Universidade de Aveiro
2021

Cátia Marisa dos Santos Valente

Gestão de conteúdos digitais em múltiplos monitores com possibilidade de interação e recolha de estatísticas

Digital content management on multiple monitors with the possibility of interaction and statistics collection

Universidade de Aveiro
2021

Cátia Marisa dos
Santos Valente

Gestão de conteúdos digitais em múltiplos monitores com possibilidade de interação e recolha de estatísticas

Digital content management on multiple monitors with the possibility of interaction and statistics collection

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Professor Doutor António José Ribeiro Neves, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e da Professora Doutora Pétia Georgieva, Professora associada do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho à minha família, por todo apoio dado durante esta jornada

agradecimentos / acknowledgements

Quero agradecer aos meus orientadores, o Professor António Neves e a Professora Pétia Georgieva, pela disponibilidade e apoio dados durante o desenvolvimento e concretização deste trabalho.
À minha família pelo apoio e motivação para concretizar de mais um sonho, o de concluir um curso superior.
E por último, um agradecimento muito especial a todos os meus amigos que me acompanharam ao logo do meu percurso académico.

Palavras Chave

Sinalização Digital, Deteção de Faces, Reconhecimento Facial, Reconhecimento de Emoções, Deteção de Olhar, Ecrãs públicos, Visão por Computador, Extração de Dados

Resumo

Com o aparecimento de novos mecanismos de publicidade digital, nomeadamente ecrãs públicos, tem havido cada vez mais necessidade de perceber o impacto que a informação disponibilizada tem entre as pessoas para a qual estes sistemas são dirigidos. Pretende-se assim com o desenvolvimento desta dissertação criar um sistema de sinalização digital capaz de gerir remotamente conteúdo multimédia e de recolher métricas associadas às pessoas que observam os conteúdos. Essas métricas, como por exemplo a emoção demonstrada e grau de atenção prestada, serão apresentadas sobre a forma de uma dashboard, para que seja possível a interpretação do comportamento das pessoas ao longo do tempo. Além disso, pretende-se que as pessoas possam interagir com o sistema, navegando ou mesmo parando a lista de reprodução caso tenham interesse especial em algum dos conteúdos visualizados.

Keywords

Digital Signage, Face Detection, Face Recognition, Emotion Detection, Gaze Detection, Public Screens, Computer Vision, Data Extraction

Abstract

With the appearance of new publicity mechanisms, in particular public displays, it starts to have necessity to understand the impact of the information that is displayed to the target public. In this dissertation, it is intended to develop a digital signage system capable of remotely managing the multimedia content to be displayed and collect metrics associated with the people watching the content, such as emotion and gaze. These metrics will be displayed in a dashboard that will help to understand how the audience reacts to the media being displayed. In addition, it is intended that people who pass through the digital signage system can interact with the playlist if they are interested in any of the contents viewed.

Contents

List of Figures

List of Tables

Acronyms List

admin	administrator
app	application
apps	applications
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheets
GDPR	General Data Protection Regulation
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IEETA	Institute of Electronics and Informatics Engineering of Aveiro
JS	Javascript
JSON	JavaScript Object Notation
MCM	Media Content Management
PWA	Progressive Web Application
REST API	RESTful API

CHAPTER 1

Introduction

Nowadays, companies are looking for new ways to attract customers' attention to their products. For this purpose they use several methods, for example, handing out flyers, putting advertising on vehicles, billboards, and others. But most of the advertisements are static and don't change often which causes people to tend to ignore them [1].

A Digital signage can be a smart TV or a form of electronic display combined with a computer, such as a Raspberry Pi, that shows video or multimedia content in public places (figure 1.1) for informational or advertising purposes [2]. It can be found in many places such as hospitals, restaurants, airports, stores, or in bus stops[3].

The system is characterized by visual content (images, videos, and texts) that are typically found in a fixed location and can provide a variety of information, as well as dynamic forms of advertising based on customer's preferences.

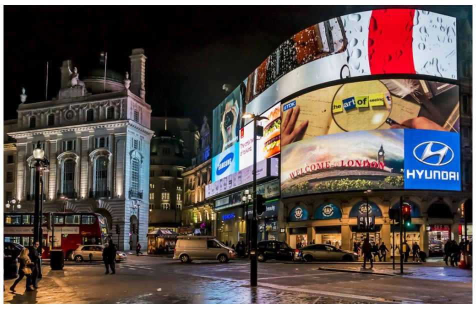


Figure 1.1: Digital Signage in Piccadilly Circus, London [1]

Their focus is to improve the value of advertising and bring higher immersion for the customers leading to an increase in sales.

Advertisements can be better targeted to customers if personal information is available, raising the chances of the right person seeing the right advertisement at the

[1]Piccadilly Circus [Online]. Available: `https://londresnalata.com/2020/07/22/piccadilly-circus-e-regiao/`

right time. However, the General Data Protection Regulation (GDPR) [4] significantly limited the possibilities of personalization, since the personal information can only be used with a written agreement of the person, which is nearly impossible within the digital signage concept [5]. Due to this fact, the personalization can be reached through an analysis of the viewer characteristics, such as height, face emotions, watching time, age, gender, and others [1].

Another characteristic of digital signage is the capability of interaction. The use of touch screens became very popular due to the growing use of smartphones. A consequence of this popularity is that people who see non-interactive digital signage are touching the screen and expecting a reaction [6].

There are many ways to interact with the digital signage [7]:

- **Touch** - individuals simply walking up and engaging with the installed screen.
- **Gesture/motion tracking** - People can walk up and move their hands in the air in front of the sign. And the cameras of the system will register the types and speed of the movements being made and trigger corresponding reactions on screen.
- **RFID/NFC** - involves the use of passive chips usually present in smartphones to deliver information or trigger a reaction.
- **QR Codes** - offering passers-by the ability to use their phone to receive information or navigate to a website by scanning a QR code.
- **Augmented reality** - there are some digital signages that have augmented reality capabilities that allow the user to engage and interact with them. It can be a fun way to bring people to have a fun new experience (figure 1.2).

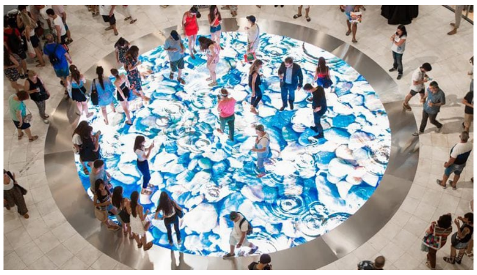


Figure 1.2: Las Arenas de Barcelona - Augmented reality digital signage[2]

1.1 Goals

The main goal of this work is to develop a digital signage system capable of collecting metrics of the audience such as emotions and attention, to be able to analyze the impact of the content being exhibited among people.

[2]TRISON digitalise the "Arenas de Barcelona" shopping center - Trison. [Online]. Available: `https://www.trisonworld.com/en/projects/trison-digitalise-shopping-center-arenas-barcelona/`.

Another goal is to give people a way to interact with the content being displayed, by using a smartphone.

Finally, it is intended to develop a platform that helps to manage the content displayed on several monitors.

Therefore, it was created three applications (figure 1.3):

- A Media Content Management responsible for managing the content displayed in the digital signage and displaying statistics.
- An agent application responsible for building and playing content and collecting pictures of the spectators.
- A mobile application responsible for interacting with the displayed content.

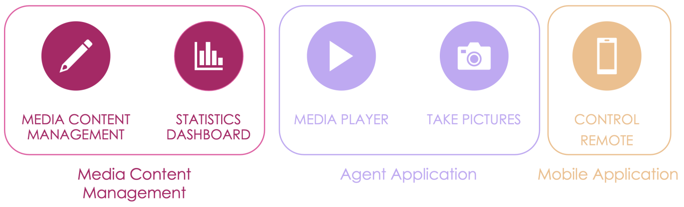


Figure 1.3: Project Structure

1.2 Dissertation Structure

In addition to the introduction, this dissertation has seven more chapters. The ones remaining are the following:

- Chapter 2 - **State of the art** - review of related work;
- Chapter 3 - **Architecture** - system architecture and its components;
- Chapter 4 - **Media Content Management** - describes the Media Content Management implementation;
- Chapter 5 - **Statistics Dashboard** - describes the dashboard designed for data analysis;
- Chapter 6 - **Agent** - describes the agent and how it interacts with the system;
- Chapter 7 - **Mobile application** - describes the application that allows interaction with the digital signage;
- Chapter 8 - **Conclusion** - an overall analysis of the results obtained and possible future improvements.

CHAPTER 2

State of the art

This chapter will introduce some solutions available on the market. However, these solutions created by companies that sell digital signs and that is why their implementation is closed to the public. But it is interesting to know what solutions exist and relevant functionalities to take into consideration.

2.1 Yodeck

Yodeck [8] is a digital signage based on the Cloud, with all services and media available on-demand, so the system is quick and reliable. It is based on a Raspberry Pi that runs the Yodeck's player. The main services available in Yodeck are:

- media cloud repository
- media management center
- configure media playlists
- schedule content to be displayed on the monitors
- customization and implementation of new features through widgets

The hardware necessary to use Yodeck consists on a Raspberry Pi, but it isn't necessary to buy one from the company (figure 2.1).

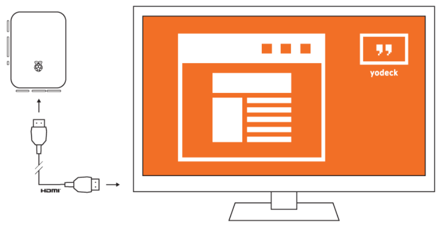


Figure 2.1: Yodeck[1]

[1]Display a web page on the Raspberry Pi like a pro – Yodeck. [Online]. Available: `https://www.yodeck.com/news/display-web-page-raspberry-pi-like-pro/`

It supports all media formats to be displayed and they can have different set of policies, depending on the user role. Despite using the cloud as the media repository, Yodeck can also work off-line. The monthly fee of using Yodeck consists in three plans[9]:

Standard	Pro	Enterprise
7,99 $ per screen	9,99 $ per screen	12,99 $ per screen

Table 2.1: Yodeck Cloud fee plans

The price presented doesn't have the Raspberry Pi price included. Yodeck can also be customized by developing Widgets to incorporate with it.

2.2 ScreenCloud

ScreenCloud [10] is a content management system (CMS) that can be accessed via a web browser or via an application (app) that helps to manage all contents displayed in a digital sign. It supports all media files such as:

- Image formats: JPEG, GIF, PNG, and SVG
- Document formats: PDF, all Office/iWorks formats
- Video formats (don't reproduce Audio): MP4, AVI

The media files can be organized into Playlists or Channels to show on the screen. The content can be dynamically changed by scheduling it for a period of time. This is a nice feature for time-sensitive advertisements, like a seasonal promotion.

ScreenCloud can inter-operate with several dashboards tools like Grafana, Kibana, and can also broadcast the information to Microsoft Teams or other conference platforms such as Zoom. In terms of hardware to use ScreenCloud it's necessary to have a Smart TV like Android TV or a TV with a Smart Box such as Amazon fire TV stick or Google Chromecast to run a player.

ScreenCloud lacks, compared to other systems, not having Playback Reports (Playback Reports, consists of reports that demonstrate that the video is played) and does not support audio files [9]. The last disadvantage it's not a major one because many digital signs don't need audio to create impact. In terms of pricing ScreenCloud Software has publicly three main plans with this monthly fees:

Starter	Teams	Business
20 $ per screen	30 $ per screen	40 $ per screen

Table 2.2: ScreenCloud fee plans

Similar to Yodeck the price is only for the software license, it doesn't consider the necessary hardware.

2.3 NoviSign

NoviSign [11] is a digital signage studio, that is easy to design by using customized templates and easy to remotely manage the media content. It supports the integration with third-party media repositories using RSS feeds, widgets, and applications (apps) such as Youtube.

In terms of media files, NoviSign permits only images and video types, it doesn't allow PDFs or Word. In terms of hardware to use NoviSign it's necessary to have a Smart-TV or a Smart-Box with one of the following operating systems:

- Samsung Tizen
- LG webOS
- Android
- Windows
- ChromeOS

NoviSign can also perform dashboarding on hardware's metrics, video uptime, and monitor the status of the digital signage screens. In terms of pricing, NoviSign has only one public plan monthly fee, after the trial period and has a cost of 20 $ per screen.

2.4 OnSign TV

OnSign TV [12] is a cloud-based solution that lets the user manage remotely multiple screens and is suitable for businesses of all sizes. It has a powerful set of applications that can be integrated with several systems and it has drag-and-drop layout customization (figure 2.2) what facilities the creation and manipulation of videos with several types of media more specifically:

- Picture formats: JPEG, PNG, SVG
- Video formats: AVI, MPEG, MKV, MP4
- Audio formats: WAV, MP3, MP4, WMA

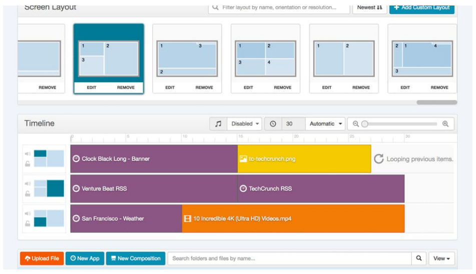


Figure 2.2: OnSign TV [2]

[2]OnSign TV | Mass Transit. [Online]. Available: `https://www.masstransitmag.com/technology/product/12303579/onsign-tv-onsign-tv`.

In terms of media OnSign can also introduce a separated audio file into a video, this means that a video can run and the audio that the hardware is emitting it's from another media file and not from the video. The software can operate in any Smart-TV, Smart-Box, Raspberry Pi or other that run the following operating system:

- Android and Android TV
- Windows
- Chrome OS
- Mac OS
- Linux
- LG webOS
- Samsung Smart Signage Platform
- BrightSign

It has some scheduling features that allow the administrator (admin) to schedule when the media are pushed into the digital signage. The admin can also define a set of different accesses for several users of OnSign and even create User Roles that aggregate a set of users with the same policies.[13].

OnSign only has two monthly fee plans they are the following:

Professional	Enterprise
19,99 $ per screen	29,99 $ per screen

Table 2.3: OnSign fee plans

The price illustrated only represents the license fee and doesn't has into account the hardware necessary to run the software.

