearance back in the 19th century. The creator, an employee of the university library, by whom the model is now named, spent all his free time working on this product. Despite being admired, his work consumed a significant amount of money which was mostly paid from his own pocket. Without being patronaged, he died early in poverty. His piece of art, the model in scale 1:480, has never been finished and became a property of the National museum in 1840. After more than 100 years later its holding was transferred to The City of Prague Museum, where it has become the most sought after part of historical exposition. Only today makes it possible to assess the real significance of Langweil's model, the significance of a first-class documentary and an exceptional work of art. [2]

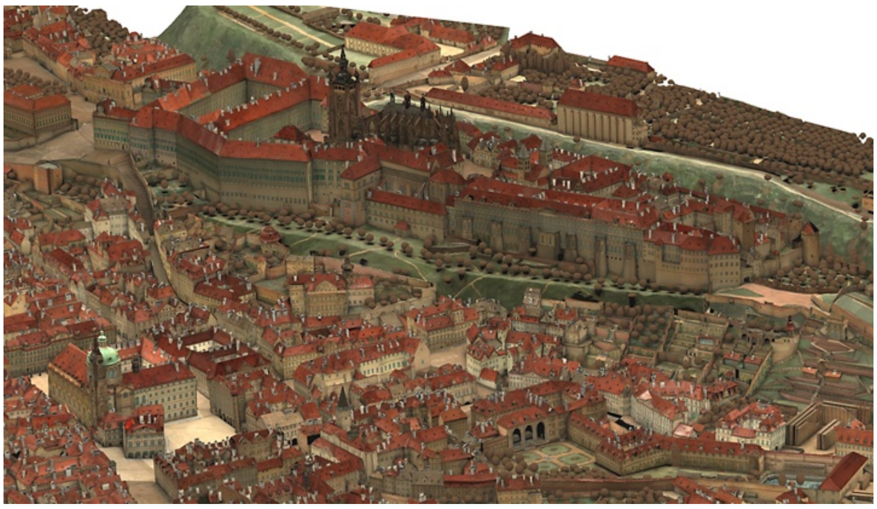


Figure 1.2: Langweil model Hradčany view

Chapter 2
Analysis

In this chapter, we will firstly present existing solutions related to our aim, then available technologies out of which we will evaluate the most suitable ones. Before the requirements stating, system design and the further development we have undertaken an investigation which is elaborated in the sections below.

2.1 Feasibility Study

Please see project constraints at 3.5. From the time perspective, the project deadline is September 2022. The major part of the software and tests shall be delivered in June 2020 whereas the artistic part of the exhibition is to be commissioned in mid-2020. The financial part of the system creation is out of the scope of this study. Given the time between the project release and the present state, the project is by all accounts feasible.

2.2 Multi-projection system

Multi-projection system. Is comprising a plurality of projectors for projecting images onto a plurality of projection surfaces, wherein at least some of the projection surfaces are not parallel to each other. The terminology is based on the patent [3] where the authorship of this concept is claimed. In other words, a system with multiple projectors and multiple projection surfaces can be called a multi-projection system.

Projection mapping. Also known as video mapping represents a system for projection in free space on designated objects, such as house facades or interiors. Is often combined with the audio track and aims at creating a more immersive experience. Both artists and advertise-creators use it for visual experience enhancement or cultural events. Videomapping requires low light conditions and powerful projectors.

Thanks to the terminology's similarity, we can state that both terms projection-mapping and multi-projection systems are often interchanged despite their differences.

2.3 Existing solutions

Firstly, we have performed the research on existing multi-screen video systems where we have eventually found some existing projects, ideas and inspiration.

The overall idea of content streaming on many screens is not new, after all the TV with multiple channels is a great example and has been presented for decades. However, having more delicate control over the program and being truly capable of deciding the content in a contemporary manner came with the computer age. With the habit of "video on demand " media distribution [1], the audience can choose what they want to watch. It is a common practice to stream a video stream in real-time and has several client players just displaying the received content, for example, the VLC streaming function in case of running client program. Such practice is certainly well suitable for many uses. [4]

2.3.1 Projection solutions

However, based on the multi-projection definition we shall investigate existing solutions for this purpose

- Unreal's Engine nDisplay - nDisplay renders the Unreal Engine scene on multiple synchronized display devices. An increasing number of visualization systems aim to immerse the viewer more effectively in the environment by rendering real-time content through multiple simultaneous displays. [5]
- vvvv - is a hybrid visual/textual development environment for various multimedia tasks. It comes with a set of players and has functionality for client node synchronization. However, the whole toolkit is tightly built around DirectX and is therefore only targeted on Windows platform. [6]
- TouchScreen - is a visual development platform. Is artistic oriented and allows to create large multi-machine networked infrastructures using sync operators and hardware framelock. [7]
- GreenHippo - set of solutions for projection mapping and synchronized playback on multiple HW. Is developing high-performance video processing software [8].
- Disguise - is a live event visualisation solution's provider, concentrating on the software, hardware and support services that allow creative production to simulate and deliver their 3D shows in real-time. [9]
- 7th sense - is a provider for HW and SW solutions for multi-media projections such as projection-mapping and multi-projector layout designer. [10]

[1] Youtube, Vimeo, Netflix

The kiosks and video-walls do not match the strict definition of a multi-projection system, but in a relaxed terminology are relevant to the planned exhibition for Langweil model as the technology available for these use-cases are very likely appliable into our project. Therefore by studying their implementations, we shall be able to design and choose technologies for our project in a better way.

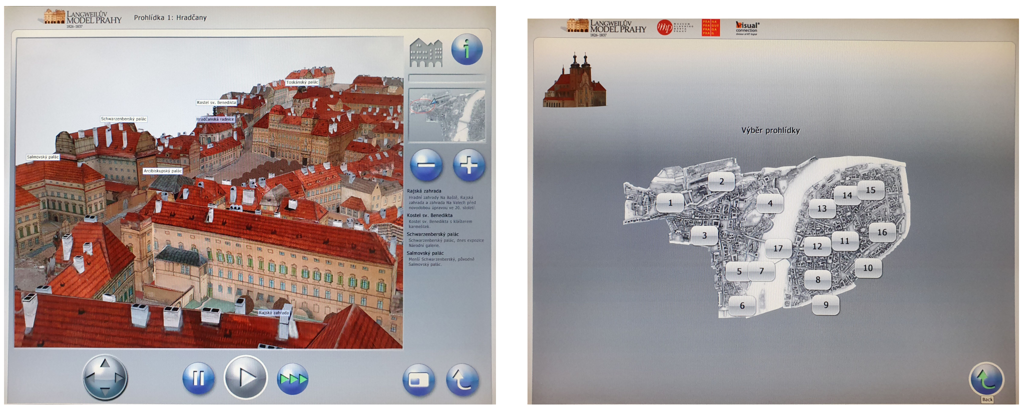
- The video wall is a term describing a set of variable display panels (monitors, video projectors or televisions) mounted together forming a wall or a different shape object. It is apparent that for a satisfactory replacement of large screen with multiple smaller monitor, we should not be distracted by a monitor's side bezels. As a response, the industry has brought a whole category of professional minimum bezel monitors. They have been introduced together with various tools for "smoothing" the gap by the attachment of lenses. The significant advantages of the multi-monitor setup over a single large screen, are the more adaptable ability of layout customization, greater screen area and higher pixel density last two regarding the per-unit cost. Often a single video content is divided by an HW device or SW programme into many video streams fitting the current layout of the video wall. [11]

- Kiosk system is an interactive or non-interactive device featuring specialized hardware and software that provides access to information and applications for communication, commerce, entertainment, or education. [12] Such a device typically has a screen (TV, monitor or projector) and a device outputting some content to the screen. Nevertheless, it can be shipped all-in-one box. The content may run in a loop, change on input or not require any direct manipulation or control. It could serve as an autonomous screen, presenting an advertising device. Alternatively, it could assist as a feedback device where customers can fill a questionary. Technically, such systems are either bought directly in case of Android tablets or are developed independently (Raspberry Pi and any TV) [13]

2.3.2 Previous interactive exhibitions

To see the current state of the interactive exhibition of Langweil model, we went to the main building of The City of Prague Museum located in Florenc and experienced the visit as a typical visitor. On this occasion, we examined the existing interactive exhibition which description follows.

Video-camera and an interactive Kiosk exhibition build over a protective box of the real Langweil model. It was created on the occasion of digitization of the Langweil model with the motivation of presenting the Langweil model in a new interactive form for a broad audience. It was developed by Visual Connection company between 2006-2009 and is from the modern technological perception little outdated. It allows visitors to go through the virtual tour in the digitized historic paper-model of Prague as well as browse with camera focus on specific areas in the real model.



(a) Animated city walk-trough (b) Interactive choices

Figure 2.1: Existing interactive kiosk

2.4 Acceptable technologies

Secondly, our interest was to determine suitable technologies and frameworks for matching our aims stated in section: 1.2. For system design, please see design chapter 3. We can further divide our development into following modules.

- Website - for admin management
- Mobile app - for user control
- Back-end - server-side
- Video player - for video playback
- System integration

This division is better explained in the particular section in Design chapter 3.4. While their analysis is elaborated in the sections below, technology choices are dependent on the project requirements 3.3.

2.4.1 Website

Front-end for exhibition management, control and interact with the server. Shall be robust, secure and responsive. Based on HTML, javascript and CSS, core languages which have become an industry-standard shall be complemented with a framework delivering the ready site to the client. As the final choice depends on the chosen Back-end framework, the details will be delivered at the specific section below. Nevertheless, below-listed options are acceptable candidates for front-end toolkit: [14]

- Server-side rendering - using server-side scripting for dynamic content generation is able to display a fully rendered page on a request.

  - ASP.NET - uses Razor pages and cshtml format for Model-View-Controller development approach [15]
  - Thymeleaf - is Java-based template engine for XML, XHTML and HTML5. Placed on view-layer of the MVC model is known to give great performance for small scale applications. [16]
  - JavaServer Pages - more commonly known as JSP is another Java-based MVC template engine for dynamically generated web pages based on HTML, XML, SOAP. [17]

- Client-side rendering - typically single-site-page, page routing is managed dynamically without the need to refresh the page with each query. Has advantages in UX a responsiveness
  - Angular - Google-backed SPA framework, uses Typescript [2], clear component division and its reuse, client-side rendering
  - React - Facebook backed SPA framework, uses JavaScript, has huge community and support, components are reusable with React Native to a certain extent
  - Vue.js - Opensource MIT license frontend framework, uses declarative rendering

2.4.2 Mobile app

The mobile app is used as the exhibition's remote control. Shows available playlists, where user can either start, pause or stop them. When developing a mobile application, we must know what platform are we targeting (can be found in requirements 3.3). Nowadays, the vast majority of mobile apps are developed for two main architectures: iOS and Android. The final reasons, supporting evidence and chosen approaches are accessible in chapter design 3 and section decision-making 2.5. Furthermore, in addition to the platform, we should select, whether are we developing a native or a hybrid app. [18]

- Native - developed exclusively for a specific platform, difficult code-reuse during migration to the next platform. Native apps are known for higher performance, require complete access to all the hardware and functionality of a device and has a more complex development process.
  - Android - Java/Kotlin based tools for Android app development. Offers NDK (Native development kit) for native code (C/C++) execution which gives even better control over performance.
  - iOS - XCode and Swift based development is a traditional way of iOS app creation. Evolved from Objective-C, is now a most "direct-control" way of development for this platform.
- Hybrid - merges mobile apps written for web or using another abstraction layer, are not platform exclusive, can share a large portion of codebase

[2]Compiled to javascript

cross-platform. The app is accessed through a native mobile app store while still delivering good performance for certain [3] use-cases. [19]

- Flutter - Google-backed cross-platform mobile framework with widget UI oriented development flow. Supports iOS and Android development uses Dart language and is very performant in both execution and development with the functionality of Hot Reload. Trending in popularity is also part of an established group of technologies.
- React Native - a stable JavaScript-based framework for hybrid app development. Offers code re-usability, follow a lot of "Web React" design patterns React. Uses components that are analogous to widgets in Flutter
- Ionic - Web framework for hybrid app development.HTML, CSS, JS skills are enough to deliver sufficiently performance app, an easy learning curve for web developers [19]
- Xamarin - Microsoft backed platform for creating modern and powerful applications for iOS, Android and Windows. Is build on Mono, an open-source version of the .NET framework

2.4.3 Back-end

It is the system's central point which resolves HTTP requests, control video-players, manages the database and its consistency and lastly performs the actions such as node-synchronization and video-track distribution.

Framework. To make development easy, we are looking for a preferably full-stack framework offering MVC, IOC and easy RESTful functionality which we can leverage during the development process. The considered choices are shown in the list below.

- ASP.NET Core - A free, cross-platform, Microsoft backed, the open-source developer platform for building many different types of applications. Has great community libraries such as Entity Framework Core, .NET Identity Easy and LINQ and is easy to integrate with UI frameworks [4]

- Spring Boot - Provides a comprehensive programming and configuration model for modern Java-based enterprise applications. Great support at the application level, delivers enterprise-level security and stability. Although it comes with a level of complexity, offers unique features. Spring Boot is based on the standard Spring framework but offers rapid project start and many OOTB preconfigured features.

[3] Universal, simple function
[4] React, Angular, Vue.js

- Django - high-level Python Web framework carry rapid development and clean design, good for fast prototyping. Easy to start with, with many tutorials and guidelines
- Node.js - JavaScript back-end framework, is based on an event-driven, non-blocking I/O model, delivered a good performance. Easy to approach and develop with for web developers.

Database. Given the scale of the application, and the fact that data are not changed in a highly concurrent manner, we are well off with the choice of a relational database in opposed to the NoSQL one. Being said, we are left with a choice amongst a relatively large group of databases out of which we picked the ones listed below:

- MySQL - is widespread, widely used, open-source, multi-platform RDMS (Relational Database Management System) well known and tested by the time. Easy to use, comes with many tools, but not compliant with ACID [5] by default.
- PostgreSQL - is an ORDBMS (object-relational database management system) with great concurrency support. Trending, gaining popularity and under active development.
- SQLite - relational database system contained in a relatively small library written in C

2.4.4 Video player

There is a considerable amount of commercial and non-commercial video-players capable of displaying videos of various sizes and resolution. However, as our system requires remote control over the played track, its ability to manage the content of the screen, we need something better than simple playback program. We have, in general, two approaches for content management.

- Content streaming - a certainly plausible solution where we would stream video content to client devices in real-time over the network. However, let us keep in mind that in a situation where each client demands different data, we would have to stream a considerable amount of content at higher numbers of clients. Given the aim at video quality FHD/4K, we would be excessively overloading both server device and the network.
- Content playback - this option means that we would be running a program capable of receiving commands of what to play, pause, stop and jump from track to track, while the content distribution would already be done in advance. Does not put massive load over the network, but the issue of the proper synchronization arises. With the remote control, we must reach a sufficient level of synchronization either by good design or by choosing a specialized player.

[5]Atomicity, consistency, isolation, durability

After drafting two conceptual approaches let us introduce candidates we can use in video playback itself.

- VLC - is an open-source multimedia player from VideoLAN project which delivers elaborate support for many types of video codecs and formats. Shares libraries with the FFmpeg project and offers a valuable set of plugins and functionality developed by the community over the years. Let us emphasize the VLC video streaming functions, a remote control interface (accessible through socket or a network with a telnet-client). It comes with a module called "all splitter" which can split the video into multiple streams and display it on a specific monitor.
- Distributed video player - is a player developed as a bachelor thesis by A. Kuznetsova in 2015 in modular architecture and is based on libyuri[6]. It is a content streaming solution with a good synchronization mechanism. [21]
- MPlayer - is a multimedia player, has a streaming functionality with synchronization mechanism. Each MPlayer instance can play a different video while not going out of sync. [22]
- Syncplay is less a player, but more a plugin for a player network synchronization. The original idea was to synchronize content being watched with friends over the network. What is interesting about this project is that there is no content streaming during the playback whatsoever. Each connected client has the video files present on his device, and only synchronization messages run through the network. Disadvantage is that at a current release [7] there is no function for varied synchronized content playback.

2.4.5 System integration

The server and client HW device choice are out of the scope of this analysis. Nevertheless, the final software solution aspires to be designed as reusable and platform/HW independent. Furthermore, we aim at using open-source and free-licensed software wherever possible. Therefore we prefer OS based on Linux, and we try not to close paths, by choosing platform dependent technology and like not being able to run the system on ARM processors for example. The options for an operational system can be narrowed to two possibilities:

- Windows - Microsoft well-known OS, for platform x86,x64 in its standard version. Version RT runs on ARM, but the software is accessible only through the Windows store.
- Linux based system - is free, open-source, operating system which comes with many distributions, desktop and display managers options. Linux

[6]Framework for parallel multimedia data management, more on [20]
[7]Syncplay 1.6.4a

is highly customizable and the right choice for server-backend or client pc running a specific task.

2.5 Evaluation

This section may be little ahead of the design chapter but gives an explanation for the chosen technologies.

As we do not meet the use-cases, [8] for using a single-site-page website, the use of server-side rendering and multiple page structure is adequate. The pages will primarily be built on the server and delivered as a whole HTML page. That provides better-debugging capabilities, more manageable code, better maintenance and less client code. Having said that, we could exclude front-end frameworks such as React and Angular.

Flutters architecture is based on the prevalent reactive programming (it follows the same style as React), but is UI oriented, allowing rapid development, while following suggested concepts and architecture. The highlighted functions such as Hot Reload (Extremely fast hot-swap between IDE debugging running mobile application) suggest Flutter as a great candidate for prototyping or simple universal projects (such as this one). Flutter [9] now emerges as a proved but still trending candidate as a leader in the mobile hybrid app market. Advantages of the use of Flutter is that the future migration to iOS devices will be very easy and straightforward. A natural disadvantage is that we have an abstraction over native code and that Flutter runs on new programming language Dart.

Given the scale of this application, the potential future work and the need for robustness and maintainability, we inline to the well tested and familiar solution such as .NET or Java as we are experienced with them. The reason why we have eventually chosen Spring boot as our full-stack framework is its flexibility and familiarity for us as developers. It is to be said that both frameworks deliver similar functionality, and choosing .NET is undoubtedly acceptable.

Nevertheless, based on our choice, we follow with suggested front-end template engines which are JSP and Thymeleaf. Very similar usage (scripted code in the .html or .jsp pages), comparable community but we incline to the Thymeleaf as this technology is newer and deliver better performance in small scale sites [16] (our case). Furthermore, we can include Thymeleaf tags into a standard HTML page.

Database choice is not based on many demands as the database in our case serves as a simple data storage and is not queried in high rate. We have eventually chosen the PostgreSQL database as it is an easy to use solution, with ready JDBC adapter which we can use in our Spring Boot project.

As for video playback solution, we incline to the content playback solution in oppose to the content streaming as in this case; we will be able to deliver high-resolution content even in a numerous client system. The limited bandwidth

[8] Responsiveness oriented, linear navigation, small chunks content

[9] Flutter 1.17.0

would be a bottleneck for scaling up the number of screens. As we are having requirements for full HD video playback 3.2 and different content on all screens in real-time streaming approach is not acceptable. Consequently, we have chosen VLC player as it is commonly used and well tested by the community, is able to play a large number of video formats, has remote control interfaces and is runnable on a majority of platforms. The issue of how to achieve sufficient synchronization will be further discussed in the Design chapter 3.

From the reasons given above, we singled out the use of a Linux system for both server-backed and client station devices as the best solution. It gives us needed customizability, and we can tune the system configuration if needed.

To conclude this decision-making, we have now a set of frameworks, languages on which the following Design chapter can build 3. It is important to keep in mind that the decisions to choose solutions in this section are be based on the objectives and requirements of our system. We want to mention that to a certain level all listed technologies in this chapter 2 would be acceptable.

Chapter 3
Design

In this chapter, we will describe objectives we want to achieve, describe possible use-cases follow with requirements formalization. Then the section 3.4 presents composition and core structure of the designed system where division into Web-site, Mobile-app and Server-side is detailed. Furthermore, sections: UI 3.5, Software architecture 3.6 (with diagrams) and Security 3.8 are presented.

3.1 Objective

As is stated in the official assignment, our task is to learn the clients business requirements of The City of Prague Museum newly planed exhibition and based on the requirements design and implement a control system for scalable multi-projection exhibition for an arbitrary number of displays. The system shall be designed distributed, therefore one device assigned to manage another 1-4 displays. Moreover, we need to introduce friendly user interface together with sufficient multimedia content synchronization. Input multimedia content is video tracks, audio tracks and pictures. Next, we need a web interface through which the administrator can manage an exhibition. Lastly, we are to design and implement a controlling mobile application. The emphasis must be put on the design of interaction scenarios between presenter, admin and system. Designed user interface implement and perform testing with the targeted group of users.

3.2 Use case

Visualization and presentation system that allows lecturers to put into a close relationship a global view of the floor plan of the Langweil model with details displayed in a different angle. The individual side views will be formed either by a widescreen projection or by assembling several widescreen screens side by side. Emphasis will be placed on the high resolution of the whole system in order to show the most delicate details of Langweil's model. The system will consist of a projection system from above on a square surface with high resolution (map) and supplemented with 4 side views above the given projection area.

The system is focused on research-oriented teaching, where the primary material is Langweil's model, which is supplemented by other multimedia content and written sources, or material sources. Possible examples of educational programs implemented within the system: What the floor plan of the city reveals - connection of history, geography and environmental education, Impact of remediation on the Old Town of Prague, the City of Prague and the Vltava River, What remains of the Prague fortifications, Prague as a communication node. Topics generally concern the following disciplines: history, geography, environmental education, cultural heritage protection and local topography. [23]

3.2.1 Architectonic vizualization

At the phase, prior to the software design, there has been only a single visualization of the exhibition setup, which can be seen in Fig. 3.1. It shows a frame forming a box with outer sides mounted with monitors. The area map is then displayed on the ground. Later in the development process, the architectonic and product design team improved this setup, and the updated version is to be examined in Fig 3.2. This inconspicuous change of layout has a significant impact on the design outcome, consequently. Nevertheless, it is to be said that the latter might not be the last exposition arrangement.

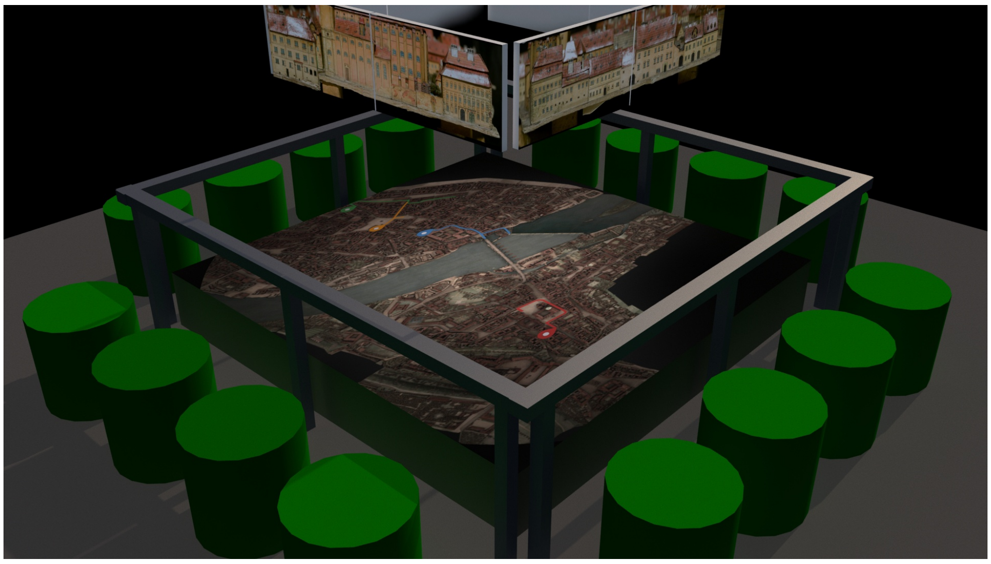


Figure 3.1: Prototype layout

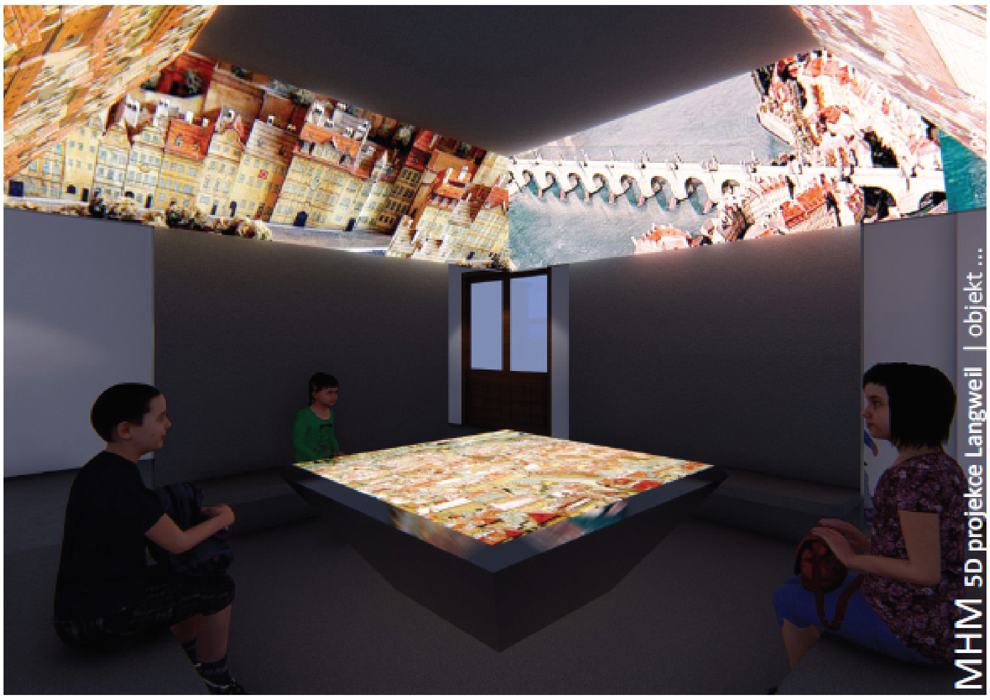


Figure 3.2: Updated layout

3.3 Requirements

3.3.1 Functional requirements

Functional Requirements define the functionality that the application needs to have or the tasks it needs to fulfil in order to achieve the objectives.

F1	The system shall be able to display multiple files on multiple screens at a given time
F2	The system shall have administrator console for maintenance and fine-tuning
F3	The system shall be complemented by a mobile application for control
F4	The system shall have basic control request such as play/pause/next/prev
F5	The admin console shall have the functionality of deliberate display set formulation
F6	Both admin console and mobile application shall have authorization and authentication mechanism
F7	The application shall have an autonomous mode which would play displaySet in loop
F8	The screens control shall be distributed to several client machines connected through network

Table 3.1: Functional requirements

3.3.2 Performance requirements

Performance requirements quantify to what level the functional requirements will be fulfilled.

P1	The system synchronization shall be limited sufficiently, so it is not noticeable in a real-time
P2	The server shall manage communication with a mobile application and web frontend
P3	The server shall manage videofile upload and playlist configuration
P4	The distributed clients machine shall display image to one or n monitors
P5	The distributed client shall run on the same HW machine as server back-end
P6	The mobile application shall be targeted to Android platform
P7	The system shall operate within LAN network
P8	The mobile application shall control exhibition with commands (play, pause, next, previous, go-to item)

Table 3.2: Performance requirements

3.3.3 Design requirements

Aspects that the system needs to fulfil in order to achieve the objectives

D1	The application shall implement the Server-Client architecture
D2	The application shall be complemented with a REST web-service architecture
D3	The application shall be equipped with an automated system of connection establishment and management to the client playback instances
D4	The back-end shall have scheduled job to query the current state of playback
D5	The back-end shall have scheduled job to query whether a client is connected and responding
D6	The back-end shall have scheduled job to query current clients playlist
D7	The playback instance shall be in the form of network-connected device running VLC player in the remote control mode
D8	The application shall be run on Android 9.0 or newer
D9	The back-end application shall be developed in Spring framework using Java language and Thymeleaf template engine
D10	The mobile application shall be developed using Flutter framework using Dart programming language
D11	The application shall follow Spring framework development patterns such as MVC, and Service - DAO architecture
D12	The back-end system shall be containerizable and complemented with docker-compose.yml

Table 3.3: Design requirements

3.3.4 Operational requirements

Operational requirements are requirements that the system has to fulfil to be handled and operated safely and reliably.

O1	The application shall be equipped with the latest software
O2	The mobile devices shall be connected to the network provided by the exhibition
O3	The videofiles shall be compliant with the guidelines stated in the pertinent section
O4	Any user UI input shall be sanitized and validated
O5	The user text input shall be controlled for non-standard characters

Table 3.4: Operational requirements

3.3.5 Constrains

Constraints are those things which limit cost, schedule and implementation techniques available to the developer.

C1	The application shall be completed prior to the exhibition installation
C2	The system shall not be dependent on proprietary software with non-free license options

Table 3.5: Constrains

3.4 Composition

This section describes the system's composition. Starts with a proposed structure and follows with clear division and description of the system components.

3.4.1 Proposed structure

1. Server and clients
   a. Server
      - Management (data input, admin control, user management)
      - Communicates with a mobile application, distributes commands for each client nodes
      - Playlist configuration (education program)

   b. Client player
      - Play video content for one or more monitors
      - Behaves accordingly to servers commands in a time-synchronized manner
      - Can be at the same machine as the hosted server
2. User inputs

a. Mobile application
   - A middleman between the presenter and the exhibition
   - Communicates with server back-end
   - Sends commands such as (play,pause, next, prev, goto item)

b. Web administration
   - Intermediary between an administrator and the exhibition
   - Similar basic commands as a mobile application (serves as a backup control point)
   - Place where playable lists are configured and video files uploaded
   - Maintains application logs, user roles, autonomous mode

Above objective description is based on the preliminary division in Analysis chapter 2 and is then converted into four main points. Mobile application, Server web administration console, client media player, and Back-end framework. As can be seen in the component diagram 3.3, we can observe a division of included components and their relation. Please note that the backbone of our system is the Server station with the running back-end, databases and included storage for the configured video data. The blue highlighted area represents a running docker-compose container that indicates that both database and back-end framework are run as a docker container. The docker-composed container starts even without connected clients but only in a limited debugging mode, during which it throws errors into the console. Such a state can be observed in the administrative web console.

The component diagram description follows with connected client stations (diagram shows two, but its number is not limited by design). They have a running instance of a VLC player in a remote control mode. Furthermore, each client has a dedicated folder for videos and a generated playlists. Instinctively the client device shall be complemented with an associate display screen. The system is designed to work with both 1 or 2 monitors for each client device. After the clients are booted into the initial loop, and after successfully connected of all configured devices, the autonomous playlist is set to play. This feature is implemented due to the fact that clients may boot up in a different time and in various order which would significantly disrupt a playback experience.

Controlling device represents a mobile android based mobile or tablet with a running exhibition control app. App tries to connect to the server in a loop during app start, while the waiting is indicated. Then the data about playable sets are queried from a server, and controlling request send if needed. The users need to log into the app right after the connection establishment.

Administrator console is accessed as a local web page hosted on a server. The user uses a web browser to open the page, logs in and if possess sufficient permissions (see the role division 3.4.3) can perform many tasks. The task can be user creation, roles assignment, password reset, creation of display set and tracks, upload video file, folder management, etc. All these functions are bundled into the web site.

This proposed structure serves as a rough description of a system's structure. The mentioned components, relations and function are better elaborated in below sections, and more comprehensive thorough explanation is in chapter Implementation 4

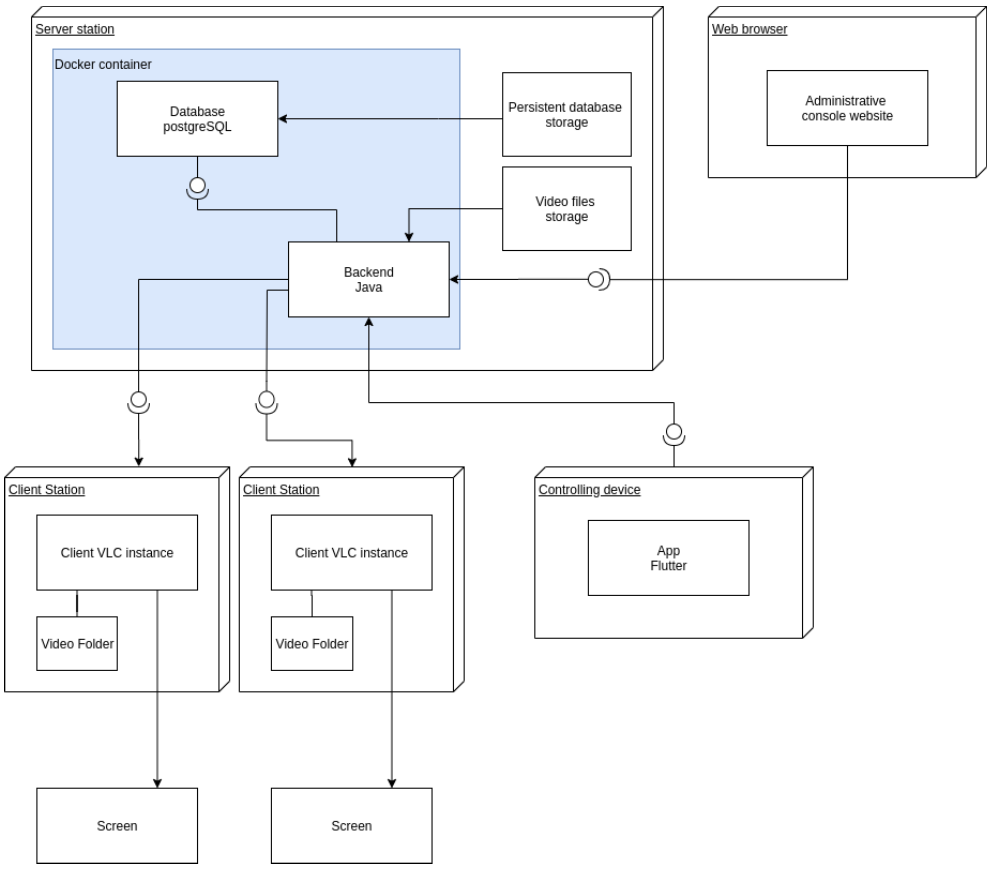


Figure 3.3: Component diagram

3.4.2 Structure of the playable track set

To settle the issue of playlist configurability, we have introduced a terminology for a description of multiple screen playlists, tracks and video files. The final playable set of tracks shall be modelled by a particular object/class/entity. The list of types located below clearly introduces such objects.

- DisplaySet - represents final playable "playlist", contains displayTracks
- DisplayTrack - represents a track for all connected displays. For each display contains a specific videofile
- VideoFile - represents a real playable file in the content of video, picture or audio file.

Please see figure 3.4, where the division of denominations above can examine. It can be observed that the Display sets consist of multiple Display track, while Display track consists of multiple Video files. The number of display track in one display set is not limited, while the amount of video files for each display track is limited by both configurations and also by a number of client devices. The video files are of types of video, picture and audio meaning that the displaytracks can have multiple types. If the displaytrack contains two videos with different duration, always the shorter will be taken in effect (the longer video will be cut). Please note that terms displaySet is sometimes referred as videoSet and displayTrack to videoTrack. This is a legacy terminology which will be eliminated in production.

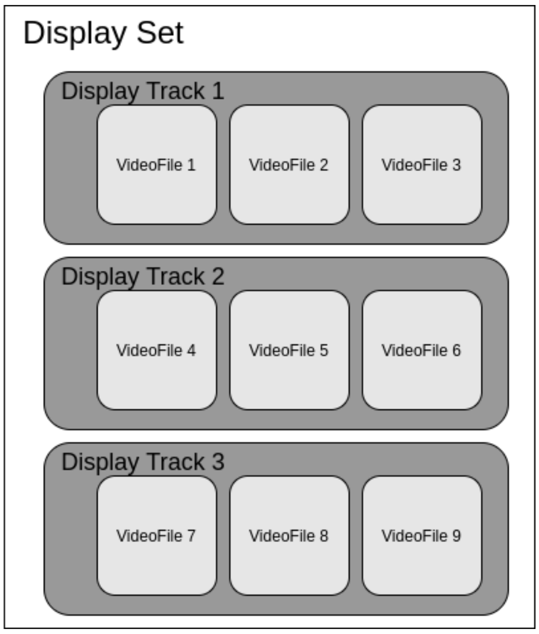


Figure 3.4: Mobile App, first design UI

3.4.3 User roles

As was briefly suggested above. The application should include two types of users differentiable by an assigned role. These roles are listed below:

Administrator. The person responsible for management and state of the exhibition creates new users, assigns presenter roles, manages the current state of an autonomous playlist. Has the ability to edit a created playlist and to create a new playlist. Should receive thorough training so that the person can perform maintenance correctly. Granted full access to the exhibition through a web portal.

Presenter. Person conducting the lectural sessions with the real audience (Either broad od targeted in various groups). Uses the mobile application for the presentation purpose. Typicaly with a good background knowledge on the subject, giving comprehensive commentary and enhancing the experience of the wide audience with the exhibition. Has limited access to the exhibition through web portal (Only simple actions such as exhibition playback control and documentation display are permited)

3.4.4 Mobile App

The requirements which mobile application was to match are mainly F3, F6, O2. Based on these requirements, the final application does need excellent performance, friendly and intuitive UI, ability to list, and control exhibition together targeting for Android platform. We are following a reactive programming pattern which is a programming paradigm oriented around data streams and change-spreading (in general, various asynchronous events). To do so, we will be using a Bloc pattern for state management and proper logic division which is complemented with core concepts of Streams. It is worth noting the difference between other "streams" used in Java. While Java's stream is a part of java.util.stream and is a sequence of elements supporting sequential and parallel aggregate operations, Stream used in Flutter (Dart) is more like a pipe, with two or more ends. When you insert something into the pipe, it flows inside the pipe and goes out by the other end [24]. The mobile app's architecture is tightly built around this Stream concept. [25]

The mobile app is designed with respect to business logic, front-end components division. Starting from the latter, front-end in Flutter development revolves around Widget term which basically means a component (Button, Text, Picture but also more complex ListView, StreamBuilders, etc.) It is a core component, and the screen content is generated by nesting Widgets into each other. That is a suggested approach even though it often creates a numerous line ending bracket "hell". The business logic would be placed in the corresponding Bloc class, which is accessible to the front-end methods. Data querying is to be done through a separated provider responsible only for server HTTP requests and serialization of the received data.

3.4.5 Web site

The web is to be designed as responsive, server-side rendering multi-site page. That said, the web-site shall contain a login system which can incorporate security token retrieval, storage and HTTP request injecting. Used method of security prevention and flows of credential and tokens is elaborated in corresponding section 3.8. Moreover, the web site must have ways of user management (new user creation, roles assignment and password restart) in the form of a site accessible by a clickable menu. The page should have a function to change the language between English and Czech. The internationalized texts are taken from the specific configuration file.

Moreover, the user needs a feedback regarding currently played track, its name, length current time of playback, information about to which display set it belongs and so on. This information is received using the process of querying (please see full description in 3.4.6) and is displayed in the form of a banner.