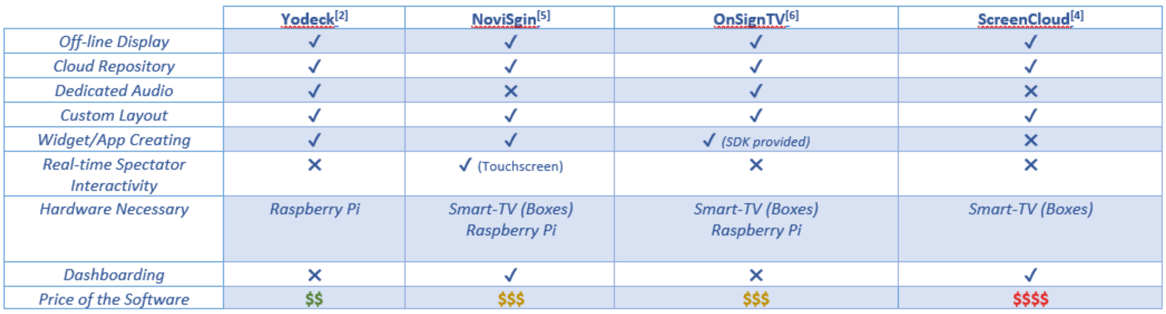
2.5 Discussion

In this section, it will be discussed the state of the art, more properly review the pros and cons of the systems referred to above. Several features are transverse between all systems and they are:

- Off-line display
- Cloud repository
- Creation of custom layout to help media display

But despite the similarities there are some key differences as illustrated in figure 2.3 one of the most important factors is the price. The price comparison only has in account the monthly license fee for one screen the most expensive license fee it's from ScreenCloud with a max of 40 $ for one screen on its premium plan. On the other hand, Yodeck has the cheaper monthly license fee per screen with the premium license with a value of 12,99 $, of course, that the choice of a product doesn't only the price into account but it has a major factor. In terms of real-time spectator interaction only Nosign has this feature through a touchscreen screen, but as suggested by the interaction the hardware must have touchscreens functionalities otherwise this feature can't be achieved.

One of the key features is customization from the client, every system except ScreenCloud permits Widgets/app creation for customization and creates new features for the system, and OniSignTV in this nature it's the only system that provides a pre-created SDK. Another key feature that defers between the systems it's the dashboarding, which means the capacity of the system to create hardware metrics of the system, and its functionalities in terms of systems only NoviSign and ScreenCloud do so, Yodeck doesn't do because it only creates reports with that information, it can't be filtered and resented on a dashboard manner.



	Yodeck[2]	NoviSgin[5]	OnSignTV[6]	ScreenCloud[4]
Off-line Display	✓	✓	✓	✓
Cloud Repository	✓	✓	✓	✓
Dedicated Audio	✓	✕	✓	✕
Custom Layout	✓	✓	✓	✓
Widget/App Creating	✓	✓	✓ *(SDK provided)*	✕
Real-time Spectator Interactivity	✕	✓ (Touchscreen)	✕	✕
Hardware Necessary	*Raspberry Pi*	*Smart-TV (Boxes) Raspberry Pi*	*Smart-TV (Boxes) Raspberry Pi*	*Smart-TV (Boxes)*
Dashboarding	✕	✓	✕	✓
Price of the Software	$$	$$$	$$$	$$$$

Figure 2.3: Systems Comparation Table

CHAPTER 3

Architecture

This chapter will go into more detail about the implemented system's architecture. It describes each component through the use cases that explain the global behavior of each actor, and the class diagram.

3.1 System architecture components

The system is composed of the following components (figure 3.1):

- a web server implemented using Django framework
- a SQLite database that stores all necessary information such as media references and agents info
- an agent, a Raspberry Pi combined with a monitor to make a smart display
- a mobile application responsible for interacting with the agent

In terms of communications between the several components, it uses HTTP requests among the server, the agent(s) and the mobile application(s). And REST for the communication of the server and the database.

There are three main modules presents in this system (figure 3.2):

- **Management Application**
  - **Media Content Management** - a web interface that will help the system administrator to manage the media content that will be displayed in the agent.
  - **Statistics dashboard** - a dashboard that will present some metrics collected from different users that watched the views.
- **Agent Application**
  - **Media Player** - for building and running the view.
  - **Take pictures** - responsible for collecting pictures from the users and sending them to the server for later data processing.

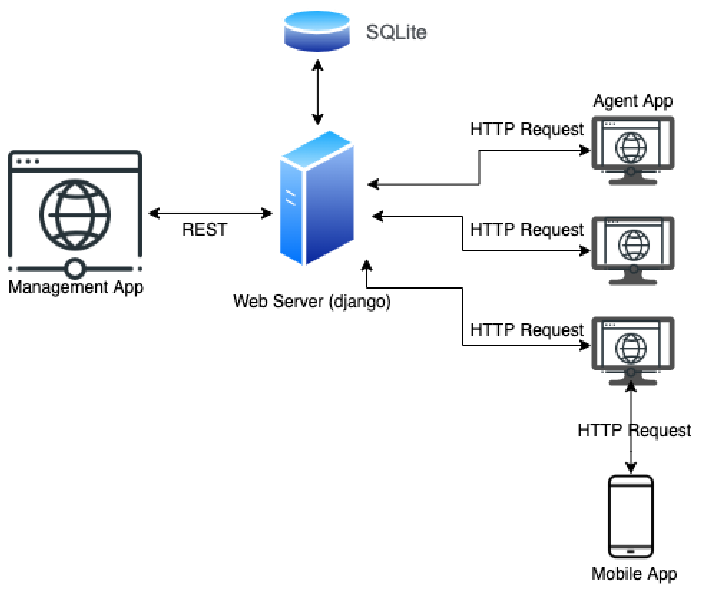


Figure 3.1: System architecture diagram

- **Mobile Application**
  - **Remote control** - responsible for letting the user interact with the player.

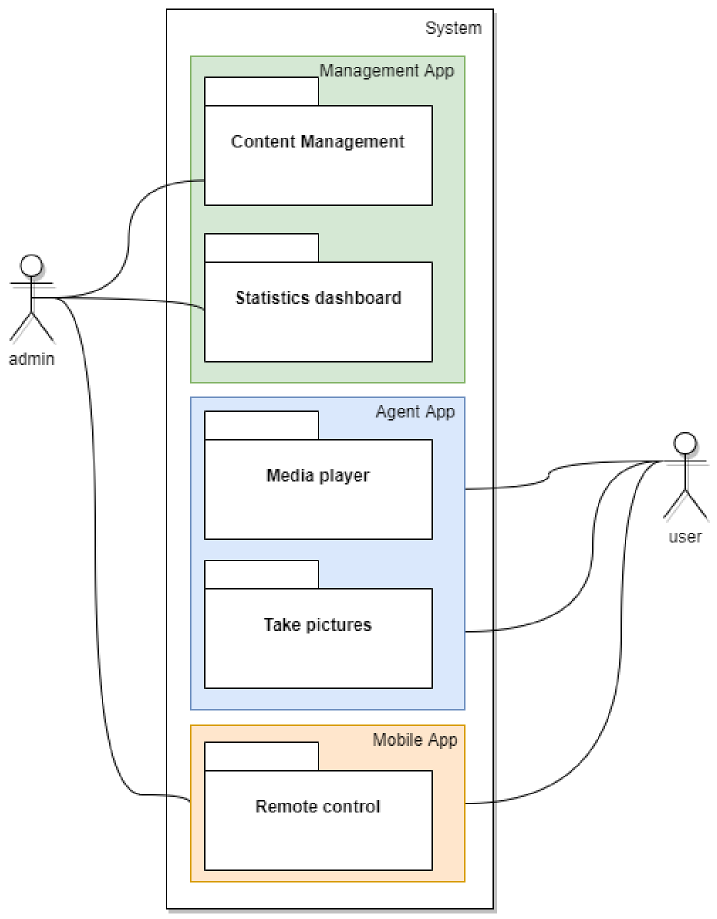


Figure 3.2: Global vision of the system

3.2 Use cases

The admin can performs several actions (figure 3.3) in the Management Application such as:

- **Check active agents** - the admin can consult information related to the agents present in the system, such as IP, MAC and host name.
- **Upload media files**
- **Create timelines** - a set of media files.
- **Create views** - a set of timelines that will be displayed on the screen.
- **Update picture rate** - configure the rate of pictures that will be taken during the view exhibition
- **View statistics** - the admin can consult statistics related to the users that watched the view

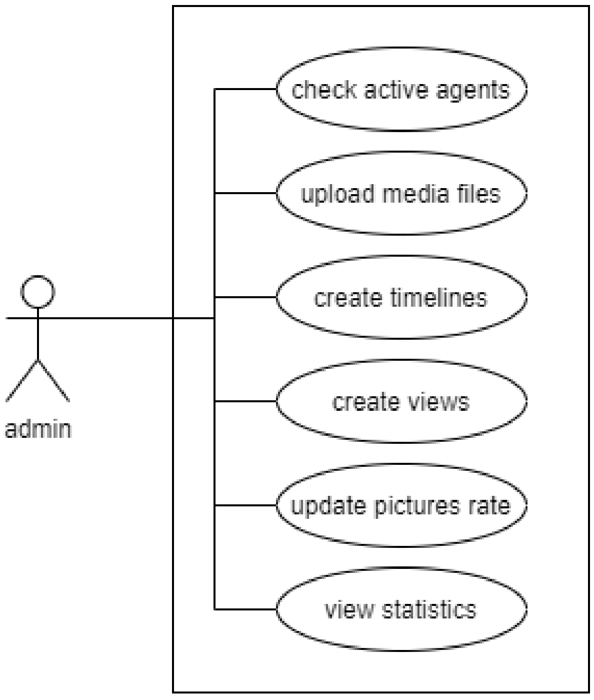


Figure 3.3: Admin use cases

The agent can perform the following actions (figure 3.4):

- **Register** - if the agent isn't registered, it must register before doing anything else.
- **Get view info** - retrieves the necessary information to determine it's view.
- **Download media files** - fetch all media files necessary to be displayed.
- **Build timelines** - assemble all media files into timelines.
- **Build a view** - compose all timelines in order to get the final view.
- **Play the view** - after the view was created, the system will play it.

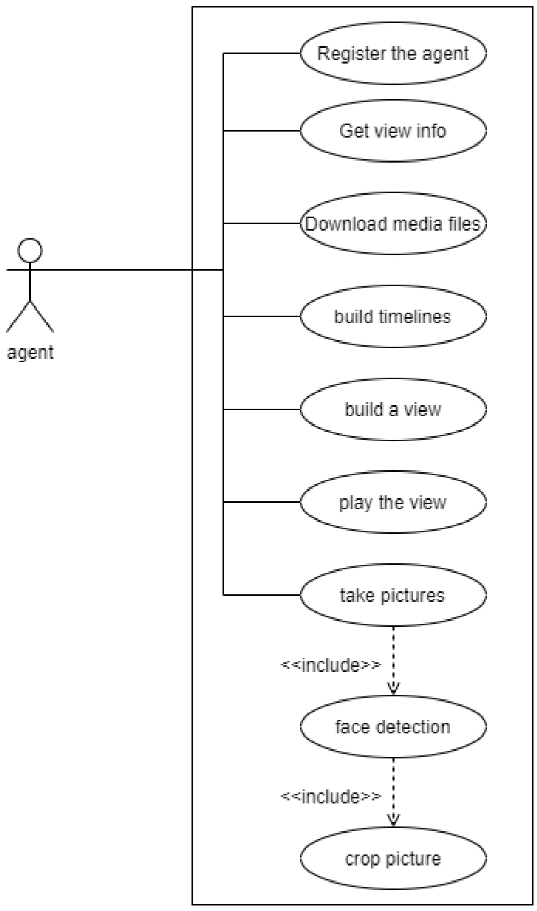


Figure 3.4: Agent use cases

- **Take pictures** - while the view is playing, the Raspberry Pi will take pictures of the users nearby the screen.

The user can do perform two main actions (figure 3.5):

- **Scan QR code** - to access the remote control for interacting with the view being displayed.
- **Interact with a view** - after scanning the QR code, the user can send commands to the agent such as pause, resume, backward and forward.

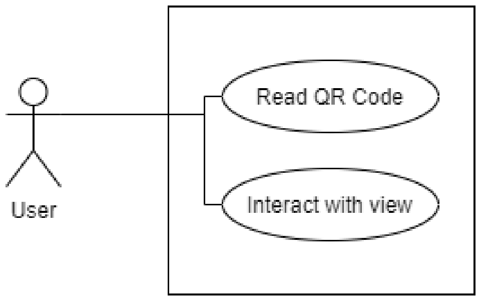


Figure 3.5: User use cases

3.3 Data Model

The data model (figure 3.9) contains eight entities:

- **Agent** - the digital signage itself. It is represented by the MAC, IP and host name of the Raspberry Pi and the display resolution.
- **Media** (figure 3.6) - a media file that is uploaded and stored in the server.

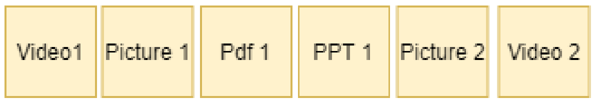


Figure 3.6: Media files

- **Timeline** (figure 3.7) - set of media files. It helps to create views faster since it is frequent to repeat media in the same view.

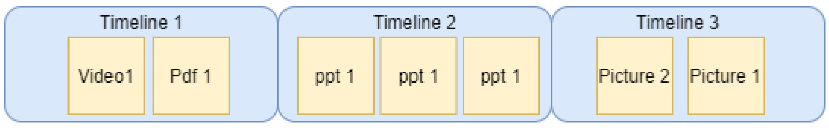


Figure 3.7: Timelines

- **View** (figure 3.8) - the content that will be displayed on the digital signage. It is composed of a set of timelines.

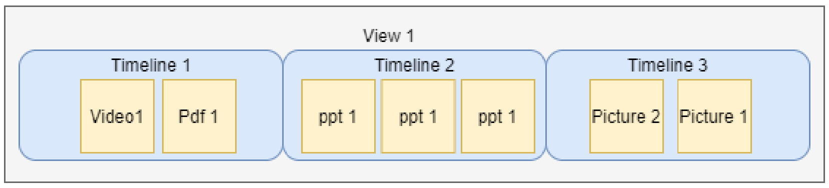


Figure 3.8: View

- **Media_Timeline** - entity used for associate multiples media to multiples timelines. The duration attribute helps to give a notion of time to static files, like pdf or images, since unlike a video that has play time, they don't. The order attribute helps to organize media inside the timeline.
- **Timeline_View** - entity used for associate multiples timelines to multiples views.
- **Agent_view** - direct association between an agent and a view. Each agent only have a view associated. In this entity is also possible to configure the picture rate from the pictures taken by the Raspberry Pi.
- **Person** - entity that represents the user that watches the media displayed in the agents. During a view exhibition, the agent will take several pictures of the user, which will be processed and analyzed to determine the user's emotion and attention.