3.4.6 Back-end

The server is based on the MVC design pattern and follows the division of business logic from data accessing services. Also is to advantage from IOC design pattern and dependency injection mechanism. All these tools are leveraged for a server-side rendering website using a template engine, RESTful architecture for communication of the created display sets, display track, or video files following by endpoints handling playback control. [26] Furthermore, the server is responsible for the client player communication establishment, maintenance and closing. In case of our chosen player [1] in remote control mode, it opens a socket into the network allowing telnet connections and control. Using telnet connection seems nowadays little deprecated thanks to the lack of encryption. However, for our purpose, it is acceptable as the network is private and without access to the internet. Lastly, the risk of exploitation is low as the socket is open only to the VLC rt interface. [27]

Playback status query

The back end system is responsible for a periodic querying to the clients and getting the information about current playback status. It can be done in two styles. Either the server can query **all** clients and get the latest information or query **only the first active** client. This division was designed for the testing purpose, and based on the evaluation process, we have not noticed any significant deficit. Moreover, the interval of querying is configurable but advised to be smaller than 500ms as the measurable change is precisely one second, applying Shannon theorem of sampling frequency. Then the front-end clients (mobile app and website) queries the webserver for the most updated playback status, in whatever period they see fit, and will get a correct value.

Video distribution and synchronization

For a robust process of playlist distribution, we have introduced a new two terms, both representing a specific action for a displaySet:

- Publish - is run on a specific display set, builds all display tracks, and generates playlist
- Synchronization - is run on all published display sets. It is a process of video file and playlist distribution from server to client display devices.

The configured display sets are until successfully published and synchronized present only on a server-side. For that reason concept of state-management was introduced. As can be seen in figure 3.11, the above-stated terms have an exact place in this state machine. Section 3.4.7 explains the process of video file renaming and further details. The file structure created on the server is detailed in the following section, and figure 3.5 shows its visualization. The process of synchronization is related to its implementation and will be further elaborated in the implementation chapter 4.3.1.

[1] VLC see chapter analysis 2

Folder structure

We shall distinguish between folder structure in server and clients. Both the server and client contains folders *vid* and *playlists*. The difference is that while in the server, folder vid contains video files following the structure of a video folder created by the user, the client has a distinct generated folder structure which can be seen in figure 3.5. Nested folder naming follows a displaySetId and displayTrackId convention. The server's playlists folder contains subfolders in the amount of configured clients. Each client subfolder then contains a corresponding playlists which are uploaded to the matching client during a the synchronization event. VideoFolder - represents the real structure of directories, into which the video files are uploaded. Is mirrored in database and user can recursively nest folders in each other. This folder structure would be useful in future when the system contains a large number of video files and would help users to navigate and search according to the folder.

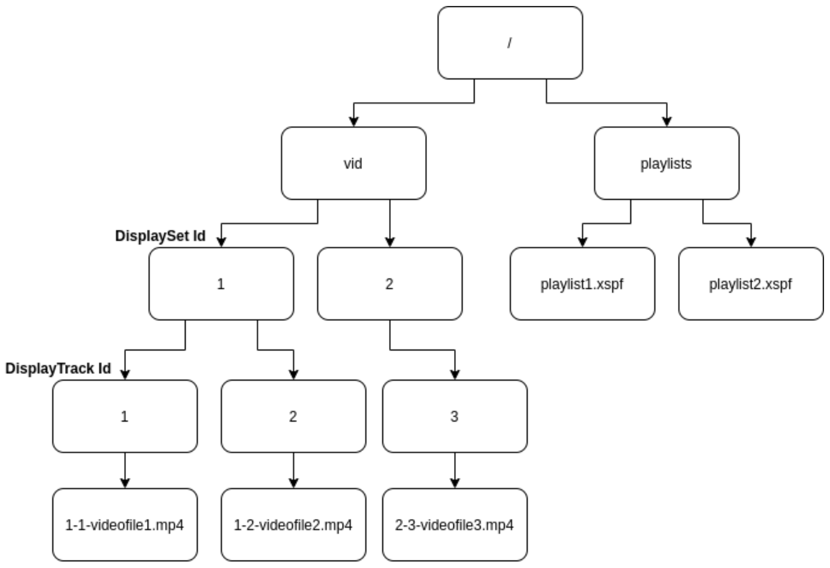


Figure 3.5: Clients folder structure

3.4.7 Video files

Every uploaded file is being validated concerning its type (video, picture, audio) using video analysis tools such as FFprobe [2]. Moreover, the file name is validated for non-standard characters and saved into the server storage. Lastly, after the verification, the database record for this video file is being created.

[2]FFprobe is a part of FFmpeg

File naming conventions

The way, video-files are named in client devices can be observed in figure 3.6. The first number represents the id of an existing displaySet while the second number represents the id of a displayTrack found in relation with displaySet. The last string name is a filename of a video-file chosen during the upload process.

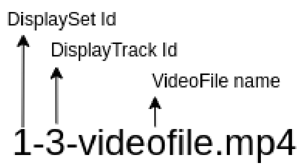


Figure 3.6: Naming conventions

Guidelines

Are to be further specified based on the production layout of the exhibition. In the below statements, our declarations are based on the last-know architectonic disposition of the monitors. Which is four computers each with 1-2 displays, forming a square as shown in the figures 3.1 and 3.2

VLC player is capable of playing the vast majority of video types. Despite that, the list below is narrowing the options for the most suitable ones.

1. Video - Video file up to 4K resolution 3840 x 2160 pixels for a single screen or 7680 x 2160 pixels for 2 screens. Encoded in common codes (WebM VP8/Vorbis, MP4- H264 for FHD and WebM-VP9, MP4-H265 for 4K)

2. Picture - preferred file type jpg, jpeg, pgn

3. Audio - in common file types and codec MP3, AAC, MPEG-4

3.5 User interface

Thanks to the recent rapid development and evolution of UX science and design. The broad audience has become much more demanding in regard to conventional standards and expectations. In our design, we tried to follow the guidelines from [28]. We firstly introduce expected user action flow, then describe the process of design with the use of wireframes and mockups. The user design for web and mobile platform is very different. Despite the web is developed as responsive (can be operated on a mobile device) we target at administrators which are using a computer with keyboard and mouse.

3.5.1 Wireframes and mock-ups

During the requirement decision process, we came with a proposed way of mobile application appearance, which was presented to the client and on which the real implemented product is based. We have designed the wireframes which are visualized mocked-up appearance of the product. In our case, we inclined to interactive mockup and used Fluid [3] reference software for the design. As it happens during the development process, implemented mobile app has differentiated in certain areas. However, the original UX draft has been preserved. Thanks to the chosen framework Flutter, which offers many ready widgets based on popular Material design, the design has embraced this prepared components and assembled them together.

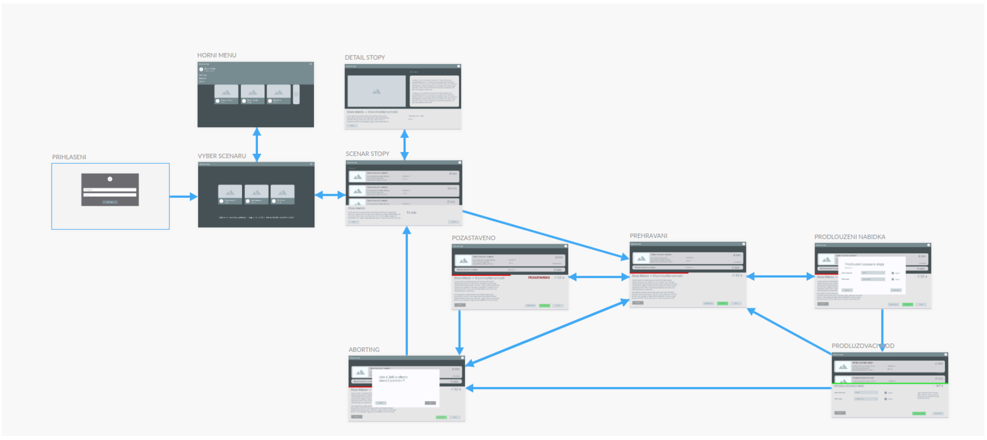
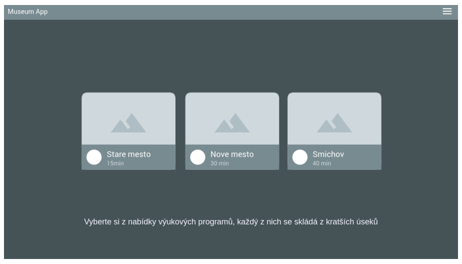


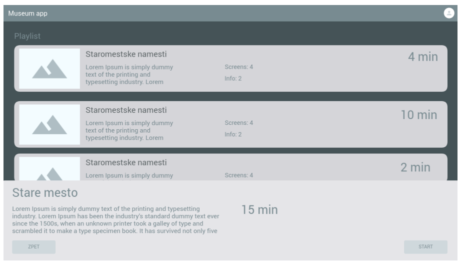
Figure 3.7: Mobile app screen relation


Museum App

Stare mesto 15min
Nove mesto 30 min
Smichov 40 min

Vyberte si z nabídky výukových programů, každý z nich se skládá z kratších úseků

(a) Homepage with playlists


Museum app

Playlist

Staromestske namesti — Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem — Screens: 4 Info: 2 — 4 min

Staromestske namesti — Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem — Screens: 4 Info: 2 — 10 min

Staromestske namesti — Lorem Ipsum is simply dummy — Screens: 4 — 2 min

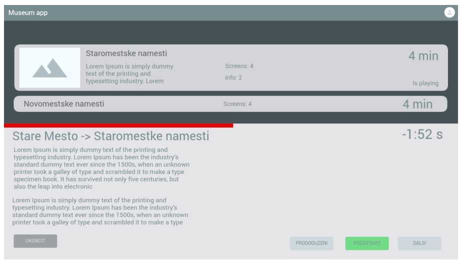
Stare mesto

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five

15 min

ZPET START

(b) Playlist detail


Museum app

Staromestske namesti — Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem — Screens: 4 Info: 2 — 4 min — Is playing

Novomestske namesti — Screens: 4 — 4 min

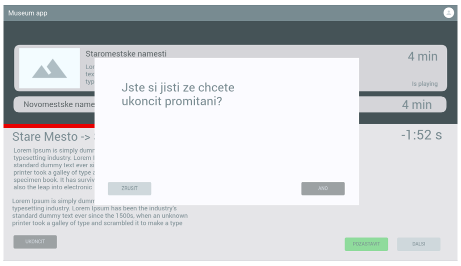
Stare Mesto -> Staromestske namesti — -1:52 s

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type

UKONCIT PRODLOUZENI POZASTAVIT DALSI

(c) During play mode


Museum app

Staromestske namesti — 4 min — Is playing

Novomestske namesti — 4 min

Stare Mesto -> — -1:52 s

Jste si jisti ze chcete ukoncit promitani?

ZRUSIT ANO

UKONCIT POZASTAVIT DALSI

(d) Popup when exit

Figure 3.8: Mobile app UI design

The mockups for web development have been rough and were not formalized

[3] A browser-based wireframing and prototyping tool

in the same manner as for the mobile application. We are to use a bootstrap toolkit, which again comes with many predesigned UX components such as navigation, alert boxes, buttons, card, and more.

3.6 Software architecture

This section describes systems architecture regarding software design. The general division is based on the diagram types: structure, behaviour, interaction and entity diagrams.

3.6.1 Structure diagrams

Emphasize the things that must be present in the system being modelled. Structure diagrams further split up into components and show the dependencies among these components. They are an excellent tool for documenting software architecture. For example, the component diagram below describes how a software system is wired together.

Component diagram. A solid source of overall block interference can be considered a Component diagram accessible in the figure 3.3 above. It presents a high-level picture of the system components in the lead with the server and clients blocks followed by a controlling device with running mobile app and web browser with administrative console for finer tuneup of the exhibition. It is worth noting, that the blue highlighted block represents a running docker-compose instance of back-end and a database. The second diagram in fig 3.9 describes the same components in slightly different fashion and is focused on their interaction. [29]

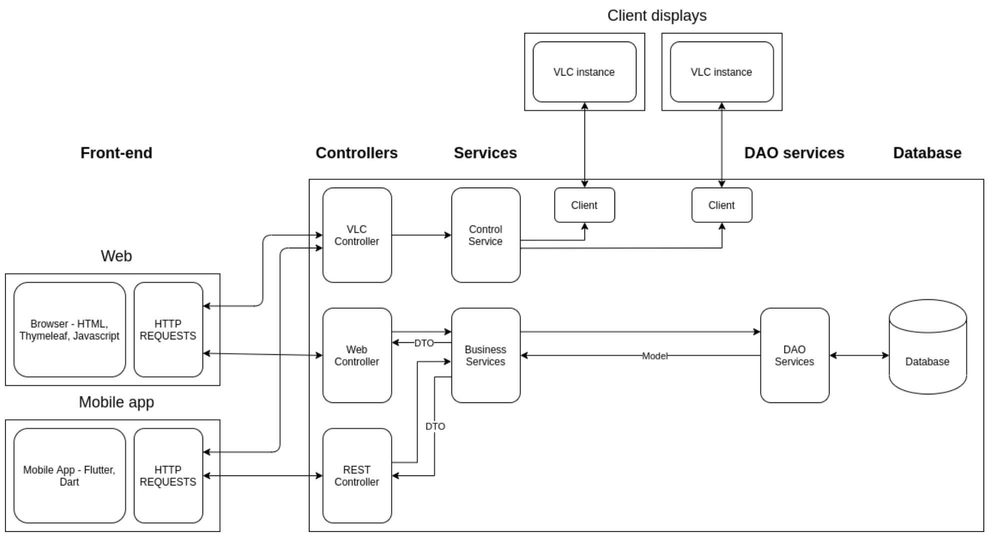


Figure 3.9: Component structure diagram

Class diagram. Is a UML diagram for the graphical representation (modelling) of classes, interfaces and their relationships. By the definition of the OOP, the class is a program-code-template description of an object structure. Therefore, the class diagram describes the static structure of the system, namely the variables and functions and their control access. The class diagram describing a subset of the system, figure 3.10 is deliberately focused on playback control mechanisms of this system. It is to be said that boldly highlighted service variables represent an autowired instances of the object out of the IOC container. Moreover SCHEDULED text after function name significant its periodic execution in a set interval (see section 4.1.2). As can be seen in the class diagram, there are five services in various mutual relations and a client class named VlcClient implementing the RemoteClient interface.

Based on the proposed display set structure in the section 3.4.2, we have introduced a service for each of the DisplaySet, DisplayTrack and Videofile objects: *DisplaySetService, DisplayTrackService, VideofileService.* It serves as an intermediate between the front-end management (controllers or other services) and database consistent records. Furthermore, it holds various business logic such as: retrieval of an object by its id, retrieval all object of a certain type, creation object and other various use cases. *ControlService* holds the instances of the RemoteClient, has an implementation of the majority of available commands (play, pause, next, prev, repeat). *ClientService* is an abstraction service for information about clients and their state of playback (current track, played time, track length, isPlaying). Function getPlayedTrack() returns a snapshot of the current playback. *VlcClient* class is responsible for communication with the client VLC itself. It has scheduled jobs for connection establishment and its maintenance. Holds an instance of the open Socket, manages command delivery and its response propagation. VlcClient also inherits from RemoteClient variables isConnected and isPlaying for clients handling.

The overall system structure is far more complex and would require many more class diagrams for a full UML description. However, we are strictly following Spring framework patterns and standard software paradigms such as business logic services and DAO services separation. Moreover, by leveraging the Spring framework tools: IOC (dependency injection), MVC, Spring Security. We minimize the risk of a bad design as these tools are well integrated and documented in Spring documentation [30]. The detailed version of how the system is implemented can be seen in chapter Implementation 4

State diagram. Is a mean of graphical record of the system having the final number of states. Such system can be a final-state machine. In our case, the state diagram shows states into which the displaySet can enter. It holds two variables: isPublished and isChanged and is initialized with both as false. In the event of new displayTrack assignment, isChanged is set to true and until the displaySet is published cannot leave this state. After publish, displaySet has generated a playable list. Consequently, any changed tracklist (adding new displayTrack) must result in a different state, that is the one which both isPublished and isChanged set to true. Again the publish generates a playlist,

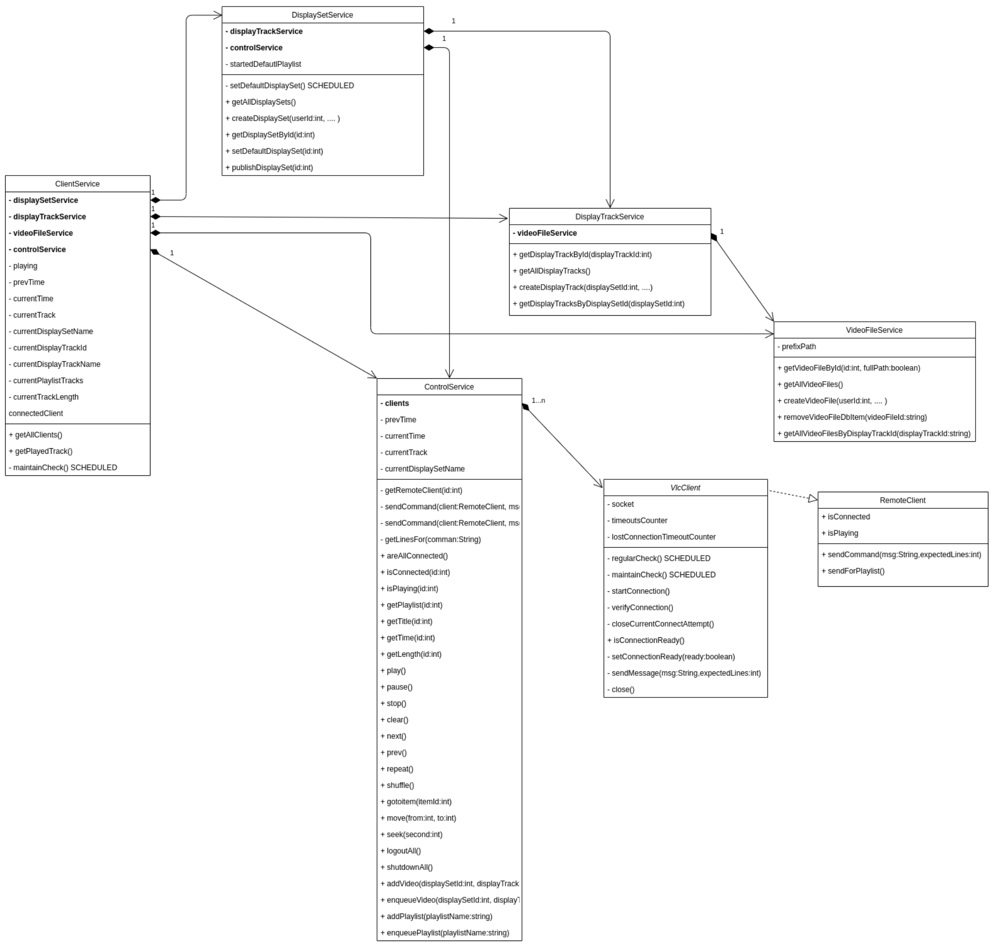

DisplaySetService
- displayTrackService
- controlService
- startedDefautlPlaylist
- setDefaultDisplaySet() SCHEDULED
+ getAllDisplaySets()
+ createDisplaySet(userId:int, ...)
+ getDisplaySetById(id:int)
+ setDefaultDisplaySet(id:int)
+ publishDisplaySet(id:int)
ClientService
- displaySetService
- displayTrackService
- videoFileService
- controlService
- playing
- prevTime
- currentTime
- currentTrack
- currentDisplaySetName
- currentDisplayTrackId
- currentDisplayTrackName
- currentPlaylistTracks
- currentTrackLength
connectedClient
+ getAllClients()
+ getPlayedTrack()
- maintainCheck() SCHEDULED
DisplayTrackService
- videoFileService
+ getDisplayTrackById(displayTrackId:int)
+ getAllDisplayTracks()
+ createDisplayTrack(displaySetId:int, ...)
+ getDisplayTracksByDisplaySetId(displaySetId:int)
VideoFileService
- prefixPath
+ getVideoFileById(id:int, fullPath:boolean)
+ getAllVideoFiles()
+ createVideoFile(userId:int, ...)
+ removeVideoFileDbItem(videoFileId:string)
+ getAllVideoFilesByDisplayTrackId(displayTrackId:string)
ControlService
- clients
- prevTime
- currentTime
- currentTrack
- currentDisplaySetName
- getRemoteClient(id:int)
- sendCommand(client:RemoteClient, ms
- sendCommand(client:RemoteClient, ms
- getLinesFor(comman:String)
+ areAllConnected()
+ isConnected(id:int)
+ isPlaying(id:int)
+ getPlaylist(id:int)
+ getTitle(id:int)
+ getTime(id:int)
+ getLength(id:int)
+ play()
+ pause()
+ stop()
+ clear()
+ next()
+ prev()
+ repeat()
+ shuffle()
+ gotoitem(itemId:int)
+ move(from:int, to:int)
+ seek(second:int)
+ logoutAll()
+ shutdownAll()
+ addVideo(displaySetId:int, displayTrack
+ enqueueVideo(displaySetId:int, display
+ addPlaylist(playlistName:string)
+ enqueuePlaylist(playlistName:string)
VlcClient
- socket
- timeoutsCounter
- lostConnectionTimeoutCounter
- regularCheck() SCHEDULED
- maintainCheck() SCHEDULED
- startConnection()
- verifyConnection()
- closeCurrentConnectAttempt()
+ isConnectionReady()
- setConnectionReady(ready:boolean)
- sendMessage(msg:String,expectedLines:int)
- close()
RemoteClient
+ isConnected
+ isPlaying
+ sendCommand(msg:String,expectedLines:int)
+ sendForPlaylist()
1
1
1
1
1
1
1
1...n

Figure 3.10: Class diagram

and finally, the Synchronize action distributes the playlists and video files to connected clients.

3.6.2 Behaviour diagrams

Behaviour diagrams emphasize what must happen in the system being modelled. Since behaviour diagrams illustrate the behaviour of a system, they are used extensively to describe the functionality. As an example, the activity diagram describes the business and step-by-step operational activities of the components in a system.

Activity diagram. Describes certain behaviour in a particular way, show information and logical flow of the action visually. It is beneficial as a workflow snapshot. The included diagram in figure 3.12 depicts the video file upload process. Firstly, the administrator selects a file and sends it to the server which performs validation and based on its outcome either accept and save to storage and creates database record or notifies the admin about the invalidation.

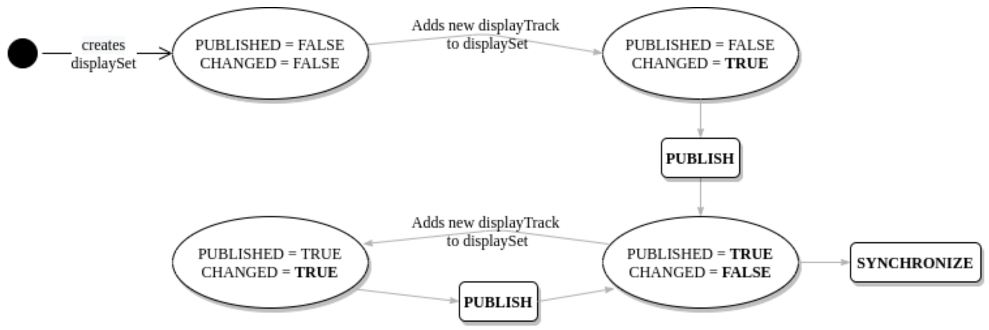

creates displaySet
PUBLISHED = FALSE CHANGED = FALSE
Adds new displayTrack to displaySet
PUBLISHED = FALSE CHANGED = TRUE
PUBLISH
Adds new displayTrack to displaySet
PUBLISHED = TRUE CHANGED = TRUE
PUBLISH
PUBLISHED = TRUE CHANGED = FALSE
SYNCHRONIZE

Figure 3.11: Display set state diagram

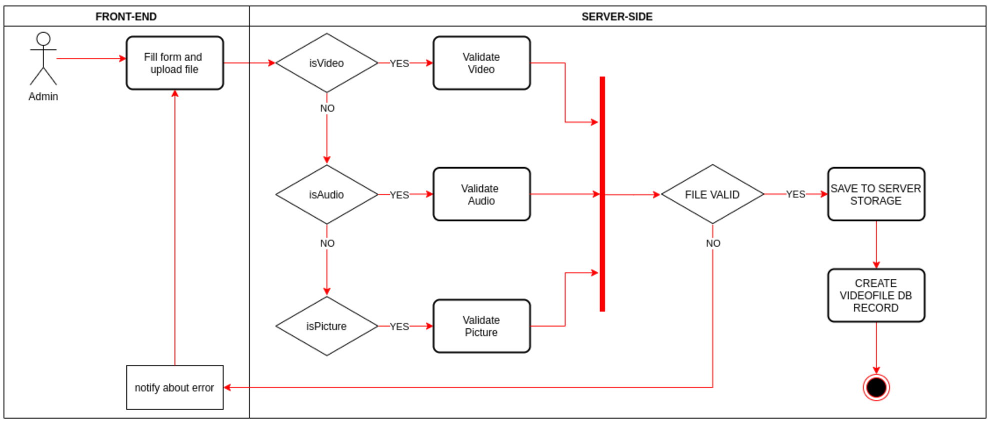


Figure 3.12: Activity diagram of Videofile upload

Use case diagram. Is the representation of a user's interaction, we can observe in figure 3.13 that the Presenter using a mobile app can perform two general tasks. First, to query available display sets and secondly to control the VLC instances through the HTTP request to the server. On the other hand, administrator accessing through the website can perform many more actions such as uploading files (see activity diagram 3.12), display set creation and display track creation. Naturally he can also control VLC instances but through the website interface. In this simplified diagram, only these essential functions have been highlighted. A more thorough description of user-system interaction can be seen in sequence diagram 3.14

3.6.3 Interaction diagrams

Interaction diagrams, a subset of behaviour diagrams, highlight the flow of control and data among the things in the system modelled. For example, the sequence diagram shows how objects communicate with each other regarding a sequence of messages.

Sequence diagram. Is a diagram showing interactions among objects arranged in time sequence. That is especially useful when looking for an actions occurring in successive order. We are presenting sequential diagrams. The

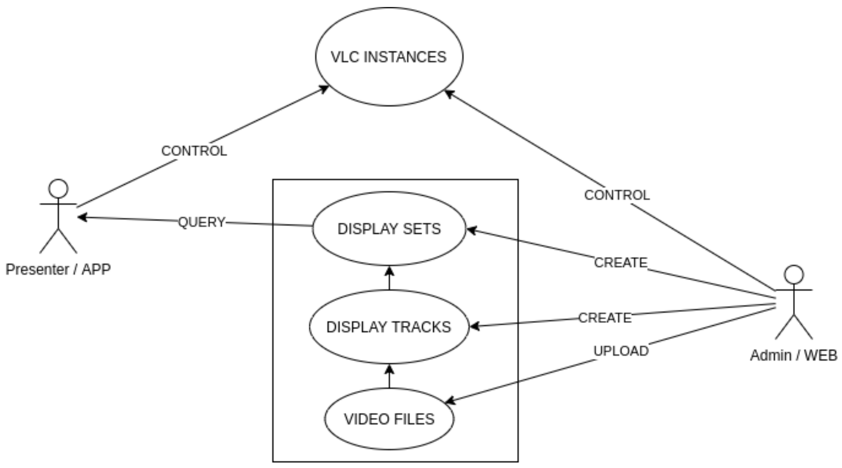


Figure 3.13: Use case diagram

first diagram, see figure 3.14, depicts the process of display set publishing and consequent synchronization. For a full description of the states, see state diagram 3.11. The administrator has prepared an unpublished set, then observe the actions in the group going through the MVC Controller, and various business-logic services. Then the Playlist service access DisplaySet-DaoService, which is an intermediary between system and database. After successful publishing, the client invokes the synchronization, which triggers the process in SynchronizationService. This process is very time demanding and simplified transmits the video files to the clients. Please see the section 4.3.1 about its implementation.

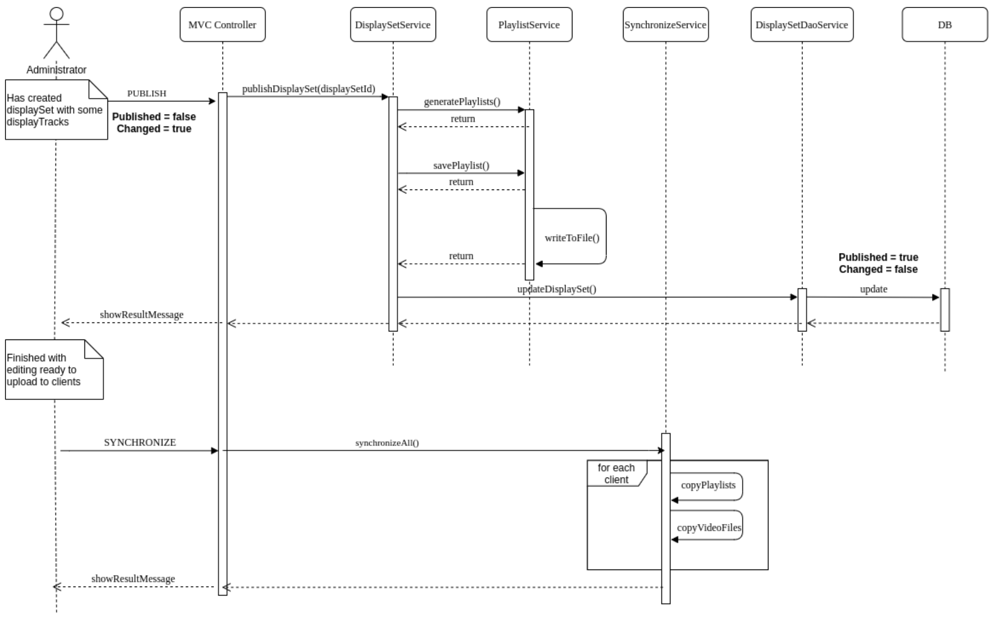


Figure 3.14: Sequence diagram - synchronization

The second sequence diagram, see fig. 3.15 shows the process of command

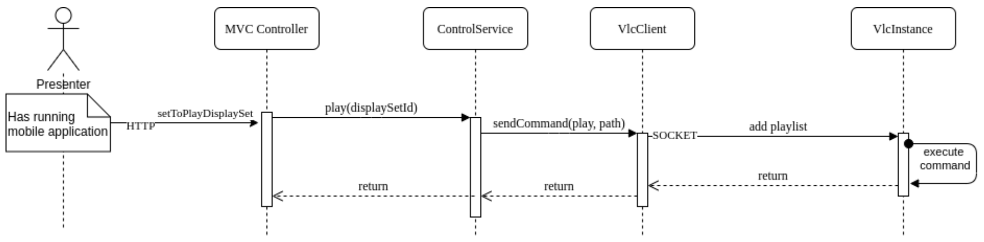


Figure 3.15: Sequence diagram - play control

sending from front-end user mobile app through controllers, services through multiple VlcClient objects, sending a command to client devices running real VLC instance in the remote control mode.

3.6.4 Entity relations

The entity-relationship diagram can be seen in fig. 3.16 and is based on the Class diagram above. It gives into relation all database saved objects with the exception of Logs table which has no relations with other tables. The diagram is self-explanatory.

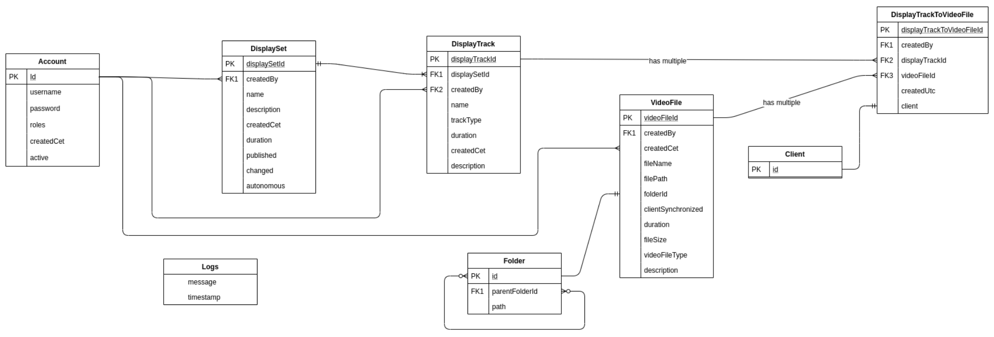


Figure 3.16: Entity relationship diagram

3.7 Documentation

This section describes places and locations where the information about the developed system can be found. Appart from this theses, we have provided a collection of API endpoints, commented code (public classes and significant methods) and lastly included user-related documentation into the website as a new tab.

3.7.1 API documentation

All back-end implemented endpoints have been, aside of git repository, recreated in Postman software and are included to this work in the exported format of JSON file Collection 2.1v. They are valuable for testing and overall API archiving. As these endpoints are not publicly available, the need for thorough documentation in the form of Swagger, RAML or Oxygen tools has not arisen.

3.7.2 User documentation

We have created 2 types of user documentation.

1. Thesis - serving as a technical, design description of the development process. Is particularly useful for future engineers working on this project or for technically educated individuals.

2. Web-hosted documentation - easy to use, describing business scenarios rather than a technical description. Serves as an endpoint for both tutors and administrators to which they look for answers in regard to this exhibition. The importance is set on the process of playback and creation of displaySets and its maintenance.

3.8 Security

The designed system is to be operated in a private local network without the need for uninterrupted internet access. Having a one secured WiFi network (to which exhibition and control device are connected), the security standards may be slightly relaxed in contrast to the other 24/7 uptime websites which are publicly accessible. Nevertheless, we have introduced several standardized concepts to protect implemented systems against most common attacks (namely SQL injection, XSS). For that, we will use the chosen frameworks library, which in detailed 4.1.2. Moreover, we are protecting both website and mobile application with an authorization and authentication mechanism. Secondly, all user input and REST input coming to the controllers is validated using the regular expressions. For example, matching only predefined patterns or allow only numbers as an id value.

3.8.1 JWT tokens

Jwt token is a new standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. Thanks to the fact that the token contains a digital signature, we pass the identity of the authenticated user in all requests. They are designed to be compact, safe and easy to use.

The reason why we inclined to this approach is that we need a Single sign-on scheme. The other means of authorization eg. Basic is not suitable or have larger overhead. Single Sign-On is a feature that widely uses JWT nowadays, because of its small overhead and its ability to be easily used across different domains. [31]

Chapter 4

Implementation

This chapter describes the implementation of the system using specific tools, frameworks and languages which are to be reminded in the upcoming section. Then is followed by the presentation of Mobile application, Website, Server-side backend and lastly playable VLC instance.

4.1 Used tools and frameworks

Based on the analysis 2 where we have concluded the most suitable tools for development, and design chapter 3, where these tools were utilized in a way to target objectives. This section gives detailed inside into the way we have used the frameworks.

4.1.1 Flutter

Flutter 1.17.1 is both a development tool and UI Widget library. Great for development of mobile apps as well as web apps (is compilable to javascript). We have used OOTB prepared Material components such as AppBar, Lists, Buttons, Texts, GridView, Side-bars and more. The process of development in Flutter is straightforward and easy when grasped core concepts and techniques. Basically, the developer includes and configure Widgets according to the needs and incorporate them in the screen. The primary source of guidelines is Flutter documentation where the examples, cookbooks and tips for development can be found [32]

Dart. Used in version Dart 2.8 is an essential part of Flutter development. It is a typed, OOP, programming language based on Java syntax, with classes and garbage collection. Its use is almost exclusively for a Flutter app development.

4.1.2 Spring

As for the Spring framework, we are using Spring Boot in version 2.2.6.RELEASE with a set of starter libraries for web, templating, security, SFTP integration, database connection and testing.

Java. Using OpenJDK Java in the most recent version Java 14. The current Java releases have been changed into feature-release with a new version coming out every six months. The last update has brought new switch syntax, NullPointerException improvement, and more.

Scheduled jobs. One of the noteworthy features, which allows asynchronous method execution in the form of scheduled scenario. [33] This can be done either in a fixed interval, delay or as a cron job. This function is used for playback status querying 3.4.6 and for VLC client connection establishment and maintenance 4.5.1. Scheduled fixed-rate waits for previous to finish, then execute itself.

Thymeleaf. Is a template engine developed for the Spring platform, is great in performance and offers easy thymeleaf tags integration into native HTML pages with prefixes such as *th:id*. The reusable component is called a fragment, and its usage is possible by simple calling

```
<div th:replace="videoSetFragments :: displaySetCreate"></div>
```

where videoSetFragments is an HTML file with multiple fragments, a displaySetCreate is a fragments name. This fragment is then injected into the page and replace this div tag. Based on this principle, enhanced library Thymeleaf Layout Dialect uses layout/decorator templates to style the content, as well as it can pass entire fragment elements to included pages. Using thymeleaf dialect, which is useful for better control over displayed content and template nesting in each other [34].

Spring security

Spring Security is a powerful and highly customizable authentication and access-control framework, with features such as protection against session fixation, clickjacking, CSRF together with MVC integration. We are protecting our system from the XSS attacks using build-in HTML escaping [1] and authentication and authorization are performed using JWT token (see security section 3.8). The JWT token is returned from `/login` endpoint on POST request with correct credentials. This token has configurable expiration set on ten days and is encrypted using a secretKey form configuration file. Furthermore, all passwords are saved in an encrypted format using BCryptPasswordEncoder to the database.

Moreover, we are protecting the URL mappings with a user roles. That means that while the logged administrator with a valid admin role can access the resource at `/user` and create a new user. The authenticated user with only presentator role assigned receives 403 HTTP response forbidden.

[1]Changes non-standard characters into ASCII table

4.1.3 PostgreSQL

If the database is empty during system startup is automatically initialized from *V1.sql* script using Flyway plugin for Spring Boot library. The mentioned script builds tables and creates relations between according to the entity relation diagram 3.16. Usernames and passwords for connection to the database are saved in a configuration yaml file.