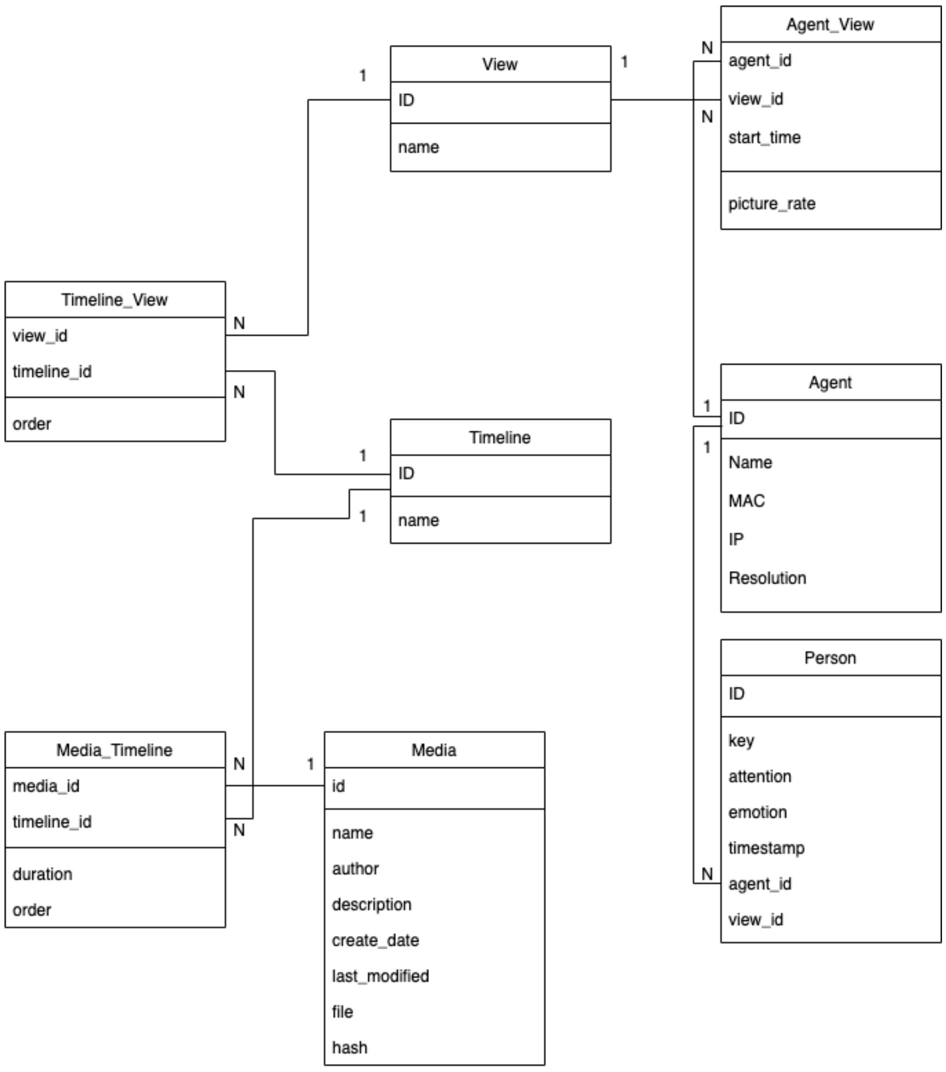


Figure 3.9: Class diagram

CHAPTER 4

Media Content Management

This chapter will describe the media content management system, by presenting what it is and its own purpose. Additionally, it will introduce the existing interfaces and discuss the technologies used in the implementation.

Media Content Management is responsible for managing the media displayed in the Digital Signage. By providing interfaces for Create, Read, Update and Delete (CRUD) operations, it allows to upload media and arrange them into timelines and views that will displayed later on in the agent.

This entity was developed as a Django web application and it is in control of the following features:

- Managing the media content
- Preview media content
- Face Recognition, Emotion Recognition and Gaze Detection

In the following sections each feature will be described with more detail.

4.1 Managing the media content

Before the agent collects all information needed to show the view, it is crucial that an administrator configure all elements necessary. The system accepts the following media types:

- JPEG and PNG for images
- PDFs
- PowerPoints
- MPEG and AVI for videos

During the file upload, it's calculated and stored its SHA-1 hash. SHA-1 was the chosen hash algorithm because despite its vulnerabilities it's still found to be standard and secure compared with MD5 [14], having a small hash size. Next, the admin can configure timelines, to do so he creates a new timeline and adds media to it. To create non-static media from static ones it's necessary to set duration into the media while configuring the timeline. After setting the timelines to a view, the admin must assign it to a(n) agent(s).

4.2 Preview media content

Each time the admin consults media information or updates some information, it is possible to preview the media content.

4.3 Face Recognition, Emotion Recognition, and Gaze Detection

It is a module developed by Daniel Canedo during his studies on Monitoring Students in the Classroom Through Computer Vision [15]. Given the faces cropped sent by the agent, it gets the id of the user present in the system (if not presented, it is added), and by analyzing the faces it calculates the emotion and the attention of the person. In the end, the resulting data is persisted in the database for later representation. Finally, every face present in the server is destroyed, to comply with the GDPR regulation [4].

4.4 Media Content Management Interfaces

The Media Content Management manages all entities' data represented in the data model (section 3.3) (with the exception of Person entity) and allows to perform basic operations, such as listing, adding, editing, and removing an element, also known as CRUD operations. Therefore, in this subsection will be presented one entity interfaces, since the rest of them are pretty similar. It will also highlight and explain relevant operations/behaviors of other entities.

4.4.1 Entity entries layout

In this screen (figure 4.1), it is possible to navigate to the other entities through the menu at the top of the page (1.1).

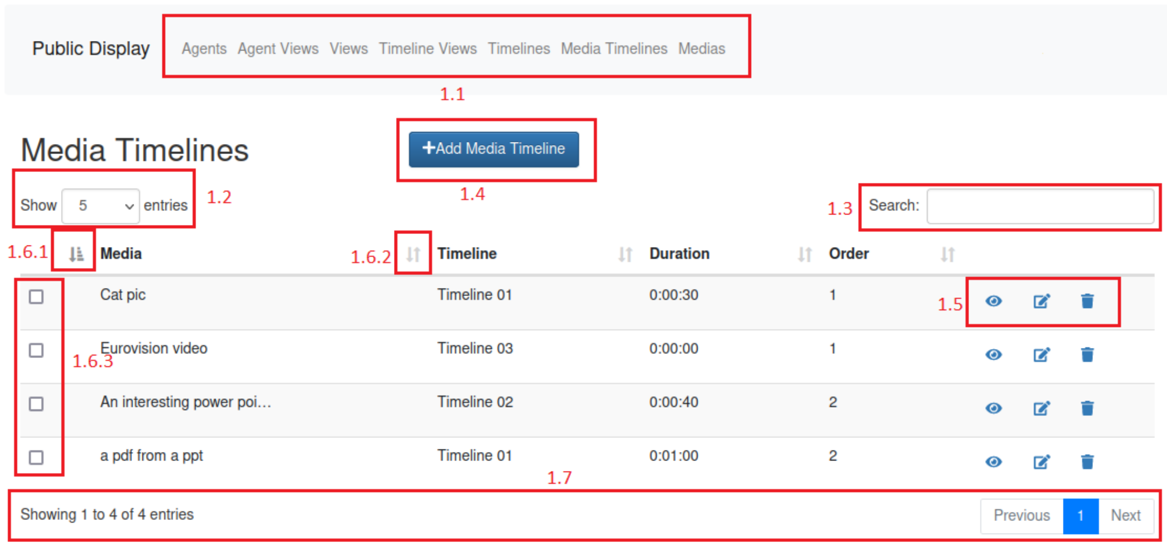
To select the number of items to display in datatable, just select the dropdown in 1.2. It will be helpful when there is a huge amount of data and it can adjust the size of each page.

It also allows doing search data in the datatable, by typing a term or a date in the search bar (1.3).

To sort data, it is necessary to click on the sort icon on the header of each column (1.6.1). By default, it is ordered by id. The dropdown presents in each row (1.6.3) represents the field id. The rest of the icons 1.6.2 show that exists more columns with the sorting capability but still aren't selected for it.

At the bottom of the page, there is the pagination section (1.7).

The operations allow to perform in each entry of the datatable are view detail, edit and remove an element of the datatable (1.5). To add a new element, just click on the "Add Button" (1.4).



	Media	Timeline	Duration	Order
☐	Cat pic	Timeline 01	0:00:30	1
☐	Eurovision video	Timeline 03	0:00:00	1
☐	An interesting power poi...	Timeline 02	0:00:40	2
☐	a pdf from a ppt	Timeline 01	0:01:00	2

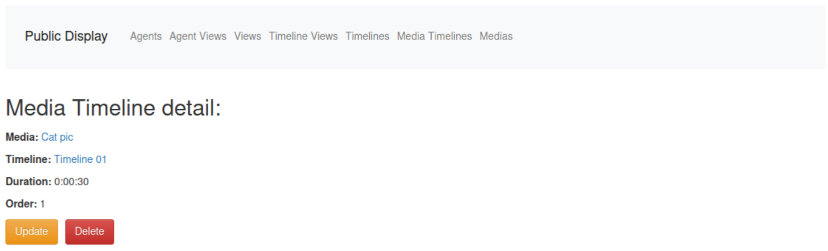
Showing 1 to 4 of 4 entries

Figure 4.1: Media Timelines list

4.4.2 Entity detail

After clicking on the eye icon(1.5), it opens a new screen (figure 4.2 that sometimes displays more info that is not available in the initial datatable screen.

In this screen, it is also possible to perform the edit or remove action.


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias
Update
Delete

Media Timeline detail:

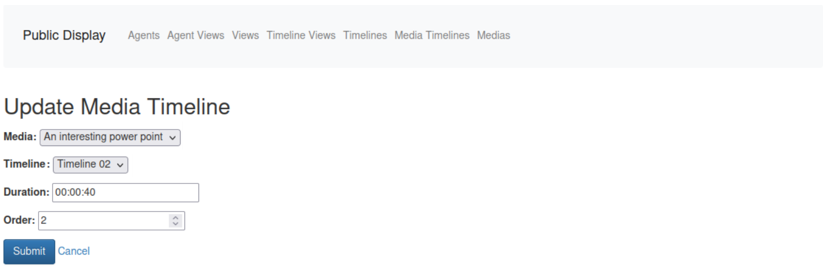
Media: Cat pic
Timeline: Timeline 01
Duration: 0:00:30
Order: 1

Figure 4.2: Media Timeline detail

4.4.3 Update an entity

To update an entity, it is necessary to click on the pencil button (1.5) and the update screen will open.

After submitting the changes, it will navigate to the detail page for the user confirm if all data is in concordance.



Update Media Timeline

Media: An interesting power point

Timeline: Timeline 02

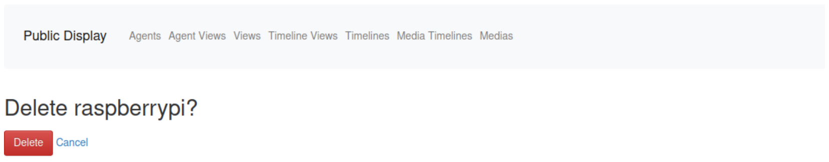
Duration: 00:00:40

Order: 2

Figure 4.3: Media Timelines edition

4.4.4 Remove an entity

To delete an entity, just click on the trash can button (1.5), and then it is necessary to confirm whether it is to eliminate or not (figure 4.4).

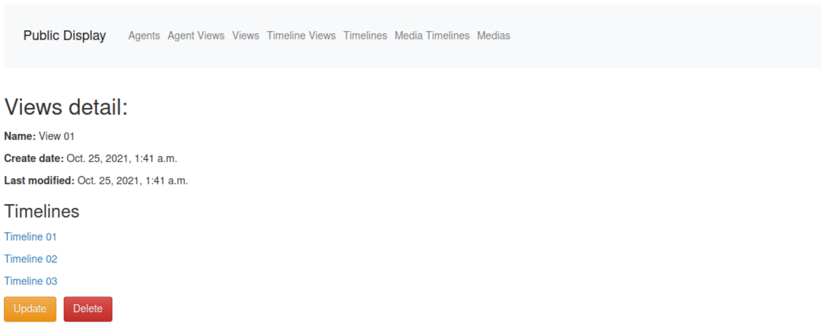


Delete raspberrypi?

Figure 4.4: Agent delete

4.4.5 Navigation between screens

In the Views detail is possible to know which timeline is part of the view. By clicking in the timeline name, it is possible to navigate to the respective timeline (figure A.8).



Views detail:

Name: View 01

Create date: Oct. 25, 2021, 1:41 a.m.

Last modified: Oct. 25, 2021, 1:41 a.m.

Timelines

Timeline 01

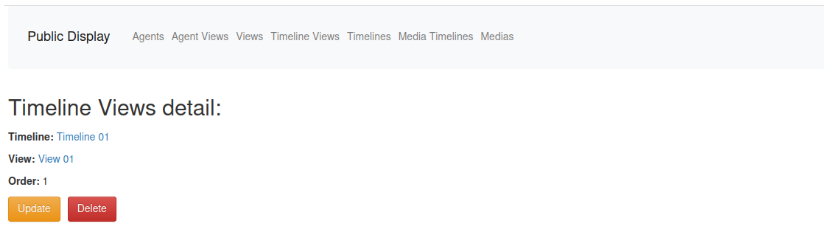
Timeline 02

Timeline 03

Figure 4.5: View detail

The same behavior appends in the Timelines detail, which displays the respective media list.

In entities that represent many to many relationships such as Timeline Views, Agent Views, and Media Timelines, it is allowed to navigate for each element that is part of the relationship (figure 4.6).