4.2 Mobile application

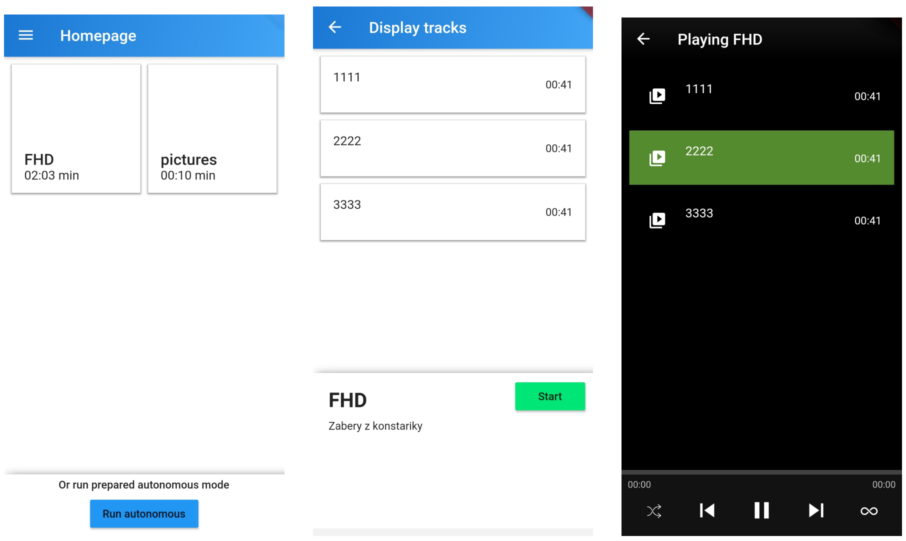
The app is written in a Dart programing language and is built on top of Flutter framework. Follows recommended software patterns such as Bloc [24] which is used for state management, UI separation from services and more. The reasons for choosing this framework are further described in analysis chapter 2 and design elaborated in 3.4.4. This section will describe the final implementation and application of the design concepts. Furthermore, see the repository [35], where the source for the mobile app is located.

The mobile application starts in the Splash screen, where it connects to the exhibition using IP address and HTTP requests for connection establishment. Waiting is highlighted with animated spinner and text notification. When the server responds with request other than Connection refused or Anything lower than 500 (Server-side errors). App automatically redirects to either login screen or homepage. This decision depends on whether the credentials have been saved during login and user is redirected to the homepage. Or whether there is no remembered combination of username and password and user is redirected to loginpage.

After a successful login app redirects to the Homepage 4.1(a) which is a main gate into the application. This screen contains a sliding side menu where additional configurations such as redirection to administrator web, logout and light/dark mode switching are placed. However, the most important content on the screen is the list of playable display sets and contains a button for return to Autonomous menu. By clicking to a display set tile user is redirected to screen 4.1(b) which lists display tracks for the selected display set. Clicking on the start button sends a command to the server about the initiation of the playback. That consists of stopping a currently played playlist and queuing the selected one in a paused state. The user is then redirected to the playback screen 4.1(c), where the exhibition starts to play the display set on his play-button press. Note, that the currently played track is highlighted in the green colour and periodically queried playback status is displayed on the playback panel below. The exhibition's control is operated exclusively from this screen and can be divided into two controlling mechanisms. Firstly, the bottom panel containing Shuffle, Previous track, Pause/Play, Next track, Loop current track, actions which are also part of the slidable panel where more functions and information can be displayed. Secondly, user can click on the listed tracks, triggering control event and setting the track to play.

On the event of going back from the playback screen, the user is prompted with a message verifying his intentions to leave the currently played display set. If answers no, he stays on the page, on the other hand, if he answers yes exhibition playback is stopped and he returns to the screen 4.1(b). After the tutorial session is finished presenter can return to the autonomous mode clicking on the button (Run autonomous) on a homepage.

The mobile application contains both light and dark mode, and the playback control is operated exclusively in a dark mode. This feature is useful on the low-light conditions, especially during the presentations, the bright light may distract the young audience or the tutor himself. Figure 4.1(a)



(a) Home page (b) Display set detail (c) Playback screen

Figure 4.1: Mobile app screenshots

4.3 Back-end

The implementation of server-side is done using Spring Boot framework and follows a Spring design patterns as stated in design requirements 3.3. Based on the MVC structure, we have Controllers serving HTTP request coming from the web and mobile app. The requests are validated, processed and forwarded to the business level services. Next, if needed to access the database, the modified request is forwarded to DAO level services and followed by a database query. The received data validation is again done mostly in the business services. [36]

The configuration is saved in the `application.yaml` file, where we mirrored config for two profiles. Each for different development run scenario. We have a dev profile for running on a native platform with dependence on running and configured PostgreSQL instance[2] and a dock profile which is triggered on a docker-compose command on a `docker-compose.yml` file. This solution

[2]Can be run from docker

is convenient as it gives much flexibility. For example, the usernames, IP address of the client device can be configured there.

Logging is done in two ways. Firstly, the system logs are saved and archived using logback tool, and secondly, business logs of actions playlist played, playback stopped are saved into the database and can be listed in the website in menu Logs. The overall current state of the server source code is in the repository [37].

File uploading

The file being uploaded is immediately validated on both client and server sides. While client-side, the file size is compared to the maximum size set in the configuration, in server-side using FFmpeg tools we measure resolution, type but mainly duration of the video file which is saved into the database. As of this stage of development, any file can be uploaded to the server; it comes as a convenient solution but bares a risk of exploitation in a publicly accessible system. We have implemented a permission change, and all files are set as not executable. This should be tested for a large number of files, and later we can state this as a safe solution.

Dockerized back-end

The server is packaged as a jar file and is run on a `ubuntu:eoan` with installed openjdk14 and ffmpeg tools for video validation. The reason for not using OpenJDK image from the beginning is that we need a capability of Unix style command execution and directory volume matching for video file and playlists. In docker-compose, we run an instance of PostgreSQL in a dependency, allowing virtualized network connection and overall portability.

4.3.1 Publish and Synchronization

As was suggested before, and can be observed in the state diagram 3.11 values of isPublished and isChanged decides the state in which all displaySets stay. To not repeat ourselves, the publish action validates display set and generates playable playlist in the form of `.xspf` file. This file format is a default for VLC media player playlist and has a form of an XML document. The *PlaylistService* is responsible for this actions.

The event of synchronization lasts between 5 - 20 minutes, and is highly dependant on the number of video files, display tracks and display sets. Such behaviour was expected because we need to transfer a significant amount of data to each client over SFTP within a network. It is to be said that there are ways of optimization, but as we are still prototyping, this will be an assignment during production deployment.

4.4 Website

Website is implemented using the Bootstrap library, which is dependent on the jQuery. The site is responsive and allows site navigation and requires authentication. The landing page of the website is a site containing a short description and logging button where an anonymous user is directed on a root HTTP request. Once there, the user can log in using a valid combination of username and password for an active client. After a successful login, client-side receives a generated JWT token and saves it as a cookie. This cookie is attached to each HTTP request from the client-side and servers as an authentication mechanism. See section 4.4 for details of this attachment. After the login, the user is redirected back to the main page see fig. 4.2 which filled with more content and lists configured clients, their status of connection and playback. Below, the client table is a panel displaying the currently played track, time played, and queued playlist. If no clients are connected only text "No connected clients" appears.

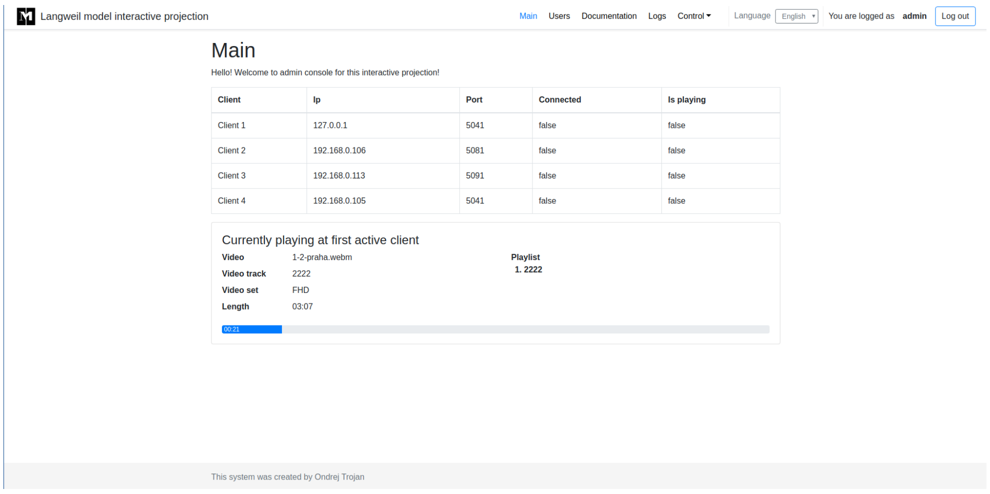
The navigation is located in the upper right bar where the menu contains buttons dependent on the assigned role. In the case of the admin role, the links to User, Documentation, Logs and Multiple control pages are present. If the user lacks ADMIN role, is only capable of simple actions such as listing client devices and control of playback. Moreover, the upper bar contains language switch and a logout button. On the click to the User button, the operator is taken to the administrative user page where the operations of user creation, role assignment, password reset and enabling and disabling selected users can be performed. The documentation is a page where business-related guidelines are located according to the documentation division stated in section 3.7. This page contains a secondary navigation menu, which helps to navigate in long documents. Logs site displays saved events such as playlist set to play, pause request, play request. Furthermore, it is a paginated list of the log message and a date.

Control drop-down menu contains three sub-sites. Firstly, a general control page which can be seen in fig 4.3. Actions such as Play, Pause, Next track, Previous track, return to autonomous together with playlist enqueue command. The same banner containing playback information is displayed here. The red buttons with Restart text in section Emergency serves as a command for VLC instance shutdown. Nevertheless, as we are running a vlc-monitor 4.6 script, VLC instance is relaunched. Secondly, an autonomous mode page, where the user can order active display sets into an autonomous sequence which is set to play on system boot-up. Thirdly, the page for displaySet creation which is further divided into displaySet, dispalyTrack, videoFile and videoFolder modules. The structure of these modules is similar. The upper part of the page lists items of selected type with detailed attributes and actions while the bottom part contains a form for creation or upload regarding the videofile.

For more-detailed screenshots, please see an appendix D and figures D.1, D.2 and D.3 .

Modular javascript

The way we structure our javascript code is using so-called javascript modules which is a tool for sharing functions and variables across the multiple javascript documents using export and import statements [38]. This tool is innovative but often overlooked even though it brings modularity we know from more directive languages such as Java, C# into a JavaScript. Allowing us not to duplicate code and offers better maintainability.



Main

Hello! Welcome to admin console for this interactive projection!

Client	Ip	Port	Connected	Is playing
Client 1	127.0.0.1	5041	false	false
Client 2	192.168.0.106	5081	false	false
Client 3	192.168.0.113	5091	false	false
Client 4	192.168.0.105	5041	false	false

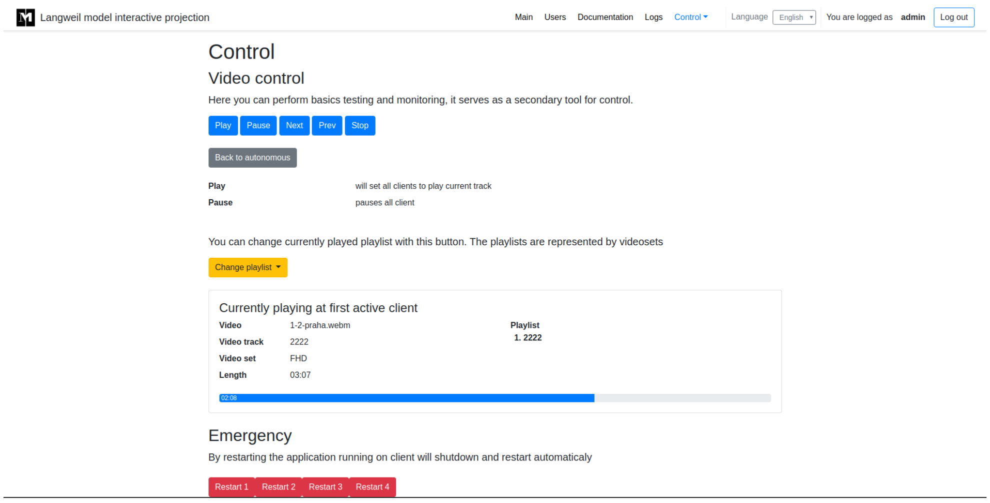
Currently playing at first active client

Video	1-2-praha.webm	Playlist
Video track	2222	1. 2222
Video set	FHD	
Length	03:07	

00:21

This system was created by Ondrej Trojan

Figure 4.2: Main page



Control

Video control

Here you can perform basics testing and monitoring, it serves as a secondary tool for control.

Play	will set all clients to play current track
Pause	pauses all client

You can change currently played playlist with this button. The playlists are represented by videosets

Currently playing at first active client

Video	1-2-praha.webm	Playlist
Video track	2222	1. 2222
Video set	FHD	
Length	03:07	

02:08

Emergency

By restarting the application running on client will shutdown and restart automaticaly

Figure 4.3: Control page

4.5 VLC client

We use VLC player in the version of 3.0.10, which is run in the remote control interface [39]. This interface is operated through textual commands typed into the invoking terminal window. Moreover, this interface is accessible through a network with a telnet-client by using the `--rc-host localhost:port`. In short, the full run command can be observed below.

```
vlc "$HOME/vid/loop.mp4" --loop --config="$HOME/.config/vlc/vlcrc"
    --fullscreen --video-on-top --no-video-title-show --no-sub-
    autodetect-file --file-logging --logfile "$HOME/vlc.log" -I rc
    --rc-host "$IP:$PORT" & echo $! >> "$HOME/vlc.pid"
```

The remote interface offers many commands such as the list below, which is taken from the interface help content.

```
add XYZ . . . . . . . . . . . . . . . . . . add XYZ to playlist
enqueue XYZ . . . . . . . . . . . . . . . queue XYZ to playlist
playlist . . . . . . . . . . . . . show items currently in playlist
play . . . . . . . . . . . . . . . . . . . . . . . . . play stream
stop . . . . . . . . . . . . . . . . . . . . . . . . . stop stream
status . . . . . . . . . . . . . . . . . . . current playlist status
```

The VLC player first plays the track loop.mp4 as can be seen in a run command above. The loop video serves as a visualized indicator of the client's activity, and after all clients connect to the server, the playlist is changed to autonomous mode.

4.5.1 Connection maintenance

The running VLC is watched by a `vlc-monitor.sh` while the server-client connection is kept with a periodic request for status. If received 1 or 0, VLC is running if received error or nothing, the sequence of stating the connection as inactive is initialized. In short, if the sequence reaches timeout, the connection is closed, and the process of its establishment is rerun.

4.5.2 Synchronization

Each device runs a VLC instance separately, and as of this stage of development, there is no comprehensive synchronization mechanism. Based on the evaluation is section 5.2.1 we suggest modified approach which is elaborated in future work 5.7 and chapter Conclusion 6.

4.6 Device configuration

To create a minimal setup for VLC player playback we installed below set of tools on top of clean Debian 10 installation.

```
sudo apt-get install -y xorg lightdm openbox dbus-x11 vlc openssh-
    server
```

Moreover, we copied vlc-monitor script together with the service configuration file and an initial loop video file. These items are installed using install script which configures systemd service, creates video and playlist directories, sets xrandr and lastly configures an autologin for this system. After the reboot, system runs the instance. Please see the repository [40] for updated source code.

As there is a requirement **P4** for playback of ultrawide video on one device across two displays, we leverage xrandr tool and created a virtual display merging the two physical ones using the command

```
xrandr --setmonitor virt auto DP-1,DP-2
```

vlc-monitor.sh. To keep the VLC instance running in case of an unexpected crash, we wrote a script `vlc-monitor.sh` which on execution verifies running openbox, get the IP address for a network interface and runs a VLC instance with the command from section 4.5. Most importantly, when running the script, periodically checks whether the VLC is running and if not, it reruns it. For that, we use PID file and `pgrep -F` command.

Systemd

Systemd is a suite of basic building blocks for a Linux system. It provides a system and service manager that runs as PID 1 and starts the rest of the system. [41] We can then configure additional services, which are then run in a reliable manner during system bootup.

vlc-monitor.service. Is a configuration file written as a service unit configuration which launches `vlc-monitor.sh` after initialization of network service, graphical target, location of XAuthority cookie and setting environment variable of `DISPLAY=:0`. Furthermore, we set a restart interval if vlc-monitor is stopped for some reason.

4.6.1 Intel NUC

Is a mini-PC based around Intel Core i3-7100U processor. Appropriate as a multimedia centre or in our case as a video playback device. It is powered with 12-24 VDC and has TDP of 15W. It is to be equipped with M.2 SSD disk and two slots for DDR4 RAM. Despite being provided with an integrated graphic card Intel HD Graphics 620, it is capable of being connected to two monitors and supports a 4K playback.

4.6.2 NVidia Jetson Nano

We have also brought this development board for testing. It is a development board made by Nvidia, has an integrated GPU with much power for graphical or AI-related tasks. However, we ran into considerable trouble while setting up the board. Firstly, the newest Linux image for the board showed signs of a severe slowdown [3]. Only after downgrading to 3 months old image, the problem was resolved. Secondly, we were unable to run a VLC in this device as the default system image is incompatible with VLC binary for this

[3] Disabling common use

architecture arm64/aarch64 provided in the ubuntu repository. Most likely, this issue will be solved in upcoming releases, but for now, we are unable to include this device into our tests. [42]

4.7 Development environments

During development, we used a set of tools and programs. This section mentions the significant ones.

4.7.1 IntelliJ IDEA

Is a development IDE developed by a Prague based company JetBrains and offers a superior functionality regarding software development for the majority of languages. It provides an excellent integration with prominent frameworks and offers plugins for productivity enhancement. It is one of the most popular IDEs for Java development, and based on this fact; we have developed a full server-side backend using it. It was an easy transformation into a mobile app development tool as the Android Studio[4] is based on a community version of IntelliJ. Just by including Flutter plugins and configuration AdroidSDK we were able to deploy, debug and advance in the development process.

4.8 Development tools

Fluido - Web-based mockup tool for prototyping and wireframe design

Docker - We have dockerized whole back-end application see 4.3 and used docker to run our database

VirtualBox - for running a virtualized Linux systems

4.9 Problems

During the development and operation, we have encountered a several problems either in terms of design, misconfiguration or bugs in software and tools we used. The below-listed points shall present the reader with a comprehensive knowledge.

- Video files have encoded title and length saved in their metadata. When running VLC playback, the query for playlist returns metadata titles and ignores the real file name.
- When adding image into VLC playlist, its length is set to 10 seconds. Ignoring duration in the playlist. Can be solved by adding extra option `<vlc:option>image-duration=600</vlc:option>` .

[4]In which Flutter development is often done

- The notable bug which reduced the ability to work with VLC player trough remote control interface is the fact of unexpected freezing while opened in fullscreen mode. It is a severe problem which limits the production use of this tool. For more information please see [43].

Chapter 5

Evaluation

This chapter describes the process of evaluation, which is initialized with a requirement verification 5.1, followed by testing and measurements. Lastly, we have carried out a usability testing in the section 5.6 which outcome is described in the following observations and next steps stated in section 5.7. It is worth mentioning that the reason for not including more people into usability testing process is due to the fact of COVID-19 pandemic and wide emergency-state, which made it impossible to study on more people.

5.1 Requirement verification

The requirements raised in the section [3.3] are to be validated and verified to check that application fulfils its intended purpose. To briefly point-out the work objectives, the task was to design and implement a control system for scalable multi-projection exhibition for an arbitrary number of displays.

To reliably check the requirements, we will use a system of the verification matrix with four following validation methods:

- Verification by test **(T)** - Verification by test is performed by subjecting the application to a physical test
- Verification by inspection **(I)** - Verification by inspection is performed by simply inspecting/looking at the application.
- Verification by analysis or similarity **(A)**. Verification by analysis is performed by a simulation on some parts of the application. Verification by similarity is performed by stating that a part of the application is similar to a part that has already marked as tested
- Verification by review-of-design **(R)**. Verification by review-of-design uses documentations to show that the application or its component will perform as expected.