Timeline Views detail:

Timeline: Timeline 01

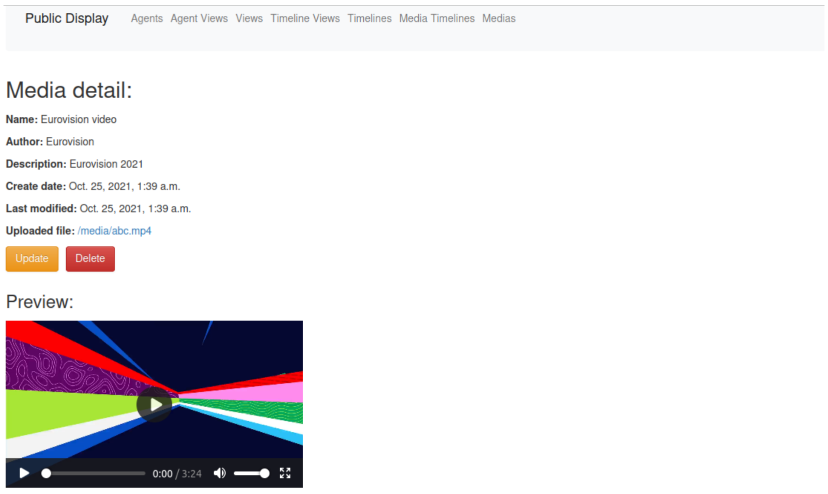
View: View 01

Order: 1

Figure 4.6: Timeline Views detail

4.4.6 Preview Media

When consulting the Media detail, it is available a media preview from that file, previously uploaded to the system (figure 4.7).



Media detail:

Name: Eurovision video

Author: Eurovision

Description: Eurovision 2021

Create date: Oct. 25, 2021, 1:39 a.m.

Last modified: Oct. 25, 2021, 1:39 a.m.

Uploaded file: /media/abc.mp4

Figure 4.7: Media preview

4.5 Technologies used

In this section, it will be discussed some technologies available in the market and justify the choice of each one for the implementation of the Web servers and the database.

One main requirement was that the Media Content Management should be written in Python and for that reason the technologies reflected upon were Flask and Django. In terms of the database there were several but it was required that the one selected would integrate well with the choose web server.

4.5.1 Web server

Flask

Flask [16] is a python-based micro-framework for web applications development and was designed to be easier to use, lightweight, fast, and to scale up to complex applications and micro-services. It provides a lot of freedom to build the web application layer since it doesn't have boilerplate code or dependencies [17].

Django

Django [18] is an open-source Python-based framework that runs on the server-side to create efficient and attractive websites quickly [19]. An application developed in Django is secured and protected against a wide range of typical vulnerabilities, such as SQL injection, cross-site scripting, etc. and scalable to meet traffic demand [20]. It provides a default admin panel that helps to manage data, permissions, and other relevant configurations.

Decision

It was decided to use both technologies for different purposes. Django was selected to be the server-side because the project integrates a facial recognition module written in python that demands a good throughput to receive and processes all the information that the module sends to it.

Additionally, since the web interface will be used mainly for CRUD operations, it is not necessary to have a complex library since these operations are created using templates in the Django framework that makes it very quick to develop.

On the other hand, Flask will be used in the agent (who have resource limitations) for a minimalist application, to have an open port waiting for receiving new commands, because it is lightweight, fast, and does not require a lot of dependencies to be installed and used.

4.5.2 Databases

After selecting Django as our web server, it was necessary to choose the right database to integrate into this project to persist data. Django is compatible with several databases such as MySQL and SQLite [21]. In this subsection will go into more detail about these two databases and their characteristics.

MySQL

MySQL [22] is an open-source relational database management system (RDBMS) developed by Oracle that allows managing relational databases based on structured query language(SQL).

MySQL is easy to install, lightweight allows multi-threading which makes it scalable, and it has high performance because of its distinct storage-engine framework.

Another point in favor is that it runs on all key platforms and it has a large number of contributions available, which makes it such a desired software.

SQLite

SQLite [23] is a file-based relational database management system (RDBMS), which means that it is a direct file system engine that uses SQL syntax, and that is why it considering a light database (in average the size of this type of databases can easily be under 600 KB [24]). This characteristic facilitates its portability since it just needs to copy the database file to the other file system. In general, runs faster, since it only loads the data which is needed instead of holding it in memory, even in low-memory environments [25] and it can be used in all platforms [26] and it doesn't require any installation or setup.

Decision

Each one has its own characteristics but it was decided to go for the SQLite since it has a better performance [27] (figure 4.8), it is portable, lighter and doesn't require any complex installation or server running [26].

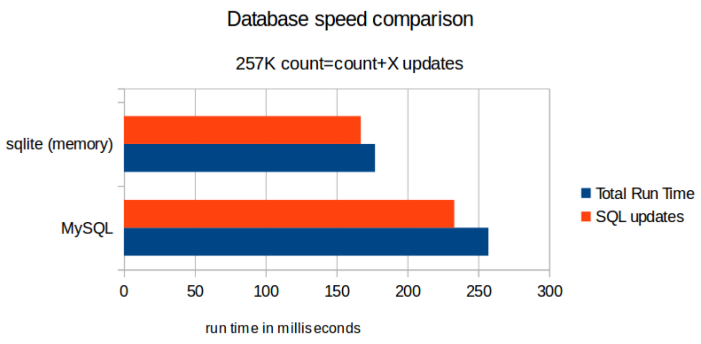


Figure 4.8: Database Speed Comparison between mySql and SQLite[1]

[1]How switching to MemSQL made my day | by jadi | Medium. [Online]. Available: `https://medium.com/@jadi/switching-to-memsql-made-my-day-4a6a3ab287b9`.

CHAPTER 5

Statistics Dashboard

This chapter presents the Statistics Dashboard designed to show the metrics collected during the content exhibition and analyzing them through time.

As mentioned earlier, today it is of great importance to know the audience better and draw attention to the content displayed on digital signage. With this purpose in mind, our system was developed with the goal of understanding if the content displayed on the screen is or is not relevant to the user, through the understanding of the level of attention and emotions felt by the users during the exhibition. Therefore, after the agent starts displaying the view, the agent also starts collecting pictures of users who pass in front of the camera. Then, from these pictures, relevant metrics are generated to better understand the user who passed in front of the monitor, such as:

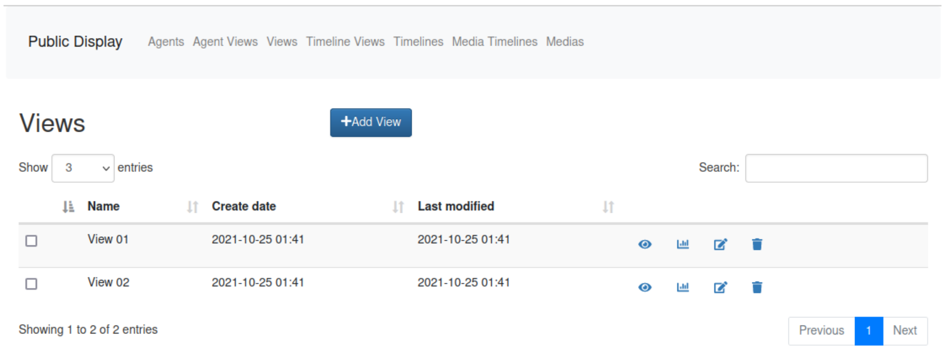
- Total frames captured
- Number of users detected
- Emotion per frame
- Attention per frame
- Average attention

Since the dashboard was designed based on a view, to access the dashboard is necessary to go to the views list and click on the chart button in the correspondent view that the user wants to analyze (figure 5.1).

The dashboard is divided into three parts. The first part is a global information section that displays frames captures, users identified, average attention, and the list of agents that used that view (figure 5.2).

The second part is the graphical information. It shows the emotion and attention detected through time.

The third part serves to filter the data displayed in the dashboard in a period of time and/or by an agent.



Public Display

Views

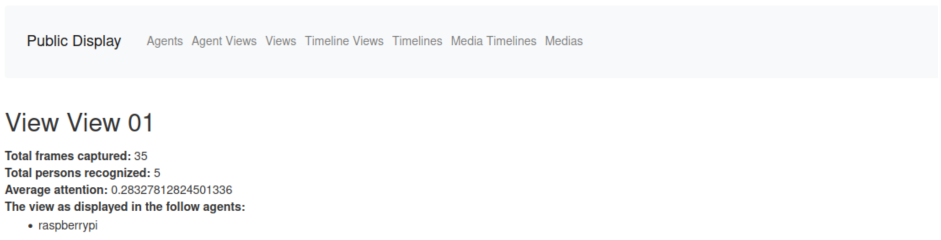
Show 3 entries

Search:

	Name	Create date	Last modified	
☐	View 01	2021-10-25 01:41	2021-10-25 01:41	
☐	View 02	2021-10-25 01:41	2021-10-25 01:41	

Showing 1 to 2 of 2 entries

Figure 5.1: View list



Public Display

View View 01

Total frames captured: 35
Total persons recognized: 5
Average attention: 0.283278128245013336
The view as displayed in the follow agents:

- raspberrypi

Figure 5.2: Dashboard - global information

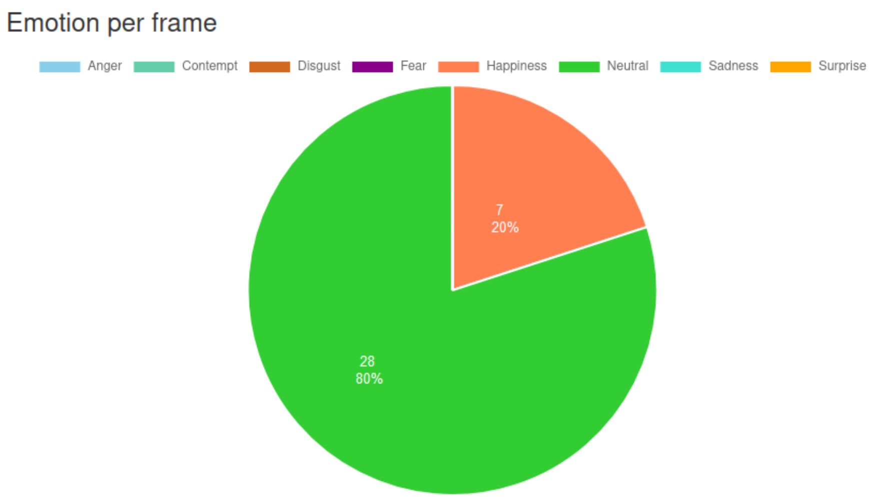
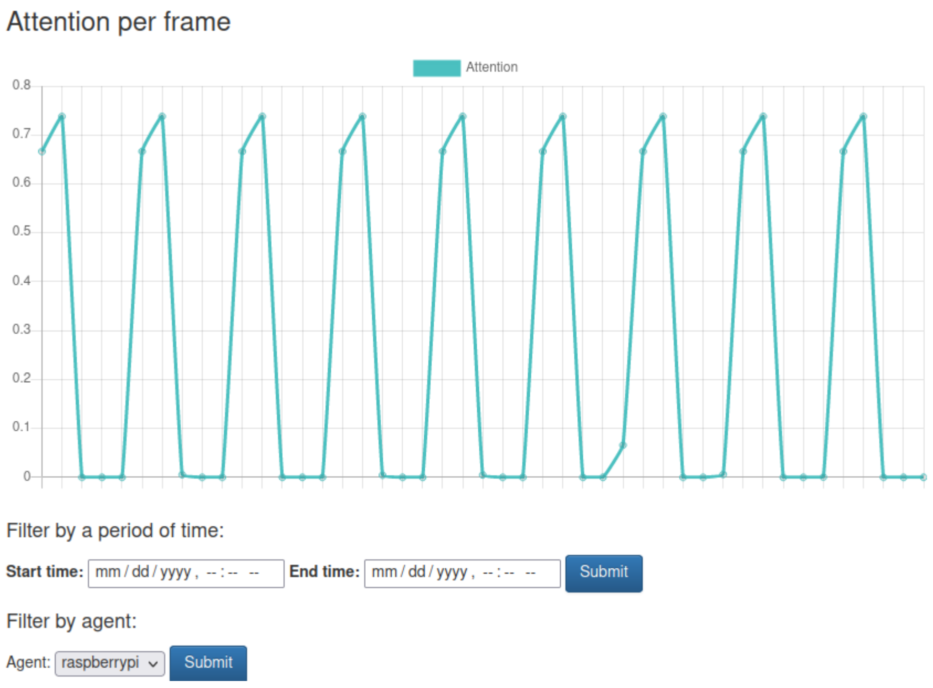


Figure 5.3: Dashboard - Emotion per frame chart



Filter by a period of time:

Start time: mm / dd / yyyy , -- : -- -- **End time:** mm / dd / yyyy , -- : -- -- Submit

Filter by agent:

Agent: raspberrypi Submit

Figure 5.4: Dashboard - Attention per frame chart and filter section

5.1 Technologies used

The graphs were designed with the ChartJs [28] library based in Javascript (JS). It was decided to choose it besides D3 [29] which is the most popular library to data visualization, since D3.js requires more effort to create charts [30] and for the necessary data to be represented, only standard charts are needed.

CHAPTER 6

Agent

This chapter will introduce the definition of an agent, what are the features available, and how the features interact with the whole system. It will present the correspondent activity, sequence and block diagram, and finally, the libraries used through the process.

6.1 The agent

The agent is the digital signage itself, composed of a display with a Raspberry Pi [31] equipped with a Pi Cam [32] module, that runs a view in a media player with some strategy associated. This entity is responsible for:

- Get media files from the management system, via HTTP requests, build and play a view
- Take pictures from the users that pass in front of the display, detect and crop faces and send them to the server
- Check for new content updates

In the following sections, it will be described in detail each agent's feature and the technologies used for its implementation.

6.2 Build and play a view

As digital signage, the main goal of the agent is to display a view (figure 6.6). After the agent initiates, it will try to register itself. If it was already registered it will update its own information, such as IP address, hostname, MAC address, and resolution. The agent will inquire about the Media Content Management to check if there is any view associated. After the retrieval of the media files, it will build a movie playlist (view). If no view was assigned, it will query the Media Content Management, every 5 seconds, until a view has been assigned.

The process of creating a movie playlist (view) starts by:

- getting all timelines of the view from the Management System
- retrieve all media files associated with the timeline
- construct a movie with all media files and its duration, based on the timeline
- append all the timeline movies into a movie playlist, the view to be displayed

As mentioned before, static files, like a PDF, a picture, or a PowerPoint, need to be converted into playable files, also known as videos. For the conversion to occur it's necessary to establish a frame rate. The frame rate is a frequency (rate) at which consecutive images (frames) are captured or displayed, so by displaying multiple frames, it is possible to create a video. Therefore all media need to have a duration associated. This duration (frame rate are stored in the media_timelines) defines the display time of that media.

Then, a media file conversion will take place. Every PowerPoint is converted into PDF and PDFs into images, one per page and each picture originated will be written as a video.

To convert between formats (figure 6.1) it was used the following python libraries:

- **soffice** - to convert a power point into a PDF
- **pdf2image** - to convert a PDF into a image

It also uses OpenCV to adjust the resolution of the media content. After the media

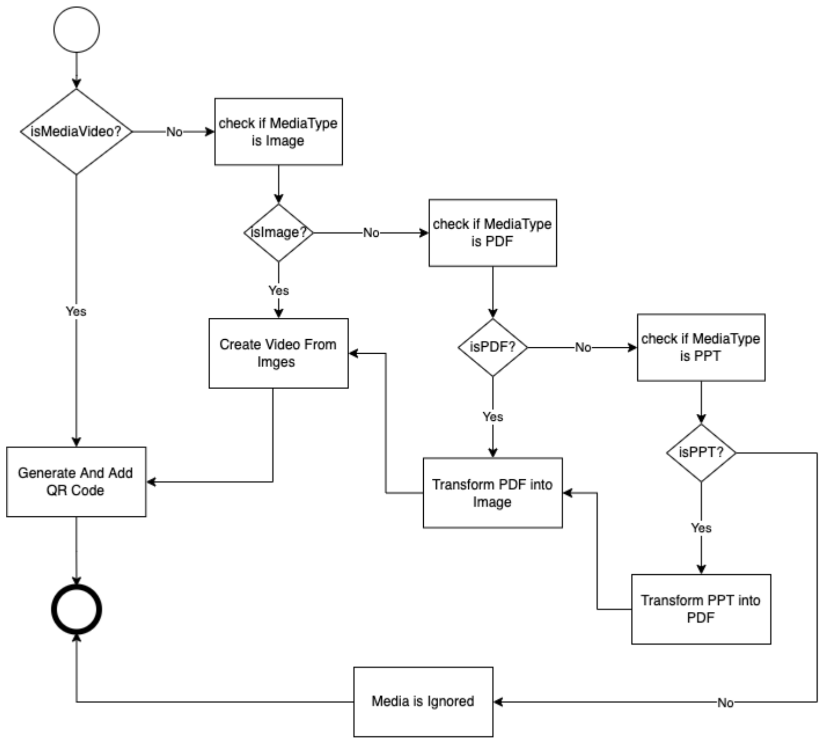


Figure 6.1: Agent video processing flow

file's conversion, the resulting videos are inserted into an ordered list that will later become the media playlist to be displayed.

The same could be achieved by concatenating all the videos into a single but, making a playable media list has several advantages such as:

- Don't waste processing time concatenating them
- Easier to identify the media file that is playing when the pictures are captured, and posterity associate emotion to a content (file)
- The possibility to select content to display, when interacting with the view.

Before sending the playlist to the VLC Media Player, a QR code is generated, with the agent's IP address, for later identifying the agent when interacting with the view.

After the QR was added (through the FFmpeg library), the playlist is sent to the VLC Media Player, who will play it continuously.

6.3 Take pictures

While the Raspberry Pi is playing a view, it is collecting pictures of the people that are passing in front of the screen, at a given rate, previously configured in the server(in the agent_views). The face detection algorithm detects every person present in the pictures taken and it will result in another set of pictures with all cropped faces. Then the agent sends the faces to the server to be processed through a face recognition algorithm to determine the corresponding person's ID. Finally, it calculates the gaze and the emotion associated with each person and persists the data the database collected (figure 6.2).

In case of a picture has multiple faces, it will crop each face into a file, if no faces are detected, the agent will discard the picture taken.

If for any reason the crop pictures were not sent to the server, they will be stored in the Raspberry Pi until it can be possible to establish the connection with the server and after sending them, all images processed will be deleted from the Raspberry.

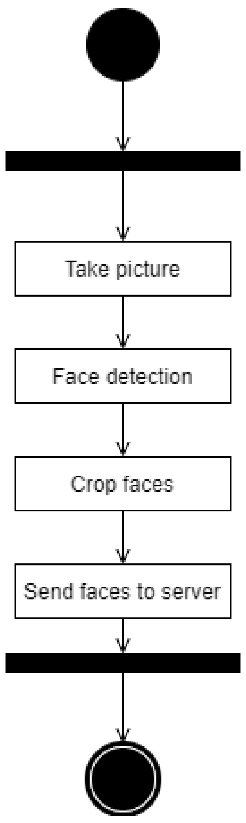


Figure 6.2: Take pictures - activity diagram

6.4 Check for new updates

After the agent gets the assigned view, and before starting the process of building a view, the agent saves locally the view's structured as the view state, in a JavaScript Object Notation (JSON) file (figure 6.3). After the view starts playing, the agent will inquire the server every 5 minutes for new updates by sending its current state.

When the server received the check_for_new_updates request, it will compare the agent's state with the information stored in the database and will send back to the agent feedback about if the agent is updated or not.

The server will check if the view is still the same assigned to the agent, will compare the number of timelines in the view, the timelines' id, the media content(through the media files' hash), the order, and the duration of each media inside the timeline.

If the view was changed, the agent will request every new media necessary to rebuild it (figure 6.4).

It was chosen to check for new updates every 5 minutes to avoid occurring bottleneck in the Raspberry Pi.

```
{
   "agentId":"8c:85:90:c7:7b:d9",
   "viewId":1,
   "timelines":[
      {
         "id":1,
         "medias":[
            {
               "media_id":1,
               "hash":"793f970c52ded1276b9264c742f19d1888cbaf73",
               "duration":"00:01:15",
               "order":1
            },
            {
               "media_id":1,
               "hash":"793f970c52ded1276b9264c742f19d1888cbaf73",
               "duration":"00:02:30",
               "order":2
            },
            {
               "media_id":2,
               "hash":"a94a8fe5ccb19ba61c4c0873d391e987982fbbd3",
               "duration":"00:01:25",
               "order":3
            }
         ]
      }
   ]
}
```

Figure 6.3: Check for new updates - View structure stored in the agent

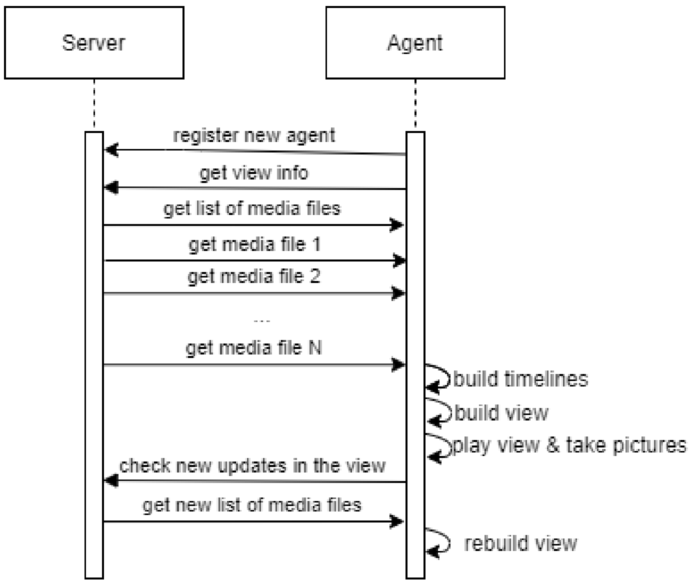


Figure 6.4: Check for new updates - sequence diagram

6.5 Overview

Resuming, the agent is responsible for retrieving data from the Media Content Management and building the view. After it, as long as the view is playing it will take pictures, crop faces and, send them to the Media Content Management for collecting metrics. To illustrate the communication and the operations between the agent and the Media Content Management regarding the Take Pictures flux, it was created the figure 6.5.

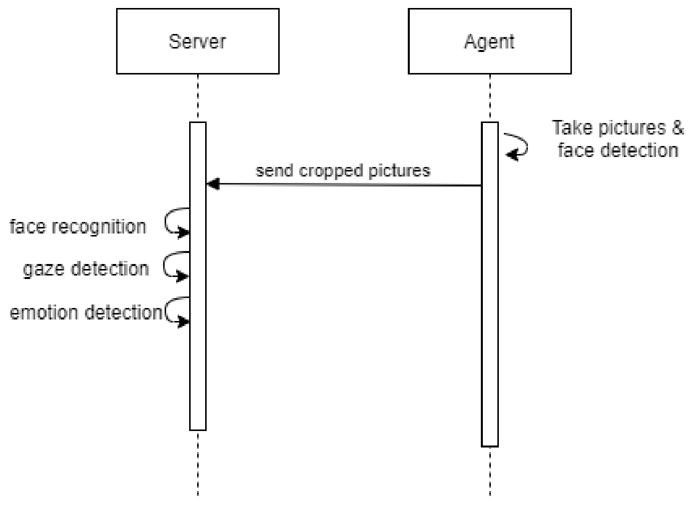


Figure 6.5: Take pictures - sequence diagram

The figures 6.6 illustrate the main activities that occur on the agent, while the figure 6.7 illustrate the communication and the orchestration between the Agent and the Media Content Management while performing the two main tasks (build a view and take pictures).

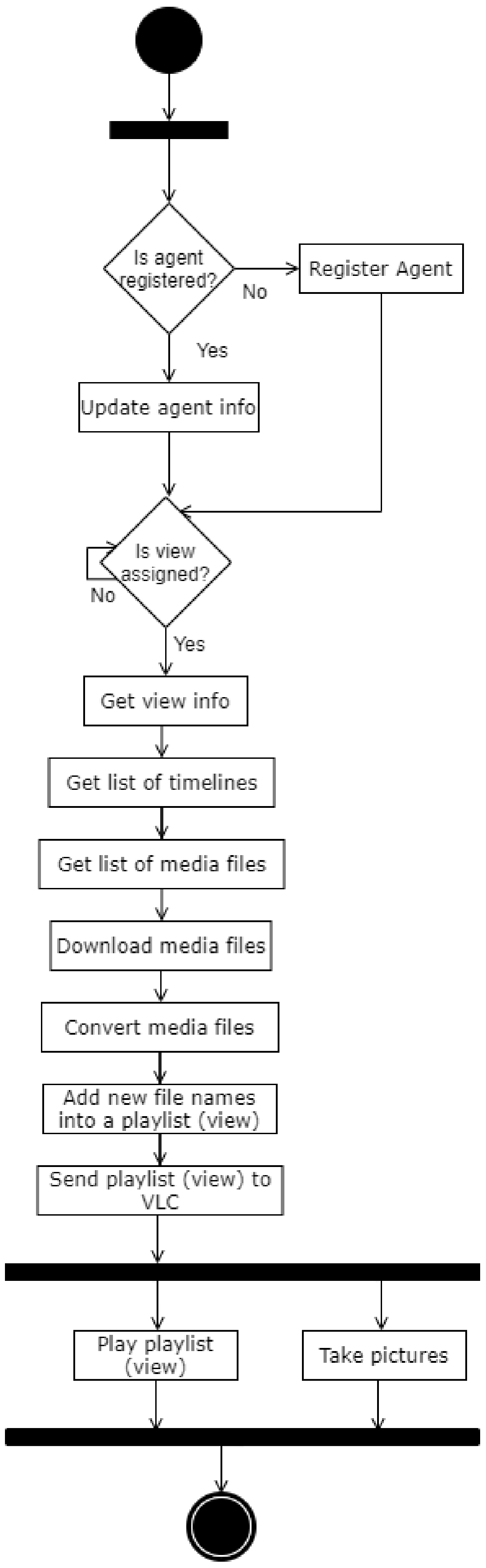


Figure 6.6: Build and play a view - activity diagram

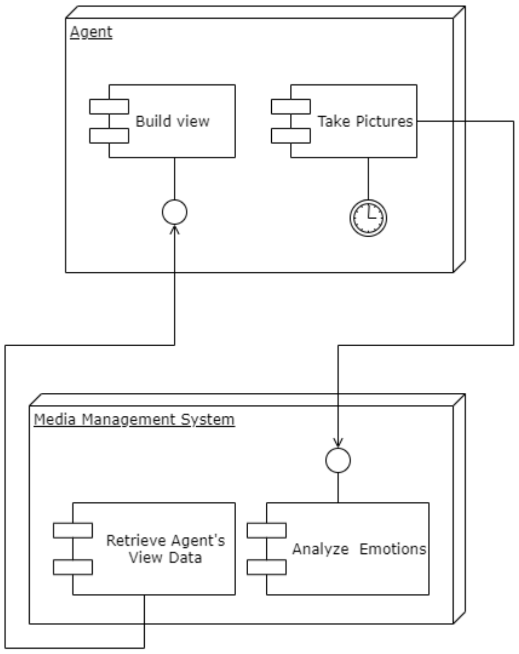


Figure 6.7: Agent Block Diagram

6.6 Media content transformation software/libraries

6.6.1 OpenCV

OpenCV [33] is an open-source library used for computer vision, machine learning, and image processing. The OpenCV offers a set of pre-built image processing functions that allows the video and image manipulation. It was used to process images, resize and manipulate them.

6.6.2 FFmpeg

FFmpeg [34] is an open-source project consisting of a set of libraries and programs to handle video, audio, and other multimedia files. It is command-line based and is considering a faster coding framework as it can be configured to work in multi-thread improving performance and quality [35].

It was used in this project to add the QR code to each frame of the produced videos, to be able to have it as a watermark present in all media displayed since it has a good performance compared with other video processing technologies since it uses a multi-threading paradigm to speed up the video processing [36].

CHAPTER 7

Mobile application

This mobile application was created in order to allow the users to engage with the digital sign, through the interaction with the view displayed. To this purpose, it was created an application that allows the reading of a QR Code for the users can perform a few commands in the view displayed in the agent:

- **Play** the view
- **Pause** the view
- **Forward** the view 5 seconds
- **Backward** the view 5 seconds

First the user needs to scan the QR code available on view (figure 7.1). This QR code will identify the agent where the user wants to interact.

After that, a screen with 4 buttons, representing each action available, will appear, to allow the user starts the interaction (figure 7.2).

To go back to the initial scanning screen and scan another agent, the user just needs to tap the arrow in the upper left corner of the screen.