Id	Requirement	Verification	Status	Test No.
F1	The system shall be able to display multiple files on multiple screens at a given time	T, I, R	Done	Test 1
F2	The system shall have administrator console for maintenance and fine-tuning	T, I, R	Done	Test 3
F3	The system shall be complemented by a mobile application for control	T, I	Done	Test 2
F4	The system shall have basic control request such as play/pause/next/prev	T, I	Done	Test 1, Test 2
F5	The admin console shall have the functionality of deliberate display set formulation	T, I	Done	Test 3
F6	Both admin console and mobile application shall have authorization and authentication mechanism	I, A, R	Done	
F7	The application shall have an autonomous mode which would play displaySet in loop	T, I	Done	Test 4
F8	The screens control shall be distributed to several client machines connected through network	I, A	Done	
P1	The system synchronization shall be limited sufficiently, so it is not noticeable in a real-time	T, I	Done	Test 1
P2	The server shall manage communication with a mobile application and web frontend	I, A	Done	
P3	The server shall manage videofile upload and playlist configuration	I, A	Done	
P4	The distributed clients machine shall display image to one or n monitors	T, I	Done	Test 1, Test 2
P5	The distributed client shall run on the same HW machine as server back-end	I, A	Done	
P6	The mobile application shall be targeted to Android platform	T, I	Done	Test 2
P7	The system shall operate within LAN network	I, A	Done	
P8	The mobile application shall control exhibition with commands (play, pause, next, previous, go-to item)	T, I, A	Done	Test 2
D1	The application shall implement the Server-Client architecture	I, A	Done	

D2	The application shall be complemented with a REST web-service architecture	I, A	Done	
D3	The application shall be equipped with an automated system of connection establishment and management to the client playback instances	T, I, A	Done	Test 5
D4	The back-end shall have scheduled job to query the current state of playback	I, A	Done	
D5	The back-end shall have scheduled job to query whether a client is connected and responding	I, A	Done	
D6	The back-end shall have scheduled job to query current clients playlist	I, A	Done	
D7	The playback instance shall be in the form of network-connected device running VLC player in the remote control mode	I	Done	
D8	The application shall be run on Android 9.0 or newer	I, R	Done	
D9	The back-end application shall be developed in Spring framework using Java language and Thymeleaf template engine	I, A, R	Done	
D10	The mobile application shall be developed using Flutter framework using Dart programming language	I, R	Done	
D11	The application shall follow Spring framework development patterns such as MVC, and Service - DAO architecture	I, A	Done	
D12	The back-end system shall be containerizable and complemented with docker-compose.yml	I	Done	
O1	The application shall be equipped with the latest software	T, R, A		Test 1, Test 2
O2	The mobile devices shall be connected to the network provided by the exhibition	T, I	Done	Test 2, Test 1
O3	The videofiles shall be compliant with the guidelines stated in the pertinent section	A	Done	
O4	Any user UI input shall be sanitized and validated	T, I, A	Done	Test 3
O5	The user text input shall be controlled for non-standard characters	T, I, A	Done	Test 3

Table 5.1: Verification matrix

5.1.1 Requirements tests

Requirements-based testing is a testing approach in which the tests are determined from stated requirements 3.3. They often address requirements such as functional, performance and design or issues of performance, reliability or usability. This part presents five tests validating all requirements marked as verifiable by tes **(T)** from the verification matrix. [5.1]

Name	Playback on 4 devices
Test procedure	We have a configured setup with distributed FHD and 4K test video. The test is initialized with FHD track. 1. Four devices with installed VLC instance, back-end with query single node option on (we are querying a current state of playback on only one node) and mobile device are connected into LAN network 2. Video camera records the content of 4 display devices 3. Prepared video set with periodic colour change (for synchronization testing) will be chosen on the mobile App 4. The playback run in a loop and is left attended running with no intervention for 10 minutes 5. Measure the display delays with respect to the monitor number 1 Repeat this test with back-end with query single node option of (we are querying a current state of playback all connected nodes) and with not querying nodes at all. Repeat with 4K video and evaluate test from video recording and observation. Repeat on three different devices other than Intel NUC.
Verification of	F1, F4, P1, P4, O1, O2,
Test completed	Done
Results	All tasks have been performed and proved the functionality of the system. We have also made a measurements elaborated in the section 5.2.1 which showed a certain level of desynchronization.

Table 5.2: Test 1

Name	Actions: play, pause, next, stop, move to using mobile application
Test procedure	We have prepared configured set of devices as a prerequisite. 1. We start the mobile application and wait for the connection establishment in the splash screen 2. We list available display sets and start the test set (FHD) 3. The play screen is shown and currently played track on client devices is stopped. Operator press play and we observe the video on screens 4. Operator press pause and observe video 5. Operator next and prev pause and observe video 6. Operator jumps to another track by pressing it in the app and observe video 7. Operator press stop and observe video
Verification of	F3, F4, P4, P6, P8, O1, O2
Test completed	Done (Successful)
Results	Tests have shown that we are able to send desired commands from the mobile app to the server and control the exhibition playback.

Table 5.3: Test 2

Name	Process of Display set creation and synchronization
Test procedure	1. Being logged in as admin in administrator console website open Control -> Edit video set tab 2. Operator creates a display set with an arbitrary name 3. Operator uploads video file either as video, picture or audio 4. Creates display track for previously created display set and uploaded video file 5. Publish created display set and run synchronization 6. Observe ability to run created display set
Verification of	F2, F5, O4, O5
Test completed	Done (Successful)
Results	All tasks have been completed and verified. We are able to create and distribute video content according to our desires to all clients.

Table 5.4: Test 3

Name	Autonomous mode
Test procedure	1. Client devices are turned on, we wait for init loop, and as all are connected we observe autonomous playlist execution 2. Then using a connected mobile app, we start an arbitrary playlist and wait for start and some progress 3. We stop current playlist by pressing back arrow and return to the homepage 4. There we press return to autonomous mode and observe the result.
Verification of	F7
Test completed	Done (Successful)
Results	The test has been performed with a satisfactory outcome. We verified that system boots to the autonomous mode and returned when instructed to

Table 5.5: Test 4

Name	Connection lost and establishment
Test procedure	1. Verify that all devices are running a displaySet have synchronized content and are shown in administrator console under the Main tab as connected 2. The operator will turn off the client (Intel NUC) device and wait for proper shutdown 3. The operator verifies that the device is shown as not connected in the administrator console 4. The device is started again; operator waits for booting and VLC instance startup 5. We see the unsynchronized video on the restarted node, but we verify that the client is shown as connected in the administrator console 6. Lastly we set a video set to play and observe synchronization recovery
Verification of	D3
Test completed	Done (Successful)
Results	All tasks have been completed and verify that the system is able to initialize, maintain and restore the lost connection

Table 5.6: Test 5

5.2 Performed tests

Test equipment. For our testing, we used a list of devices below. Furthermore, we have captured the tests on a mobile phone camera in the FHD resolution.

- 3x Intel NUC 256GB M.2 SSD, 4GB RAM, clean Debian 10
- 1x Intel NUC 512GB M.2 SSD, 4GB RAM, clean Debian 10
- 1x Intel lab computer - Intel Xeon Processor E3-1231 v3@ 3.4Ghz, 16GB RAM, running Ubuntu 18.04 from flash drive
- 1x notebook Thinkpad T460 - Intel Core i5-6200U, 8GB RAM, running Linux Mint 19.3,
- Mobile phone Samsung Galaxy S10s

Test setup. The setup followed suggested design patters as can be seen in fig. 3.3 according to specifications. In specific test cases, we followed the description in the test table.

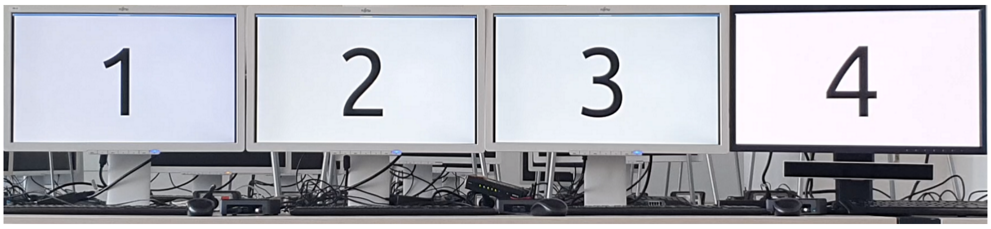


Figure 5.1: Monitors denomination

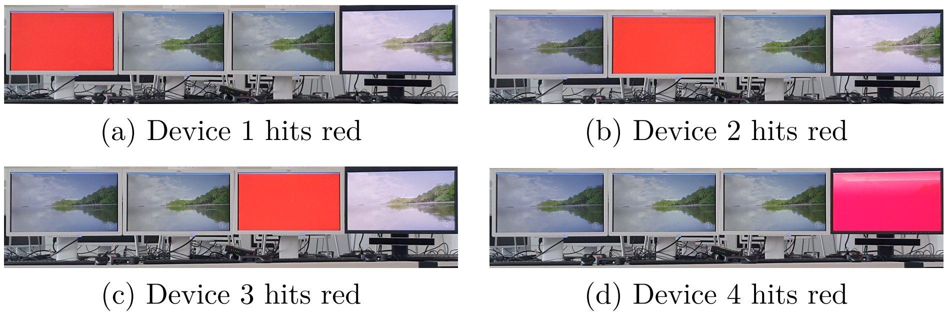


Figure 5.2: Mobile app UI design

5.2.1 Measurements

This section describes the measured outputs from selected tests above and the additional test which followed our hypothesis or gave a better understanding of the system behaviour. Unless stated otherwise, the figures plots device's delay with respect to the device one (the very left). Moreover, we have been capturing on an FHD Camera at 30 FPS rate. Please note that thanks to the bug **25** tested playback was using VLC in windowed mode as the fullscreen caused the freezing. Given that, and the duration of the measured red screen being three frames we are left with the uncertainty of 100ms (50 ± 100) ms

1. Test description can be seen in the table 5.2 - During the development, we have noticed a slight inconsistency in terms of synchronization of the connected nodes. That could have been caused by the fact that all nodes were virtualized [1]. After migration to the HW device, synchronization improved drastically but remained an issue. Following tests shall accurately measure time delays of the system. Out of this reason, we have set up 3 configurations to test the system.

   The test procedure was as follows: we launched FHD playlist on the connected and configured net of devices, we restarted VLC instances (clear playlists) and sent the command to play the FHD displaySet. Speaking of the first option (querying all devices), runtime varies but on all occasions reached 10 minutes. In figure 5.3, we can see that the delay varies for each device. However, in the 10th minute, the device 4 reached (1250 ± 100) ms delay, and the device 2 (750 ± 100) ms. It is

[1]In Virtualbox

worth noting that the devices 1 and 3 had a minimal delay very close to the measurement uncertainty. Regarding the not querying options, see fig 5.4, trends are similar. We observe devices 2 and 4 delaying to (1000 ± 100) ms and (1200 ± 100) ms respectively. Regarding the single querying, options see fig 5.5 we observe an even steeper delay of (1250 ± 100) ms for both device 4 and 2, but we cannot declare it to be caused by the query options. However, it is a possible explanation as the queried device was device number 2. As usual, devices, 1 and 3 are within uncertainty and keep adequately synchronized.

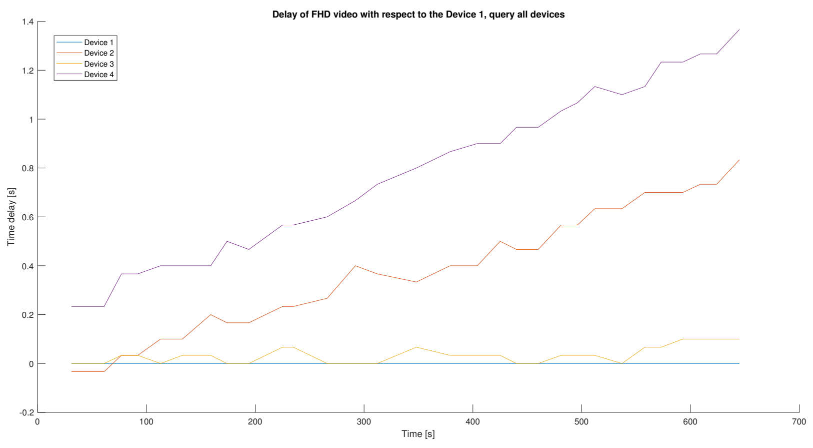


Figure 5.3: FHD - query all devices

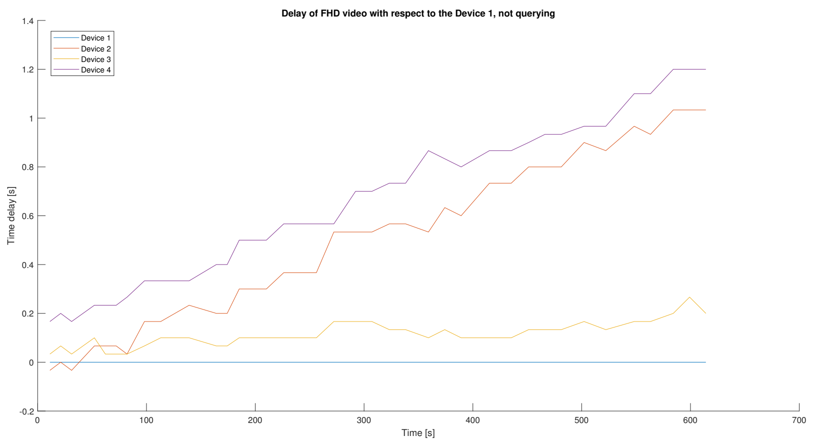


Figure 5.4: FHD - not querying

During the test, we have summarized observation that devices 1 and 3 are staying relatively synchronized. We have investigated this more thoroughly and discovered that during the setup, all other devices 3,4 had installed a PulseAudio drivers while the former set was not.

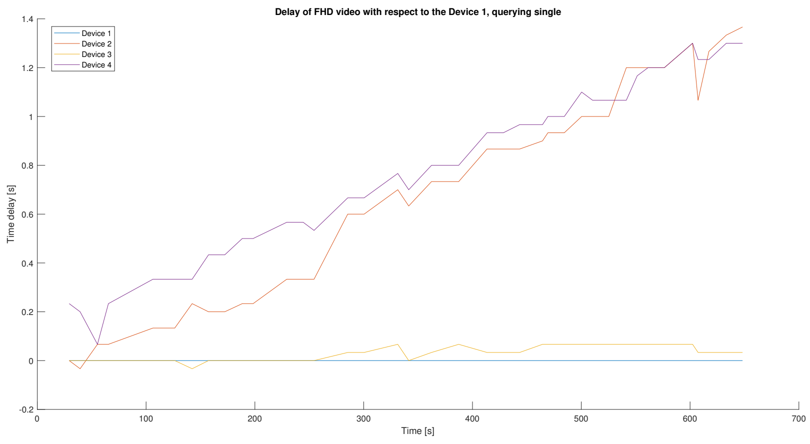


Figure 5.5: FHD - query single

The hypothesis is that the PulseAudio creates additional workload for the device and a cause delay.

a. 4K - This test is based on the Test 1, with the difference in resolution, FHD is increased to 4K resolution. Observation is that all devices are delaying to each other in relatively high order. With respect to the device 1, the delay reach (1000 ± 100) ms, (750 ± 100) ms, (2000 ± 100) ms for devices 2, 3 and 4 respectively. Such delay is not acceptable and is very noticeable.

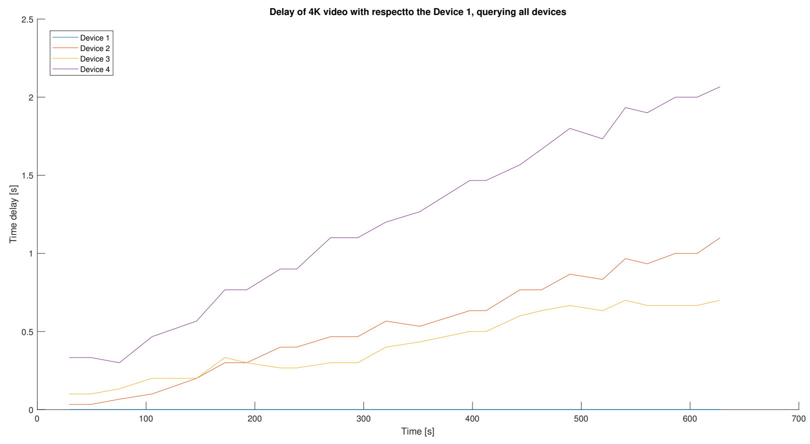


Figure 5.6: 4K - query all

b. In this test, the intention was to measure behaviour on the devices other than Intel NUC box (Device 1). As the NVidia Jetson failed to deliver VLC playback (see 4.6.2), we included a Test lab PC (Device 2) and a Development laptop (Device 3) into the testing process. They are both running Linux system each with a different distribution, HW and setup in general (detailed in the section Test setup above). We also run a VLC instance at the same device as the server. Consequently, by succeeding, we proved the requirement (P6) about this issue. We have observed that the measured delay difference is diverging from the referenced one. Please see the figure 5.7. The device 1 has reached a delay of (300 ± 100) ms while device 3 preceded the reference by (2100 ± 100) ms on a 10-minute measuring. That gives us a base understanding of how much the playback is dependent on the platform and installed software.

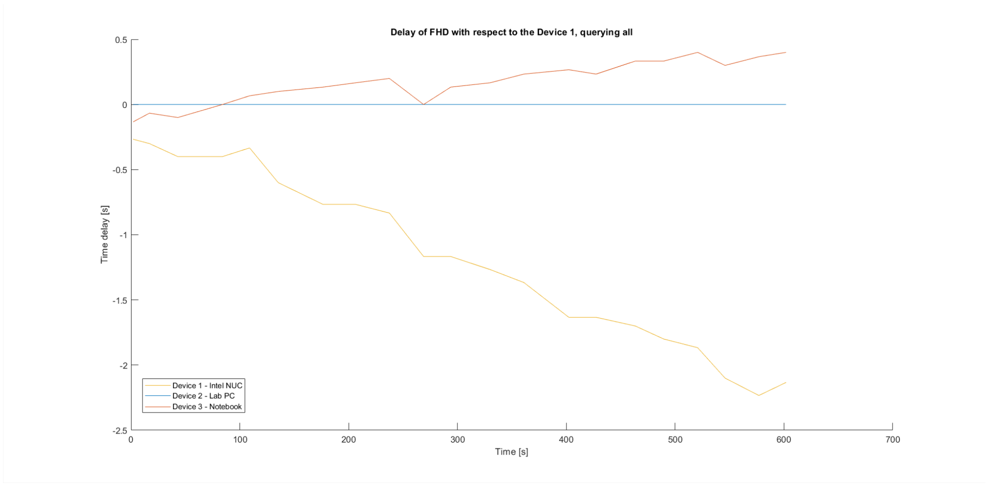


Figure 5.7: FHD - Notebook, PC, Intel NUC

c. Based on the hypothesis 1 above, we have introduced a test where we have put a new clean image of Debian on device number 3. Then we repeat a previous test[2]. We observed that the pulse audio software caused the delay on a device 3. Because the delay of the inspected device three has adapted to device 1 and 2 after the correction. Again, device 4 has reached delay of (1400 ± 100) ms and is significantly and noticeably our of synchronization.

[2] Only with query all configuration

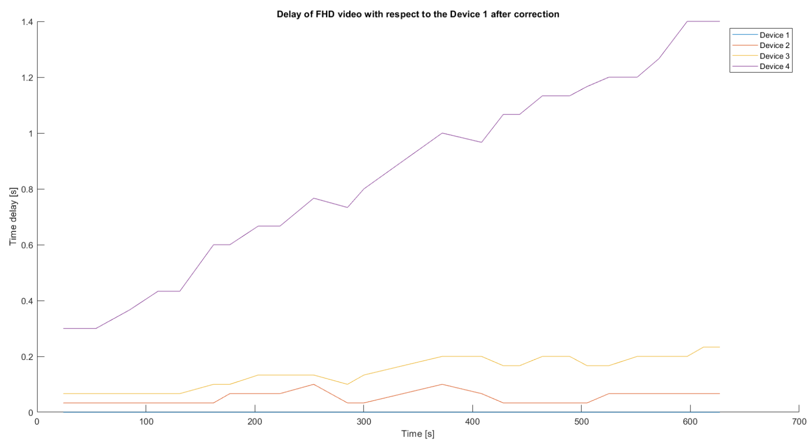


Figure 5.8: FHD - after correction

2. What we are not sure, however, whether PulseAudio delays the video playback or devices without it precede the playback. To measure the playback on devices with respect to estimated video time, we have performed a new test. We have measured for a longer period of time, and plotted the difference to figure 5.9 below. Also, we avoided using windowed mode and triggered playback on regular VLC using the parallel ssh command line. It is to be said that we have recorded this video in 60FPS format. Therefore, we get uncertainty of 50ms. On a 21 minutes measurement we measured the delay of (1500 ± 50) ms for device 4 while devices 1, 2 and 3 precede the reference by (500 ± 50) ms , (400 ± 50) ms , (370 ± 50) ms.

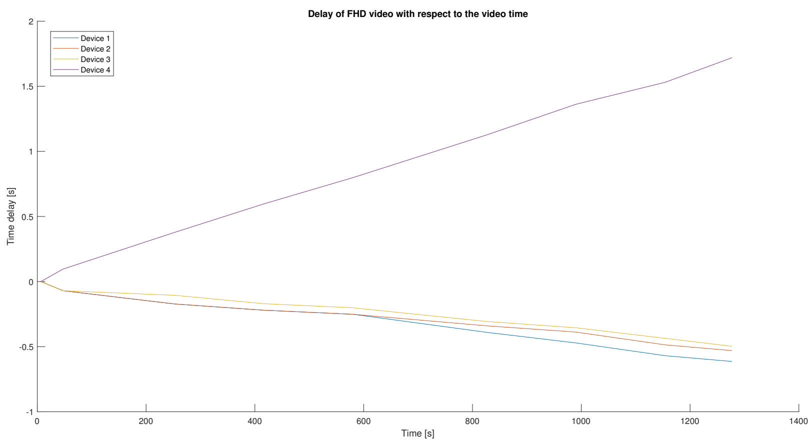


Figure 5.9: FHD - with respect to video time

3. Lastly, we have measured the average response using the remote control interface. In the virtualized client device the response varied from 200-250ms for a response with text content (for example for query what track

is playing was returned 1-1-track.mp4). For a non-response command (such as play or pause) 3 - 10ms. After migration to the HW device, the response improved to 42-47ms for text response and 5-8ms without a response. It is a potential bottleneck of the application.

5.3 Unit tests

Unit tests are executed to test the functionality of the specific components/methods of a complex system. The data used during such testing are prepared prior to the execution as static objects or mocked using various mock tools. Based on the known data, each component has expected behaviour and output, which is consequently compared to the real output. In this project, we have implemented a small set of unit test verifying only the core functionality of the system. More unit test would be appropriate, and their development is scheduled.

5.4 Integration tests

In contrast to the Unit test, integration tests are testing cooperation of a different system components and functionality in a high-level point of view. Commonly the data are not mocked but connected to the real non-productional data queried from the database. In our system, we have a few simple integration tests, and given the scale of the system, more sophisticated tests would be reasonable. This should be addressed in future work.

5.5 System tests

Are testing a complete integrated system. We can declare tests 5.2, 5.3, 5.4, 5.5, 5.6 to be system tests as they are run on a implemented, integrated application and are validating the requirements 3.3 stated in Design chapter 3.

5.6 Usability test

This section will focus on target-group testing regarding applications usability, handling and operation. The notable conditions are the control and user interface inspection. For that reason, three scenarios are introduced which shall evaluate objectives raised in section 3.1.

5.6.1 Initial survey

Question	Answer
Education	Humanities
Do you use a smartphone, if so to what extent?	Does not use smartphones, avoids touchscreen devices and is uncertain in their operation
What is the experience with multimedia applications?	Basic operation of other exhibitions
What is your expectation from the application testing	Adventure, testing the functionality of the controls and the first experience with a sophisticated type of control. As a leader, I expect that it will be ready for the user to learn how to use it easily and for myself that I learn it well so I can advise the lecturer if they are confused.

Table 5.7: Participants initial survey

5.6.2 Test plan

The below-listed scenario contains the literal task which the usability participant has received. The test was divided into two parts. Firstly, the participant was testing the web administration, where he uploaded videos, created and configured playable displaySet. Secondly, the participant used the mobile app for the exhibition control, which consisted of playing the created displaySet.

Web page test plan

This test is examining the administrator console and the process of uploading, creation of playable sets and its maintenance.

1. Log in to the web portal
2. Verify that the clients are connected
3. Click through the web application, get acquainted
4. Create a video set with a specific name
5. Create a video folder with any name
6. Upload a video file from `otrojan/Videos/ForTesting/FHD/3constarica.webm` to this folder as a video and rename it as you wish.
7. For the display set, create a display track with the uploaded video file
8. Publish and synchronize the display set

9. Run created playlist
10. Perform actions Pause, Play, Stop, Next track, Previous track, Jump to track of your choice
11. Return to autonomous mode

Mobile app test plan

This test is focused on the use of the mobile app and expected control scenario examine its core functionality.

1. Log in to the mobile application (username: admin, password, nimda)
2. Select the displaySet and run it
3. After the 1 minute run, skip to
4. Then stop playback and return to standby mode
5. Sign out

5.6.3 Test results

Comment on the testing process: The test was undertaken at a computer lab with a full setup. Participant showed enthusiasm and interest in the tested system. Firstly, we gave the participant an initial explanation of the testing process and a description of core concepts necessary for the correct system operation[3]. That followed by the web-console test execution. Answers, impressions and participants observations are stated in the table 5.8. There was an occurrence of a minor technical problem which was promptly fixed and test continued. After the successful creation of a playable display set, the participant started to test a Mobile application and results are stated in the same table.

5.6.4 Observation

The testing pointed out an abundant number of things. Generally, all tasks were completed successfully, and the system is functional. The significant objective observation is that the Web application has been hard to navigate, and the process of display set creation should be more intuitive. That has a lot to do with an overall complexity of this task and inexperience of the operator. What certainly played a role in the usability is the limited knowledge of English in which the vast majority of text and terminology was positioned. Nevertheless, all test task has been accomplished, and more noteworthy observations are listed below.

[3]Explain terminology of DisplaySet, DisplayTrack, VideoFile, Synchronization and Publish

Web results	
Question	Answer
What are your first impressions?	Web is confusing
Commentary on the process of playable display set creation?	Problems with Czech terminology, web in English, I am a non-technical individual
Did you experience any issues with test task fulfilment?	Insufficient knowledge of English, some meanings are illogical. Non-intuitive
Did the application behave as expected?	Not at first (technical problem), gradually start to orient in control and navigation. User-made several mistakes, in the end, it was easier to control
Mobile app results	
What are your first impressions?	Horror, uncertainty, purposefully avoids smartphones, graphically clean, app likes, is natural, intuitive
How difficult would you evaluate the app control?	It is intuitive, simple, offers itself what to do. Only troubles with returning to autonomous mode
Did you experience any issues with test task fulfilment?	No
Did the application behave as expected?	Yes
Final survey	
Observation to the user interface?	Not really, focused on the content,
What needs to be improved/changed?	Terminology for users, clarify the order of steps, create sequences, sort and then name it. For me, it is natural to create first and name, in the end, add the possibility of editing display sets

Table 5.8: After testing survey

- Process of displaySet creation was hard to find, should be made more intuitive
- User intuitively fills forms from above to bottom
- Lacking full support for Czech characters in form validation
- DisplayTrack configuration is hard to find
- Missing feature for displayTrack ordering
- Hard to find a synchronization button
- Adds more straightforward terminology such as (Select, Choose)

5.7 Future work

Based on the usability testing outcome, we state, that UI should be a priority to improve. Especially regarding the web console, which was difficult to navigate and perform the tasks. Mobile app yields better outcome but still has use-cases to improve. In terms of system design and functional perspective, we shall focus on improved testing in the form of more unit tests and integration tests. Furthermore, we shall address a problem of synchronization delay during playback as can be seen in section 5.2.1 by introducing a new video player solution, preferably with periodic synchronization clock and algorithms [44]. After broader research, we propose a candidate of MPlayer which is presented with synchronized playback function as it is stated in its documentation: *"Multiple instances of MPlayer can synchronize playback over a network. This is useful for creating "video walls" with multiple screens controlled by different computers. Each MPlayer instance can play a different video, but they all will try to stay at the same time offset in the file."*

Moreover, the process of synchronization action is time-demanding, and further optimization shall take place. Lastly, features such as searching by the video folders would be welcomed by the users.

5.8 Known bugs

- During filling the form for video file upload user fills its name before selecting the file from the computer, his selected name is changed to the name of the file.
- Assigned role (ADMIN, PRESENTER) endpoints are malfunctioning due to misconfiguration in the security configuration

Chapter 6
Conclusion

The focus of this work was to analyze, design and implement a system for multi-projection control. The developed system delivers the functionality described in objectives 3.1. Based on the requirements test 5.1.1 and usability test 5.6, we can state that system is fully operating, even though specific UX scenarios were not that straightforward and some minor problems were observed.

Lets briefly remind the reasons for VLC player, the intention was to avoid content streaming which would be challenging to scale for many clients. But instead, we distribute video files in advance and only control the playback through remote interface commands. Based on the first prototype layout visualization (see fig. 3.1), a visitor could see only one, maximally two displays simultaneously. That resulted in slightly relaxed synchronization requirements. However, as the layout was updated, this assumption is no longer valid.

A significant drawback in terms of reliable operation and maintenance is the synchronization results of our chosen video player. Reaching delay of (1400 ± 100) ms in 10-minute measurement makes this solution unsuitable for daily operation. In future work, it is necessary to address this issue which is limiting a production use of the system's current state. As it was suggested in section 5.7 to resolve the synchronization point, we shall introduce a new multimedia player. A promising candidate is MPlayer which has features for synchronized playback and delivers the same qualities in terms of video playback.

The used technologies which were evaluated out of the Analysis chapter 2 have proved suitable except for the VLC player. The Flutter framework is a straightforward, fast-development, professional tool for mobile app development. Moreover, Spring is a reliable and robust server-side back-end framework.

Finally, we shall state that as the project deadline is in 2022, further work will be undertaken and the majority of imperfections addressed. For now, the system has undoubtedly proved the feasibility of the design as well as usability by the operator.

References

[1] D. Michael-Grigoriou, N. Pelekanos, I. Chrysanthou, P. Zaharias, L. Hadjigavriel, and Y. Chrysanthou, “Comparative study of interactive systems in a museum”, Nov. 2010, pp. 250–261. DOI: `10.1007/978-3-642-16873-4_19`.

[2] (©2020). Langweilův model prahy, [Online]. Available: `http://www.muzeumprahy.cz/langweiluv-model-prahy/` (visited on 04/22/2020).

[3] Y. H. Choi, I. J. Seong, H. C. Kim, and S. R. Kang, *Multi-projection system*.

[4] Y. Nakamura and N. Hashimoto, “Easy multi-projection system over a whole indoor space”, *2018 International Workshop on Advanced Image Technology (IWAIT)*, pp. 1–4, 2018.

[5] (©2004-2020). Rendering to multiple displays with ndisplay, [Online]. Available: `https://docs.unrealengine.com/en-US/Engine/Rendering/nDisplay/index.html` (visited on 05/21/2020).

[6] (2015). Video synchronization, [Online]. Available: `https://vvvv.org/documentation/video-synchronization` (visited on 05/11/2020).

[7] (©2020). High performance media systems, [Online]. Available: `https://derivative.ca/feature/high-performance-media-systems` (visited on 05/21/2020).

[8] (©2020). Green hippo coretech, [Online]. Available: `https://www.green-hippo.com/about/green-hippo-core-tech/` (visited on 05/22/2020).

[9] (©2020). About disguise, [Online]. Available: `https://www.disguise.one/en/about/` (visited on 05/22/2020).

[10] (©2020). Displays, [Online]. Available: `https://7thsensedesign.com/displays/` (visited on 05/22/2020).

[11] c. ren and J. Xue, “Seamless multi-projector displays using nonlinear edge blending”, *Applied Mathematics*, vol. 09, pp. 764–778, Jan. 2018. DOI: `10.4236/am.2018.96053`.

[12] (2001-2020). Interactive kiosk, [Online]. Available: `https://en.wikipedia.org/wiki/Interactive_kiosk` (visited on 05/21/2020).

[13] Scalefusion. (2018). What is kiosk mode in android devices?, [Online]. Available: `https://medium.com/@Scalefusion/what-is-kiosk-mode-in-android-devices-f1018e7bef13` (visited on 04/22/2020).

[14] G. Breux. (©2010-2020). Client-side vs. server-side vs. pre-rendering for web apps, [Online]. Available: `https://www.toptal.com/front-end/client-side-vs-server-side-pre-rendering` (visited on 05/09/2020).

[15] E. Polat and S. Belkheraz, *ASP.NET Core MVC 2.0 cookbook: effective ways to build modern, interactive web applications with ASP.NET Core MVC 2.0*, English, First publish. Birmingham;Mumbai; Packt, 2018, ISBN: 9781785886751;1785886754;

[16] (2016). Mbosecke/template-benchmark, [Online]. Available: `https://github.com/mbosecke/template-benchmark` (visited on 05/09/2020).

[17] (2001-2020). Javaserver pages, [Online]. Available: `https://en.wikipedia.org/wiki/JavaServer_Pages` (visited on 05/21/2020).

[18] *Hybrid vs. Native, An introduction to cross-platform hybrid development for architecs and app development leaders*, 1st ed. Shorewood (WI): Drifty Co., 2020. [Online]. Available: `https://ionicframework.com/books/hybrid-vs-native` (visited on 05/21/2020).

[19] C. Griffith. (Feb. 2019). Client-side vs. server-side vs. pre-rendering for web apps, [Online]. Available: `https://ionicframework.com/resources/articles/what-is-hybrid-app-development` (visited on 05/09/2020).

[20] (2017). V154c1/libyuri, [Online]. Available: `https://github.com/v154c1/libyuri` (visited on 05/21/2020).

[21] A. Kuznetsova, "Distribuovaný přehrávač videa", Bakalářská práce, České vysoké učení technické v Praze, Zikova 1903/4, 166 36 Praha 6, Czech Republic, 2015.

[22] (©2000-2018). Synchronized playback over a network., [Online]. Available: `http://www.mplayerhq.hu/DOCS/HTML/en/networksync.html` (visited on 05/09/2020).

[23] D. Sedlacek, "Official project description", Praha, 2019.

[24] D. Boelens. (2018). Reactive programming - streams - bloc, [Online]. Available: `https://www.didierboelens.com/2018/08/reactive-programming-streams-bloc/` (visited on 04/24/2020).

[25] M. L. Napoli, *Beginning Flutter*, English, 1st ed. US: Wrox, 2019, ISBN: 1119550823;9781119550822;

[26] K. S. Prasad Reddy, *Beginning Spring Boot 2: Applications and Microservices with the Spring Framework*, English, 1st ed. Berkeley, CA: Apress L. P, 2017, ISBN: 1484229304;9781484229309;1484229312;9781484229316;

[27] C. Walls, *Spring Boot in Action*, 1st. USA: Manning Publications Co., 2016, ISBN: 1617292540.

[28] J. Zettler, “Ux design for mobile: Design apps that deliver impressive mobile experiences”, 2019.

[29] S. Newman, *Building microservices*, English, 1st. Sebastopol: O’Reilly, 2015, ISBN: 1491950358;9781491950357;

[30] R. Johnson, J. Hoeller, K. Donald, C. Sampaleanu, R. Harrop, T. Risberg, A. Arendsen, D. Davison, D. Kopylenko, M. Pollack, and T. Templier. (2020). Spring framework documentation, [Online]. Available: https://docs.spring.io/spring/docs/current/spring-framework-reference/index.html (visited on 05/01/2020).

[31] (©2020). Introduction to json web tokens, [Online]. Available: https://jwt.io/introduction (visited on 04/24/2020).

[32] (©2020). Flutter documentation, [Online]. Available: https://flutter.dev/docs.

[33] (February 12, 2020). The @scheduled annotation in spring, [Online]. Available: https://www.baeldung.com/spring-scheduled-tasks (visited on 2020).

[34] R. Borowiec. (©2020). Thymeleaf page layouts, [Online]. Available: https://www.thymeleaf.org/doc/articles/layouts.html (visited on 05/15/2020).

[35] O. Trojan. (2020). 5d projection app, [Online]. Available: https://gitlab.fel.cvut.cz/trojaond/5d-projection.git (visited on 05/21/2020).

[36] R. C. Martin, J. W. Grenning, S. Brown, K. Henney, and J. Gorman, *Clean architecture: a craftsman’s guide to software structure and design*, English. New York;Munich;Delhi;Singapore;Hong Kong;Cape Town;Taipei;Dubai;San Francisco;Milan;Amsterdam;Seoul;Tokyo;Madrid;London;São Paulo;Columbus;Montreal;Paris;Toronto;Mexico City;Sydney;Boston;Indianapolis; Prentice Hall, 2018, ISBN: 0134494164;9780134494166;

[37] O. Trojan. (2020). 5d projection backend, [Online]. Available: https://gitlab.fel.cvut.cz/trojaond/5d-backend (visited on 05/21/2020).

[38] (©2005-2020). Javascript modules, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules (visited on 05/17/2020).

[39] (2019). Documentation:modules/rc, [Online]. Available: https://wiki.videolan.org/documentation:modules/rc/ (visited on 05/17/2020).

[40] O. Trojan. (2020). Device configuration, [Online]. Available: https://gitlab.fel.cvut.cz/trojaond/5d (visited on 05/21/2020).

[41] (©2020). Systemd, [Online]. Available: https://systemd.io/ (visited on 05/17/2020).

[42] (2019). Vlc doesn’t work on nvidia jetson nano, [Online]. Available: https://forums.developer.nvidia.com/t/vlc-doesnt-work-on-nvidia-jetson-nano/77353 (visited on 05/19/2020).

[43] (2017). Computer freezes in full-screen vlc video and other times., [Online]. Available: `https://www.linuxquestions.org/questions/debian-26/computer-freezes-in-full-screen-vlc-video-and-other-times-4175617688/` (visited on 05/19/2020).

[44] S. Nam, S. Deshpande, V. Vishwanath, B. Jeong, L. Renambot, and J. Leigh, "Multi-application inter-tile synchronization on ultra-high-resolution display walls", Jan. 2010, pp. 145–156. DOI: `10.1145/1730836.1730854`.

Appendix A

List of Shortcuts

Shortcut	Meaning
UI	User Interface
UX	User Experience
SDK	Software Development Kit
API	Application Programming Interface
MVC	Model View Controller
DAO	Data Access Object
DTO	Data Transfer Object
REST	Representational State Transfer
IOS	Inversion Of Control
DI	Dependency Injection
OOP	Object Oriented Programing
AOP	Aspect Oriented Programing
IT	Information Technologies
SSH	Secure Shell
HTML	Hyper Text Markup Lanugage
CSS	Cascading Style Sheets
JSON	JavaScript Object Notation
YAML	a recursive acronym for "YAML Ain't Markup Language"
SPA	Single-page application
ACID	Atomicity, Consistency, Isolation and Durability
OOTB	Out Of The Box
UML	Unified Modeling Language
IDE	Integrated Development Environment
TDP	Thermal Design Power

Appendix B

Content of attached CD

Folder	**Content**
5d/	Device configuration project
5d-backend/	Server-backend and website project
5d-projection/	Mobile app source code
5d-postman-collection.json	API endpoints collection
docker-compose.yml	Configuration for a docker server-backend
README.md	Guidelines and manual for projects

Appendix C
Manual

This manual is a copy of a `README.md` file included to the thesis's attachment.