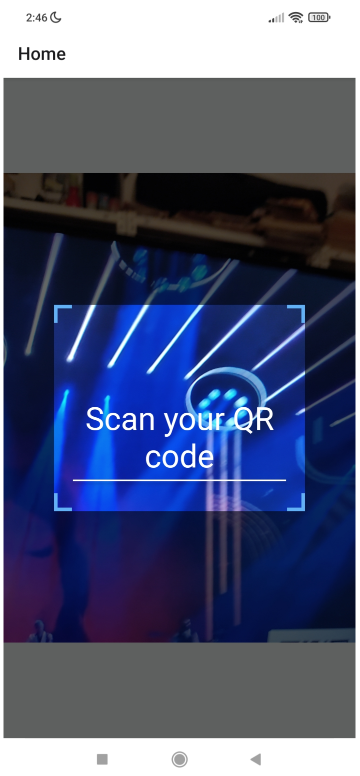


Figure 7.1: Read QR code

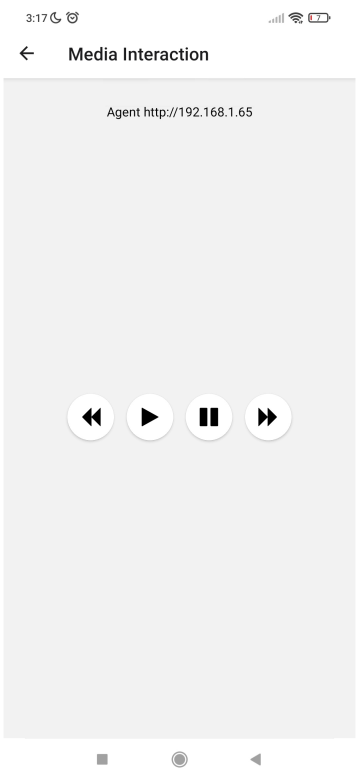


Figure 7.2: Remote Control

7.1 Activity diagram

Each time a user press a button, the application sends a command (such as "r" for play or resume, "p" for pause, "b" for backward and "f" for forward) to the respective agent, and the agent performs that action.

The agent has a flask [16] service running on the port 5000 that is listening commands, and each command received is applied to the current view that is being displayed in the VLC player (figure 7.3).

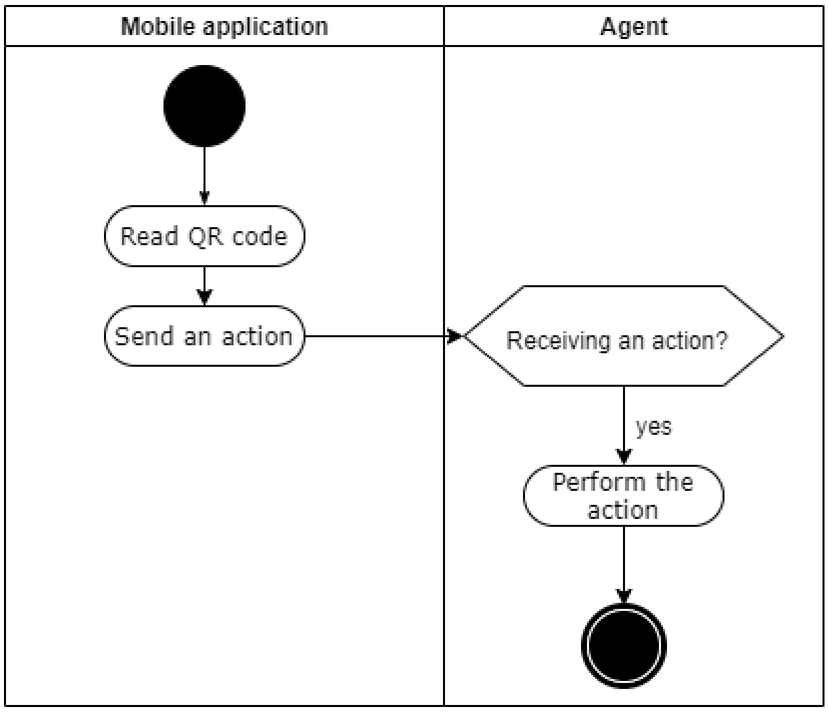


Figure 7.3: Mobile Application - activity diagram

7.2 Technologies used

Native applications

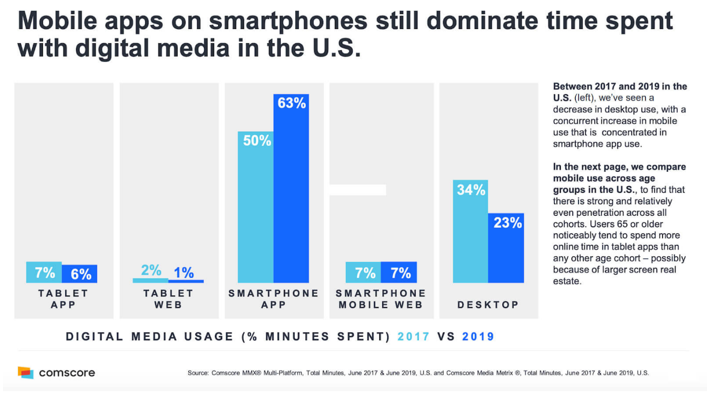
In Native applications [37], it is necessary to know different languages for developing to different devices. For instance, to develop to Android it is expected to know Java or Kotlin and to develop for IOS, Swift, or Object-C. They have the advantage of being faster since they have direct access to the hardware of the device and they are optimized for different platforms [38], and they are also more secure.

React Native

It is based on the React framework. React Native [39] apps are true native applications and are allowed to have access to the full features or functionalities of the operating system and the hardware of the device.[40] It has the advantages of not needing to develop for multiple platforms [41] and makes use of the GPU to get a higher speed performance, which makes it a good alternative for native apps [42].

Progressive Web App (PWA)

Progressive Web Apps[43] are essentially web applications that run in a browser, built using common web technologies including HyperText Markup Language (HTML), Cascading Style Sheets (CSS) and JS with access to native features like push notifications, gestures, or access to the camera. Due to the fact that they are just a web application, it is easy to distribute, since it is not necessary to publish on the google play or apple store but still, it isn't easier to spread an application when they aren't cataloged in an application store [44] (figure 7.4).


Mobile apps on smartphones still dominate time spent with digital media in the U.S.

Between 2017 and 2019 in the U.S. (left), we've seen a decrease in desktop use, with a concurrent increase in mobile use that is concentrated in smartphone app use.

In the next page, we compare mobile use across age groups in the U.S., to find that there is strong and relatively even penetration across all cohorts. Users 65 or older noticeably tend to spend more online time in tablet apps than any other age cohort – possibly because of larger screen real estate.

Source: Comscore MMX® Multi-Platform, Total Minutes, June 2017 & June 2019, U.S. and Comscore Media Metrix ®, Total Minutes, June 2017 & June 2019, U.S.

Figure 7.4: Mobile apps are more popular[1]

Decision

It was selected to work with React Native because one of its advantages is that supports cross-platform. It lacks in performance comparing with a native Android application, however, the differences are small [45] making React Native maintaining the upper-hand. Progressive Web Application (PWA) does not perform in the same way as native apps due to the lack of direct access to the device hardware[46]. In terms of app delivery React Native is cross-platform, PWA are also, but according to [47], some stores, like Apple store, don't allow the publishing of PWA in their platform. So for that reason, the mobile application was developed in React Native

[1]PWA vs Native Apps: Getting Technology Right in 2020. [Online]. Available: `https://apiko.com/blog/pwa-vs-native-apps-how-to-choose-your-technology/`

CHAPTER 8

Conclusion

The main goal of this thesis was to propose a solution to manage multimedia contents in order to display them on multiple monitors and collect user metrics while the information was being displayed. The metrics are mainly attention and displayed emotions that people have while inspect the digital signage. Overall the requirements of the system were met, despite the lack of test subjects due to the pandemic and other factors.

The system was composed of four main components the Agent, the Media Content Management (MCM), the Statistics Dashboard, and the Mobile Application. The agent was composed by three elements, the display (monitor or TV), the Raspberry Pi, and the pi camera (figure 8.1). The MCM and the Statistics Dashboard were combined into the Django web app and the Mobile Application was composed of a React Native app running on a smartphone. To combine all the components it was created Hypertext Transfer Protocol (HTTP) requests and a RESTful API (REST API) for network communication, which lead to the need to create a hot-spot network will testing the system at Institute of Electronics and Informatics Engineering of Aveiro (IEETA), since both MCM and the Agent communicate through a different listening port (the first is the Django server port and the second the flask port) and the Eduroam network only allows the 80 port to be open for communication.

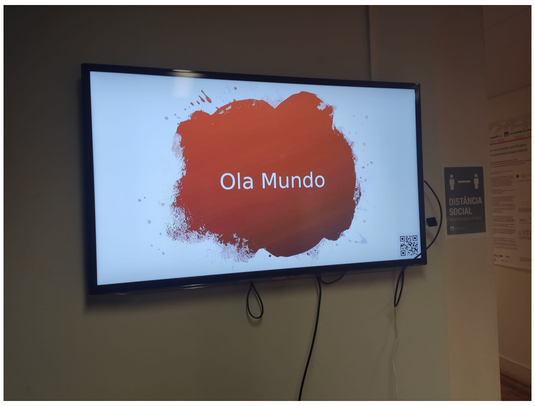


Figure 8.1: Digital Signage used for testing

8.1 Test scenarios and results

With the SARS-CoV-2 pandemic, health norms were introduced regarding the mandatory use of masks in academic community spaces. The emotional recognition and eye gaze detection algorithm developed by Daniel Canedo was not prepared for facial recognition with people using masks.

For security reasons and also due to limitations present in the system (which will be detailed in the next section), the project was tested in a smaller and more controlled environment. The digital signage was set up in a place with good lighting and the image capture rate was set to 1 picture per second.

8.1.1 Test case

Two people were asked to sit in front of the agent and for about three minutes to enjoy the atmosphere. A third person is invited to join the experiment later on. After some time, they were asked to walk in front of the agent while interacting with their smartphones.

Thus, the intention of this test case was to know if:

- The number of people was detected correctly
- How many faces were detected in the defined time period
- The average degree of attention
- The type of emotions demonstrated

8.1.2 Results

During the test case explained above, some metrics were measured and will be analyzed during this section.

The table 8.1 presents summary results obtained during the experiment.

Regarding the table above, all individuals were recognized by the agent. During the three minutes of experience were captured 67 faces with average attention of 36,641 % per face detected (frame).

Persons recognized	3
Faces detected	67
Average Attention	36,641 %

Table 8.1: Summary of the test case results

In terms of the emotions, the results were that in 53 faces the emotions were neutral (which represents 79% of the test subjects) and in the other 14 was happiness (21% of the test subjects), as illustrated in the pie chart in the figure 8.2.

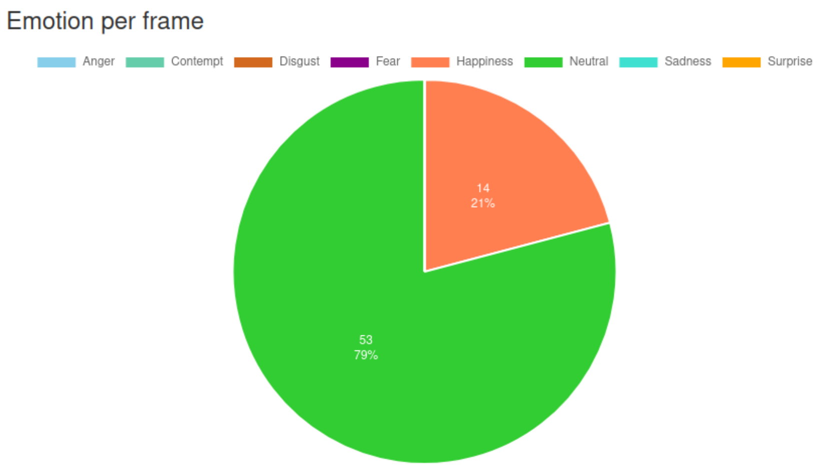


Figure 8.2: Emotions detected during the experiment

Another metric that this test focused on was attention by frame. This metric can be inspected in the graph 8.3. In the Y-axis is represented the value of attention. It differs in an range of 0 to 1. While in the X-axis represents each frame.

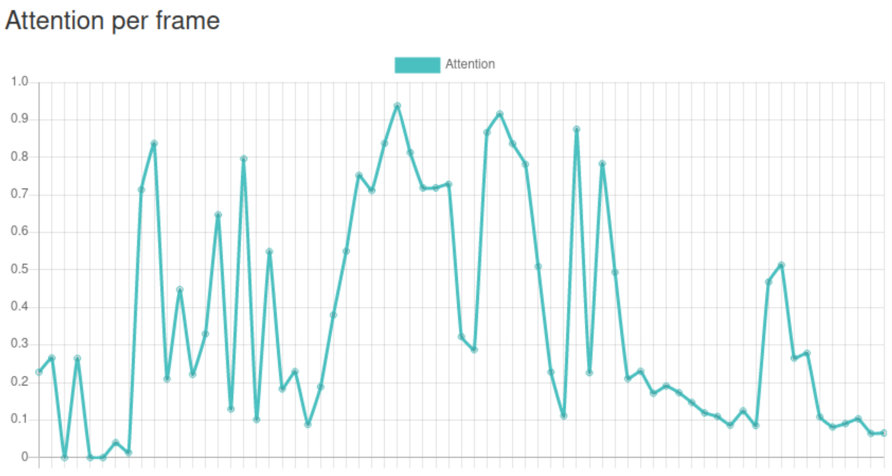


Figure 8.3: Attention paid during the experiment

It is possible to conclude that emotion is not a very relevant metric in the context of digital signage since people tend to just observe, not to display emotions. People only display emotions if the content exhibited has some impact, for example by exhibiting a dramatic video or a funny one.

But, attention is a suitable metric because it is possible to know the degree of attention given by the spectators, which allows knowing if the content was relevant and reached the target audience.

8.2 System limitations

Some limitations found during system development and initial testing were the following:

- Limitation of the emotion recognition model
- Use of facial masks
- Space lighting
- Agent performance

8.2.1 Limitation of emotion recognition model

During tests performed, the emotion detected most of the time was neutral. This is because the model used for recognition only could detect different emotions when strong expressions were given (figure 8.4), most of the time people would only perform subtle expressions. The emotion needed for the module to work properly required to be an extreme one, that in a real-world scenario it would be some limited.

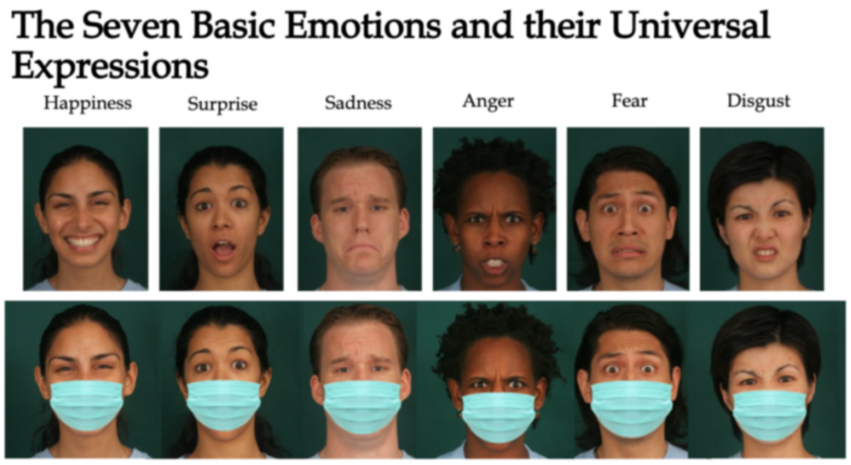


Figure 8.4: Emotions with and without facial masks[1]

[1]Face Masks Reduce Emotion Recognition Accuracy. [Online]. Available: `https://www.humintell.com/2021/07/face-masks-reduce-emotion-recognition-accuracy/`.

8.2.2 Use of facial mask

Due to the pandemic, the use of masks occurs on a daily basis and the system does not detect faces by the face-detection algorithm.

8.2.3 Space lighting

Another aspect to take into account during the development of the solution was the lighting of the room, more specific near the agent's camera. It was identified that in low and high lighting environments sometimes it is difficult for the face detection algorithm to detect them. So, due to this inconvenience, the system by now would not work in a real scenario, for example as street digital signage.

8.2.4 Processing media in the agent

Processing media files and creating videos from them was one of the key functionalities of the agent and to do so it used video/media processing software/codecs. Overall the libraries and commands used to do the processing were quite reliable and lightweight, which meant that the agent (Raspberry Pi) could use them without any effort. Although the use of FFmpeg on the agent side was a tremendous mistake in the media processing, as explained before, FFmpeg is hardware-intensive so the final part of the video processing (putting the QR code into all view) was very intensive for the agent, making it slower.

8.3 Future Work

This section, as the title suggests, will discuss and mention what the system could achieve and what could be done to do so. In the following bullets, it will be referred key aspects that could fix some limitations identified.

- Instead of adding the QR code in the agent, it could be added on the server side during the agent media retrieval phase. This would decrease the processing time in the agent because it wouldn't need to run FFmpeg and all downloaded media would already have the QR code embedded.
- Using layouts and other view templates, so that the admin could make more pleasant and appealing views more easily, for example by drag-drop, for the user to see.
- The mobile application could give more information to the user, for example, information about what is being played, all media that could be playable. And some extra features like preview and select the media that the user would like to spectate.
- Improve the module of emotion recognition so it would detect more natural expressions, and by doing so improve the overall user experience and the system.

- The Statistics Dashboard could be more content-oriented instead of view-oriented, this new approach would lead to a better understanding of what media had a more impact on the user, and by doing so it could lead to better targeting of this system usage, for example for marketing purpose.

CHAPTER 9

References

[1] N. Turov, N. Shilov, and N. Teslya, “Digital signage personalization through analysis of the visual information about viewers,” *Conference of Open Innovation Association, FRUCT*, vol. 2019-April, pp. 444–450, May 2019. DOI: `10.23919/FRUCT.2019.8711893`.

[2] E. Kim, H. J. Lee, D. H. Lee, U. Jang, H. S. Kim, K. S. Cho, and W. Ryu, “Efficient contents sharing between digital signage system and mobile terminals,” in *International Conference on Advanced Communication Technology, ICACT*, Jun. 2013, pp. 1002–1005. [Online]. Available: `https://ieeexplore.ieee.org/document/6488349`.

[3] R. Want and B. N. Schilit, “Interactive digital signage,” *Computer*, vol. 45, no. 5, pp. 21–24, 2012. DOI: `10.1109/MC.2012.169`.

[4] *For how long can data be kept and is it necessary to update it? | European Commission.* [Online]. Available: `https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/principles-gdpr/how-long-can-data-be-kept-and-it-necessary-update-it_en`.

[5] T. Ogi, Y. Tateyama, and Y. Matsuda, “Push type digital signage system that displays personalized information,” *Proceedings - 2014 International Conference on Network-Based Information Systems, NBiS 2014*, pp. 411–415, Jan. 2014. DOI: `10.1109/NBIS.2014.56`.

[6] J. S. Lee and K. Yoon, “The application of digital signage system using smart device,” *International Conference on Advanced Communication Technology, ICACT*, pp. 675–677, 2014. DOI: `10.1109/ICACT.2014.6779048`.

[7] *What is interactive digital signage?* [Online]. Available: `https://broadsign.com/blog/interactive-digital-signage/`.

[8] *Yodeck: Simple Cloud-Based Digital Signage Software - Free Plans & Players.* [Online]. Available: `https://www.yodeck.com/`.

[9] *The Professional & Affordable Screencloud Alternative – Yodeck.* [Online]. Available: `https://www.yodeck.com/screencloud-alternative/?gclid=Cj0KCQjwiNSLBhCPARIsAKNS4_drAQTBjZKdp4cr9n6CjuHoG25dT0tfqS2GbYZPQSlJ5hm4I69_7vgaAkpFEALw_wcB`.

[10] *30 Examples of Good Content for Digital Signage - ScreenCloud.* [Online]. Available: `https://screencloud.com/blog/good-content-digital-signage`.

[11] *NoviSign Digital Signage.* [Online]. Available: `https://www.novisign.com/`.

[12] *Home - OnSign TV - Digital Signage.* [Online]. Available: `https://onsign.tv/`.

[13] *OnSign TV Software - 2021 Reviews, Pricing & Demo.* [Online]. Available: https://www.softwareadvice.co.uk/software/86681/onsign-tv.

[14] A. A. Putri Ratna, P. Dewi Purnamasari, A. Shaugi, and M. Salman, "Analysis and comparison of MD5 and SHA-1 algorithm implementation in Simple-O authentication based security system," *2013 International Conference on Quality in Research, QiR 2013 - In Conjunction with ICCS 2013: The 2nd International Conference on Civic Space*, pp. 99–104, 2013. DOI: 10.1109/QIR.2013.6632545.

[15] D. D. Canedo, "Are you looking at me?: monitoring classrooms," 2017. [Online]. Available: http://hdl.handle.net/10773/23483.

[16] *Welcome to Flask — Flask Documentation (2.0.x).* [Online]. Available: https://flask.palletsprojects.com/en/2.0.x/.

[17] *What is Flask? - Flask: Develop Web Applications in Python.* [Online]. Available: https://www.educative.io/courses/flask-develop-web-applications-in-python/qZWAmEGDBkR.

[18] *The web framework for perfectionists with deadlines | Django.* [Online]. Available: https://www.djangoproject.com/.

[19] *What is Django, and why you should learn it? - DEV Community.* [Online]. Available: https://dev.to/snehal_02/why-what-is-django-and-why-you-should-learn-it-4845.

[20] *Advantages and Disadvantages of Django | Hacker Noon.* [Online]. Available: https://hackernoon.com/advantages-and-disadvantages-of-django-499b1e20a2c5.

[21] *Databases | Django documentation | Django.* [Online]. Available: https://docs.djangoproject.com/en/3.2/ref/databases/.

[22] *MySQL :: Why MySQL?* [Online]. Available: https://www.mysql.com/why-mysql/.

[23] *SQLite Home Page.* [Online]. Available: https://www.sqlite.org/index.html.

[24] *Sqlite vs. MySQL vs. PostgreSQL: A Comparison of Relational Databases | Logz.io.* [Online]. Available: https://logz.io/blog/relational-database-comparison/.

[25] S. T. Bhosale, M. T. Patil, and M. P. Patil, "SQLite International Journal of Computer Science and Mobile Computing SQLite: Light Database System," *International Journal of Computer Science and Mobile Computing*, vol. 4, pp. 882–885, 2015. [Online]. Available: https://www.researchgate.net/publication/279621848.

[26] S. Bhosale, M. Patil, M. P. I. J. C. S. M. Comput, and u. 2015, "Sqlite: Light database system," *researchgate.net*, vol. 4, pp. 882–885, 2015. [Online]. Available: https://www.researchgate.net/profile/Satish-Bhosale-2/publication/279621848_SQLite/links/5597686708ae21086d221523/SQLite.pdf.

[27] *SQLite Database Speed Comparison.* [Online]. Available: https://www.sqlite.org/speed.html.

[28] *Chart.js | Open source HTML5 Charts for your website.* [Online]. Available: https://www.chartjs.org/.

[29] *D3.js - Data-Driven Documents.* [Online]. Available: https://d3js.org/.

[30] W. Zheng and Y. Sherif, "Visualization Linter," [Online]. Available: https://www.cs.ubc.ca/~tmm/courses/547-19/projects/wei-youssef/finalreport.pdf.

[31] *Buy a Raspberry Pi 3 Model B+ – Raspberry Pi.* [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.

[32] *Buy a Raspberry Pi Camera Module 2 – Raspberry Pi.* [Online]. Available: https://www.raspberrypi.org/products/camera-module-v2/.

[33] *About - OpenCV.* [Online]. Available: https://opencv.org/about/.

[34] *About FFmpeg.* [Online]. Available: https://www.ffmpeg.org/about.html.

[35] *FFmpeg Threads Command: How it Affects Quality and Performance - Streaming Learning Center.* [Online]. Available: https://streaminglearningcenter.com/blogs/ffmpeg-command-threads-how-it-affects-quality-and-performance.html.

[36] A. M. A. A S Elliethy and H. A. A. H. Y. El-Arsh, "Performance comparison among popular implementations of H.264 encoders," *IOP Conference Series: Materials Science and Engineering*, vol. 1172, no. 1, p. 012 036, Aug. 2021. DOI: 10.1088/1757-899x/1172/1/012036.

[37] I. Malavolta, "Beyond native apps: web technologies to the rescue!" In *Proceedings of the 1st International Workshop on Mobile Development*, New York, NY, USA: ACM, Oct. 2016, pp. 1–2, ISBN: 9781450346436. DOI: 10.1145/3001854.3001863.

[38] *5 Key Benefits of Native Mobile App Development | Clearbridge Mobile.* [Online]. Available: https://clearbridgemobile.com/benefits-of-native-mobile-app-development/.

[39] *React Native · Learn once, write anywhere.* [Online]. Available: https://reactnative.dev/.

[40] W. Danielsson, "A comparison between native Android and React Native," *React Native application development*, p. 70, 2016. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-131645.

[41] B. Eisenman, *Learning React Native: Building Native Mobile Apps with JavaScript.* 2016, p. 432, ISBN: 9781491929001.

[42] *5 key advantages React Native | icapps blog.* [Online]. Available: https://icapps.com/blog/5-advantages-react-native.

[43] *Progressive Web Apps (PWAs): o que é e os melhores exemplos.* [Online]. Available: https://rockcontent.com/br/blog/progressive-web-apps/.

[44] *Should You Consider Investing In A Progressive Web App? | September 2021.* [Online]. Available: https://themanifest.com/app-development/blog/should-you-invest-in-progressive-web-app.

[45] W. Danielsson, "React Native application development : A comparison between native Android and React Native," 2016. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-131645.

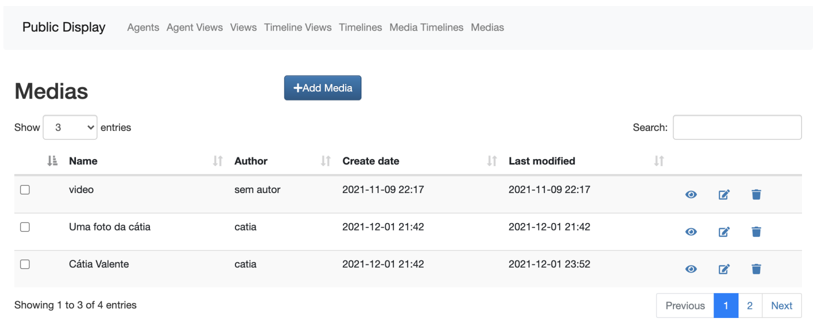
[46] W. Jobe, "Native Apps Vs. Mobile Web Apps," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 7, no. 4, p. 27, Oct. 2013. DOI: 10.3991/ijim.v7i4.3226.

[47] *PWA vs React Native: A Detailed Look - SimiCart.* [Online]. Available: https://www.simicart.com/blog/pwa-vs-react-native/.

APPENDIX A

Media Content Management Interfaces

A.1 Medias


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

Medias

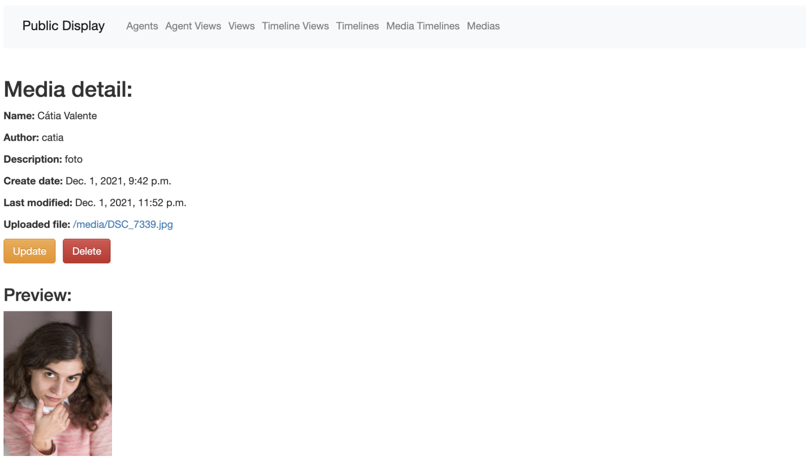
+Add Media

Show 3 entries Search:

	Name	Author	Create date	Last modified	
☐	video	sem autor	2021-11-09 22:17	2021-11-09 22:17	
☐	Uma foto da cátia	catia	2021-12-01 21:42	2021-12-01 21:42	
☐	Cátia Valente	catia	2021-12-01 21:42	2021-12-01 23:52	

Showing 1 to 3 of 4 entries

Figure A.1: Medias list


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

Media detail:

Name: Cátia Valente

Author: catia

Description: foto

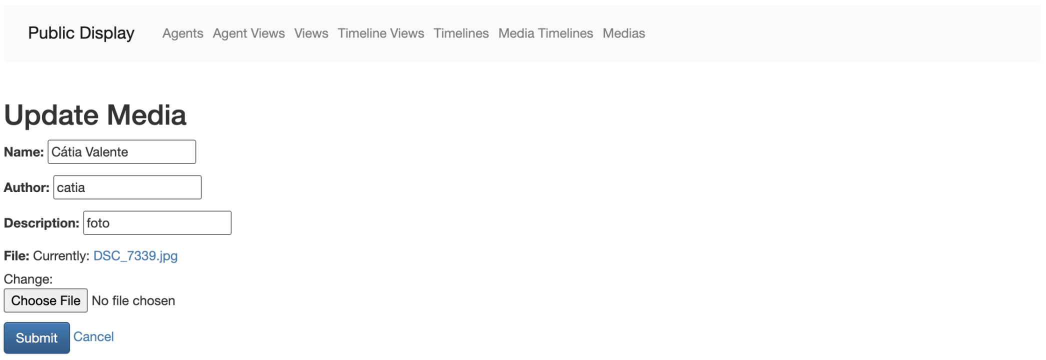
Create date: Dec. 1, 2021, 9:42 p.m.

Last modified: Dec. 1, 2021, 11:52 p.m.

Uploaded file: /media/DSC_7339.jpg

Preview:

Figure A.2: Media detail


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

Update Media

Name: Cátia Valente

Author: catia

Description: foto

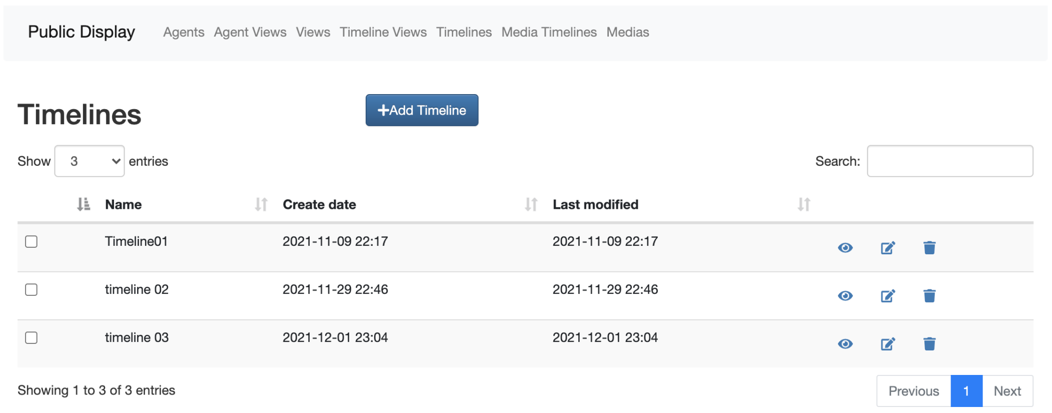
File: Currently: DSC_7339.jpg

Change:

Choose File No file chosen

Figure A.3: Media update

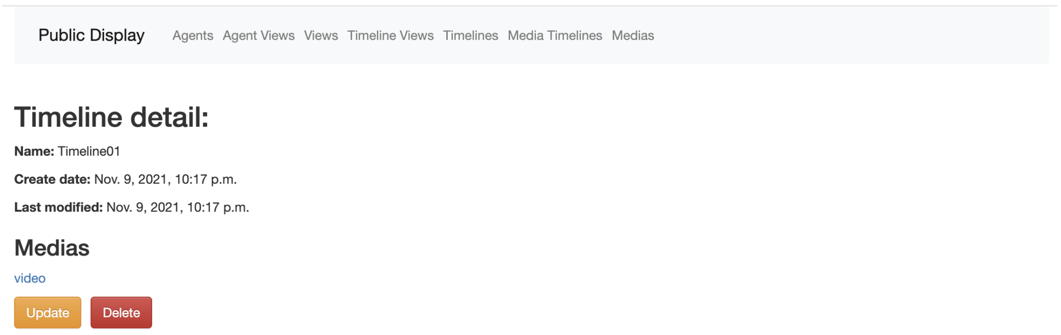
A.2 Timelines


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias
Timelines
+Add Timeline
Show 3 entries
Search:

	Name	Create date	Last modified
☐	Timeline01	2021-11-09 22:17	2021-11-09 22:17
☐	timeline 02	2021-11-29 22:46	2021-11-29 22:46
☐	timeline 03	2021-12-01 23:04	2021-12-01 23:04

Showing 1 to 3 of 3 entries

Figure A.4: Timelines list


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

Timeline detail:

Name: Timeline01

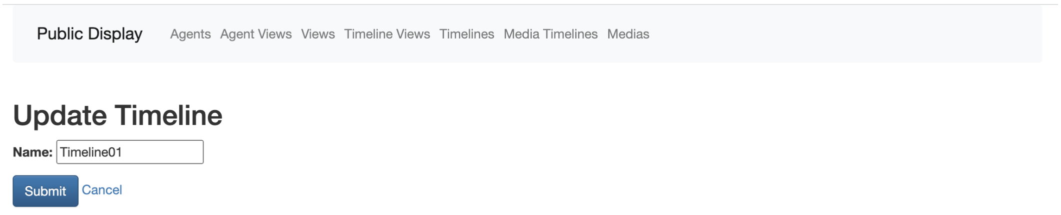
Create date: Nov. 9, 2021, 10:17 p.m.

Last modified: Nov. 9, 2021, 10:17 p.m.

Medias

video

Figure A.5: Timeline detail

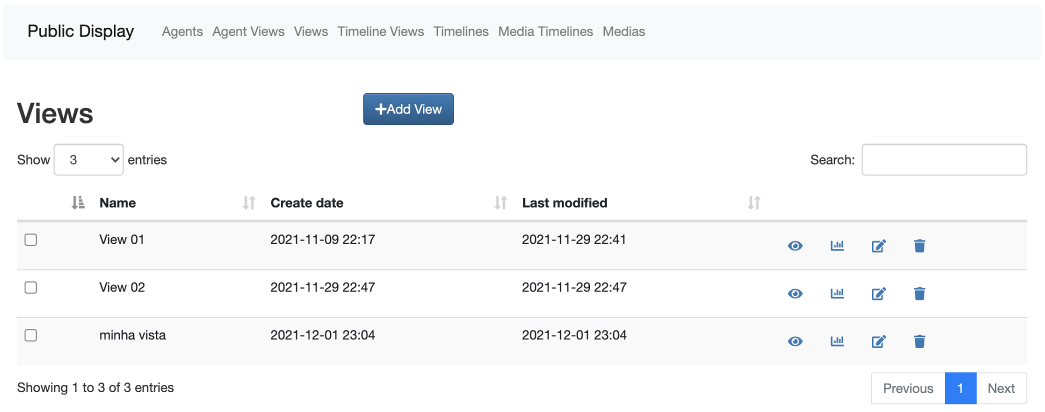

Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

Update Timeline

Name: Timeline01

Figure A.6: Timeline update

A.3 Views


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias
+Add View
Search:
Previous
1
Next

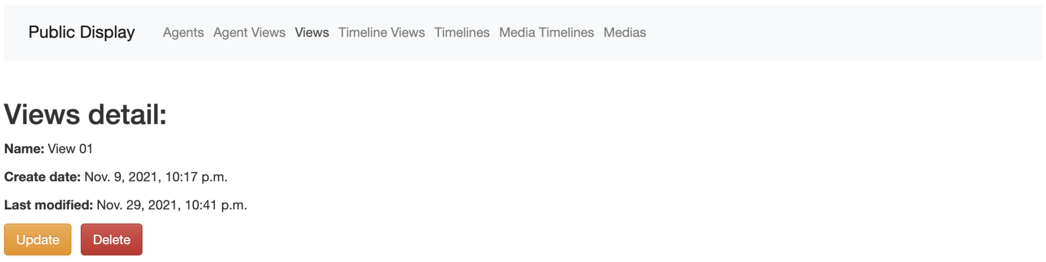
Views

Show 3 entries

	Name	Create date	Last modified
☐	View 01	2021-11-09 22:17	2021-11-29 22:41
☐	View 02	2021-11-29 22:47	2021-11-29 22:47
☐	minha vista	2021-12-01 23:04	2021-12-01 23:04

Showing 1 to 3 of 3 entries

Figure A.7: Views list


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

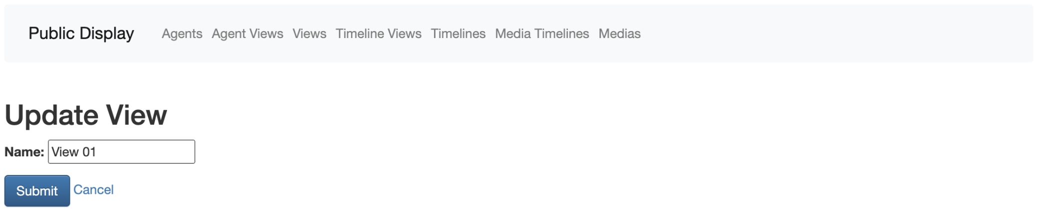
Views detail:

Name: View 01

Create date: Nov. 9, 2021, 10:17 p.m.

Last modified: Nov. 29, 2021, 10:41 p.m.

Figure A.8: View detail

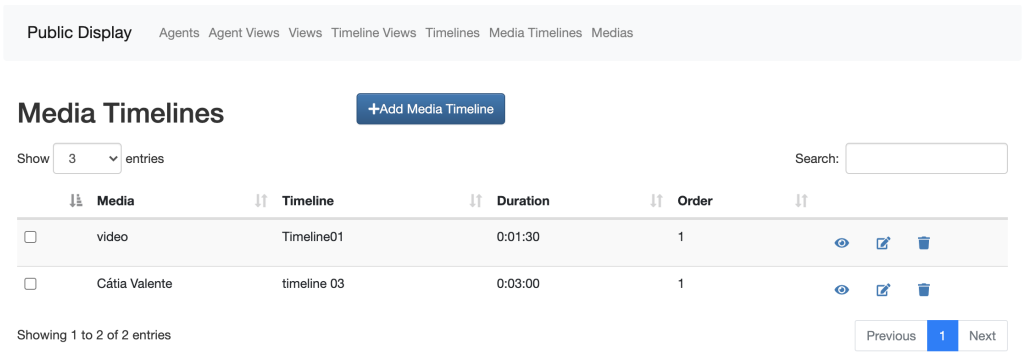

Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

Update View

Name: View 01

Figure A.9: View update

A.4 Media Timelines



Media Timelines

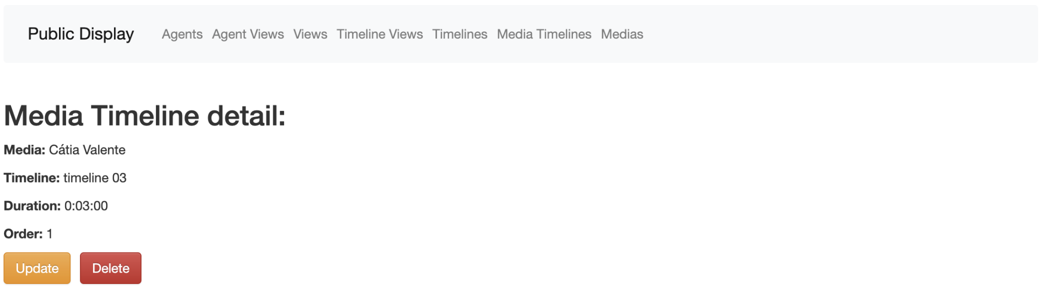
+Add Media Timeline

Show 3 entries Search:

	Media	Timeline	Duration	Order
☐	video	Timeline01	0:01:30	1
☐	Cátia Valente	timeline 03	0:03:00	1

Showing 1 to 2 of 2 entries

Figure A.10: Media Timelines list



Media Timeline detail:

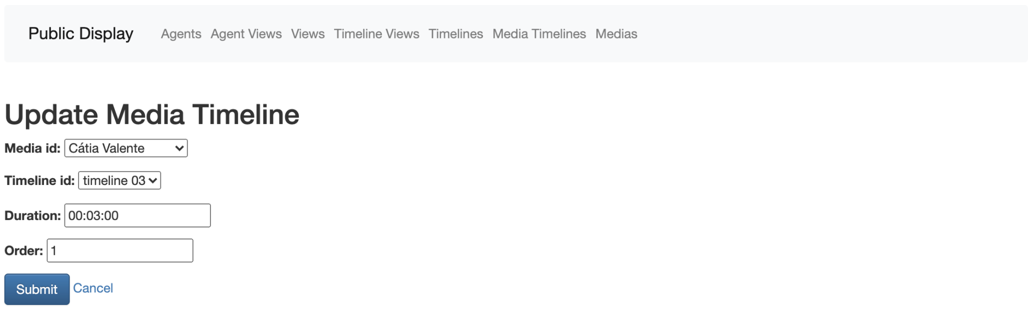
Media: Cátia Valente

Timeline: timeline 03

Duration: 0:03:00

Order: 1

Figure A.11: Media Timeline detail



Update Media Timeline

Media id: Cátia Valente

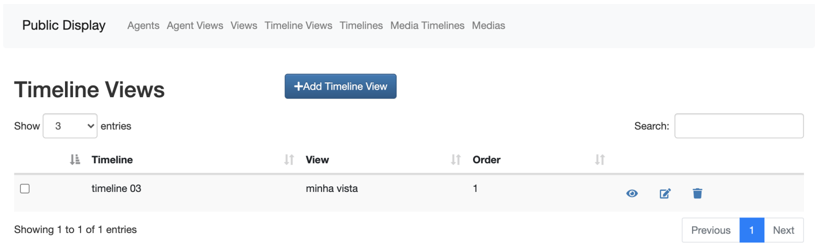
Timeline id: timeline 03

Duration: 00:03:00

Order: 1

Figure A.12: Media Timeline update

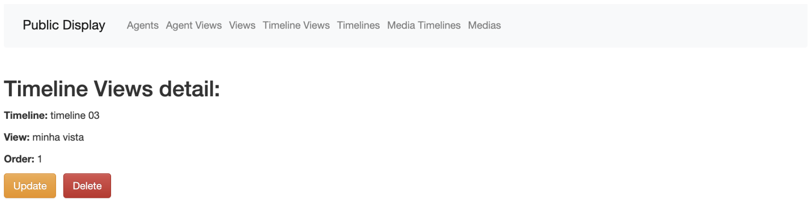
A.5 Timeline Views


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias
Timeline Views
+Add Timeline View
Show 3 entries
Search:

Timeline	View	Order
timeline 03	minha vista	1

Showing 1 to 1 of 1 entries

Figure A.13: Timeline Views list


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