C.1 1. Device configuration

Located in directory **5d** Linux systemd service configuration, together with script maintainting active VLC Player instance. Conf

C.1.1 Getting started

Follow README in the 5d project, in clean Debian 10 install run `bash sudo apt-get install -y xorg lightdm openbox dbus-x11 vlc xrandr openssh-server` Copy files from folder *5d* to the device. Then configure vlc-monitor.sh field ON DEPLOY CHANGE and optionaly vlc-monitor.service if username is different from name "user". Then run `bash ./install user`

C.2 2. Server backend

Located in directory **5d-backend** This projects consists of Spring Boot backend and web-site for administrator management. The Mobile app requires running server for expected behaviour.

C.2.1 Getting started

See configuration file `./5d-backend/src/main/resources/application.yml` there you are advised to configure connected and active clients, together with passwords, ips and ports.

> **Note:** The configuration **.yaml** file consists of two profiles. As we run system in docker, we use **dock** configuration

Then build source using located maven runing `bash ./5d-backend/mvnw clean package -Dmaven.test.skip=true`

Then run located `./docker-compose.yaml` file with command `bash docker-compose up -build` The server starts in a docker container **5d-spring** together with database **5d-postgres**. After initialization, administrator website is accessible on *localhost:8080*. Currently tested browser is Google Chrome. Default credential are username: **admin** and password **nimda**

C.3 3. Mobile app

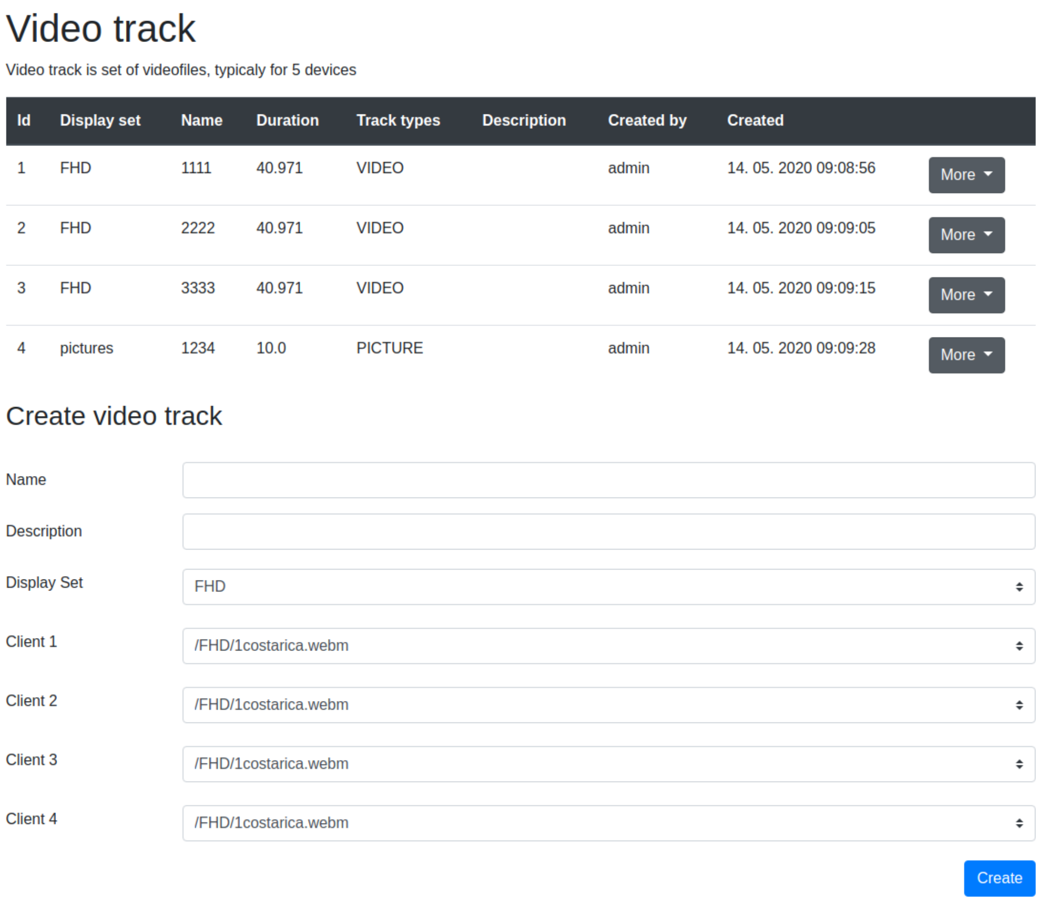
Is located in directory **projection**

> **Note:** Mobile app has hardcoded ip adress of the exhibition server. That is located in `./projection/lib/src/blocs/bloc.dart` in static field `serverRoot`. Migration to configuration style adress binding is scheduled, now if you want to connect to server, you need to change manualy

C.3.1 Getting started

Requirements for build is located Android SDK 28. Then open project in favourite IDE (Android Studio or IntelliJ), configure device and deploy on the mobile phone.

Appendix D
Screenshots



Video track

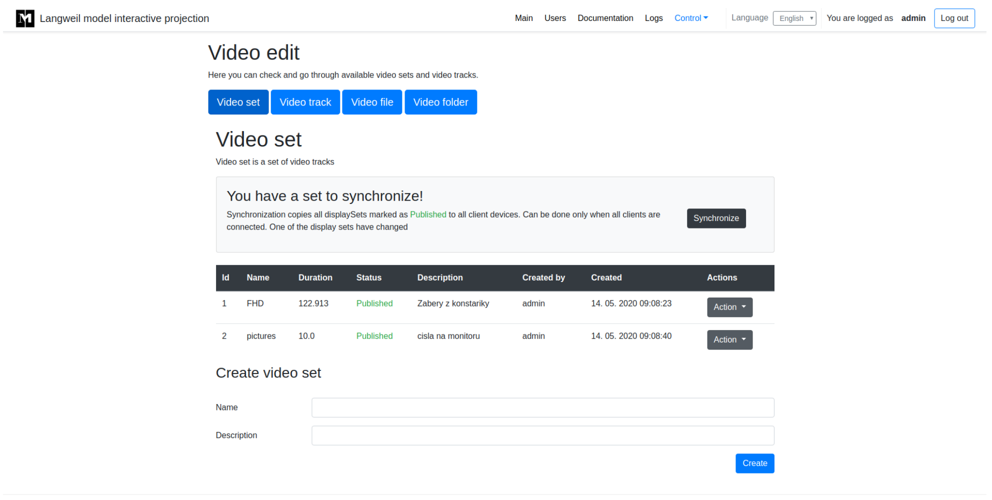
Video track is set of videofiles, typicaly for 5 devices

Id	Display set	Name	Duration	Track types	Description	Created by	Created	
1	FHD	1111	40.971	VIDEO		admin	14. 05. 2020 09:08:56	More
2	FHD	2222	40.971	VIDEO		admin	14. 05. 2020 09:09:05	More
3	FHD	3333	40.971	VIDEO		admin	14. 05. 2020 09:09:15	More
4	pictures	1234	10.0	PICTURE		admin	14. 05. 2020 09:09:28	More

Create video track

Name	
Description	
Display Set	FHD
Client 1	/FHD/1costarica.webm
Client 2	/FHD/1costarica.webm
Client 3	/FHD/1costarica.webm
Client 4	/FHD/1costarica.webm

Figure D.1: Web - Display track configuration



Video edit

Here you can check and go through available video sets and video tracks.

Video set

Video set is a set of video tracks

You have a set to synchronize!

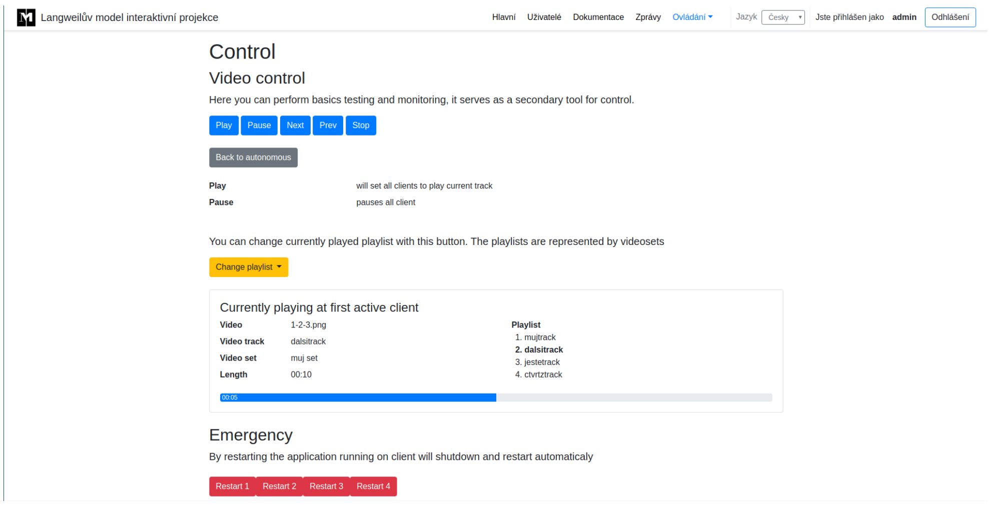
Synchronization copies all displaySets marked as Published to all client devices. Can be done only when all clients are connected. One of the display sets have changed

Id	Name	Duration	Status	Description	Created by	Created	Actions
1	FHD	122.913	Published	Zabery z konstariky	admin	14. 05. 2020 09:08:23	Action
2	pictures	10.0	Published	cisla na monitoru	admin	14. 05. 2020 09:08:40	Action

Create video set

Name	
Description	

Figure D.2: Web - Display sets configuration



Control

Video control

Here you can perform basics testing and monitoring, it serves as a secondary tool for control.

Play	will set all clients to play current track
Pause	pauses all client

You can change currently played playlist with this button. The playlists are represented by videosets

Currently playing at first active client

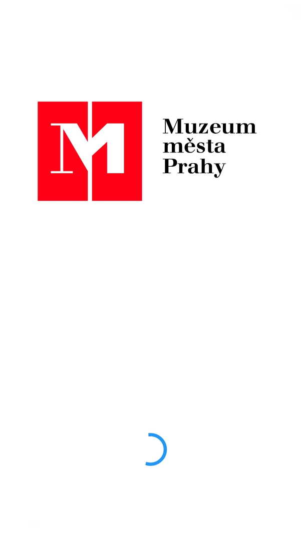
Video	1-2-3.png
Video track	dalsitrack
Video set	muj set
Length	00:10

Playlist
1. mujtrack
2. **dalsitrack**
3. jestetrack
4. ctvrtztrack

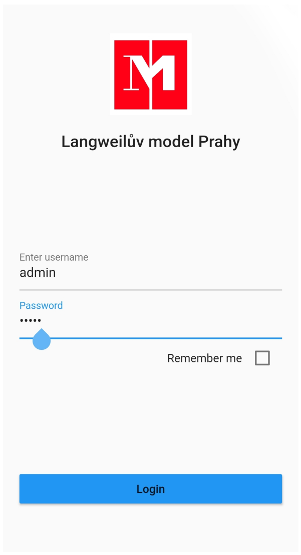
Emergency

By restarting the application running on client will shutdown and restart automaticaly

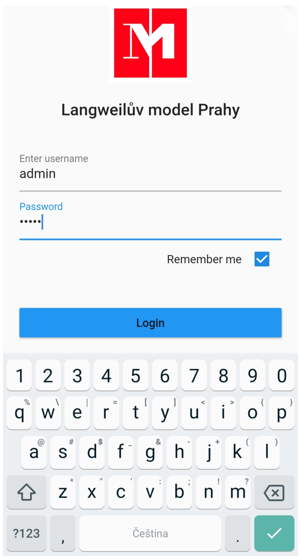
Figure D.3: Web - control screen



(a) Splash screen

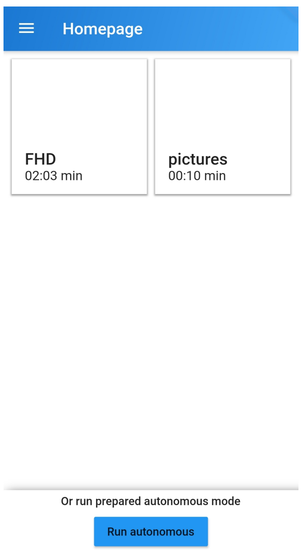


(b) Login

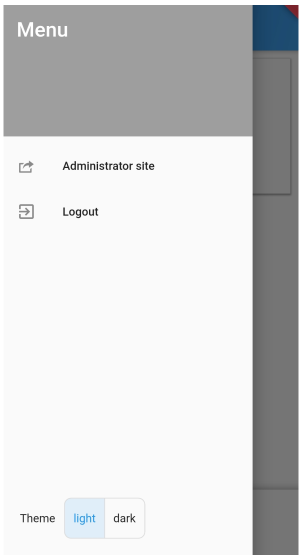


(c) Login with keyboard

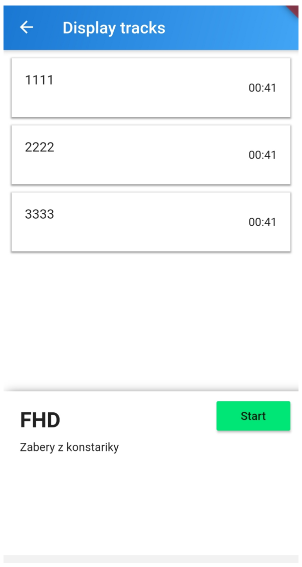
Figure D.4: Mobile app I



(a) Homepage light theme

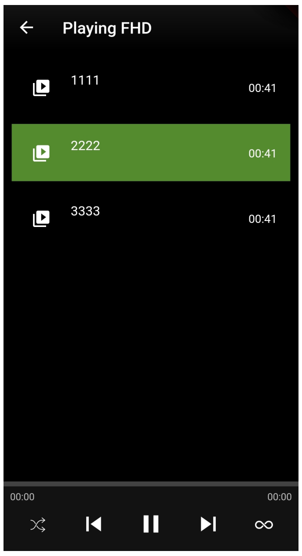


(b) Homepage sidebar

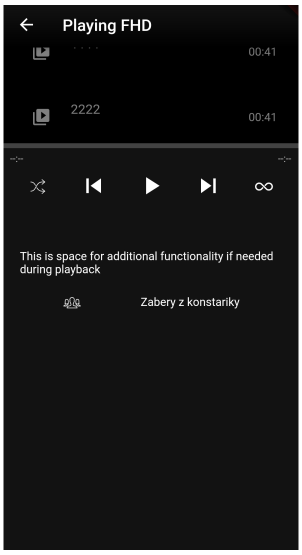


(c) Display set detail

Figure D.5: Mobile app II



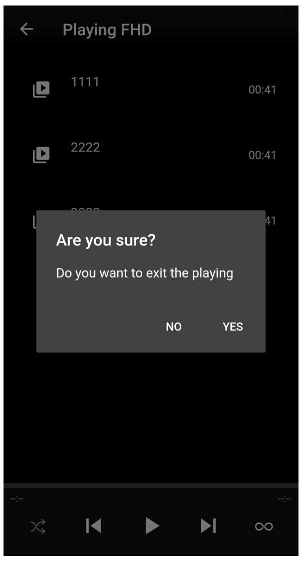
(a) Playback dark theme



This is space for additional functionality if needed during playback

Zabery z konstariky

(b) Playback below bar dark theme

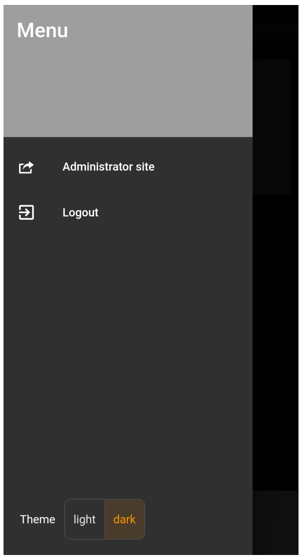


Are you sure?

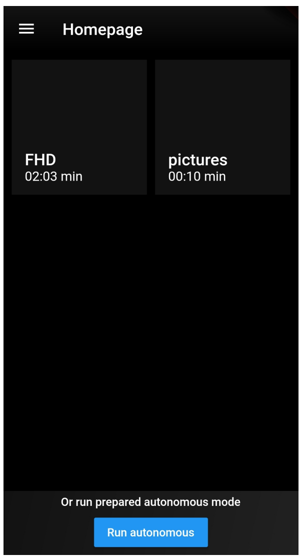
Do you want to exit the playing

(c) Popup on playback exit

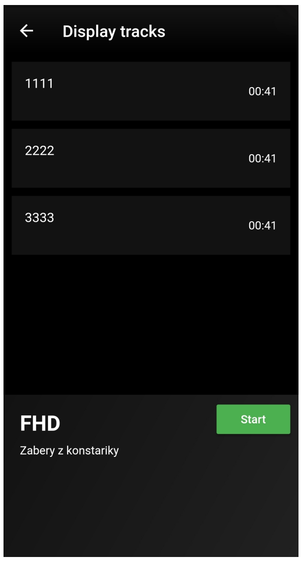
Figure D.6: Mobile app III



(a) Homepage sidebar dark theme



(b) Homepage dark theme



(c) Dispaly set detail dark theme

Figure D.7: Mobile app IV

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Trojan**	Jméno: **Ondřej**	Osobní číslo: **435005**
Fakulta/ústav:	**Fakulta elektrotechnická**		
Zadávající katedra/ústav:	**Katedra počítačů**		
Studijní program:	**Otevřená informatika**		
Specializace:	**Softwarové inženýrství**		

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Řídící systém pro multiprojekci

Název diplomové práce anglicky:

Multiprojection controll system

Pokyny pro vypracování:

Seznamte se s projektem 5D projekce připravovaným pro Muzeum hlavního města Prahy (MMP). Navrhněte a implementujte řídíci systém pro multiprojekci škálovatelný pro libovolné množství obrazovek. Systém navrhněte distribuovaný, kdy jeden počítač může řídit projekci na 1-4 displeje. Vyřešete uživatelsky přívětivou distribuci a synchronizaci multimediálního obsahu. Vstupním obsahem budou videa, zvukové stopy, složky s obrázky.
Pro systém navrhněte a implementujte řídící aplikaci pro mobilní zařízení. Navrhněte způsob interakcí dle požadavků MMP a navržené uživatelské rozhraní implementujte a otestujte s cílovou skupinou.
Celý systém otestujte s daty dodanými vedoucím práce.

Seznam doporučené literatury:

[1] J. Zettler, &quot;UX Design for Mobile: Pablo Perea and Pau Giner [Book Review],&quot; in IEEE Transactions on Professional Communication, vol. 62, no. 1, pp. 112-113, March 2019.
[2] Y. Nakamura and N. Hashimoto, &quot;Easy multi-projection system over a whole indoor space,&quot; 2018 International Workshop on Advanced Image Technology (IWAIT), Chiang Mai, 2018, pp. 1-4.
[3] Chen, R., Xue,J.-M. and He, M.T. (2018) Seamless Multi-Projector Displays Using Nonlinear Edge Blending. Applied Mathematics, 9, 764-778.
[4] SAGElab – diplomové a bakalářské práce této skupiny na FIT

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. David Sedláček, Ph.D., katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **14.02.2020** Termín odevzdání diplomové práce: **22.05.2020**

Platnost zadání diplomové práce: **30.09.2021**

Ing. David Sedláček, Ph.D.	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce		podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta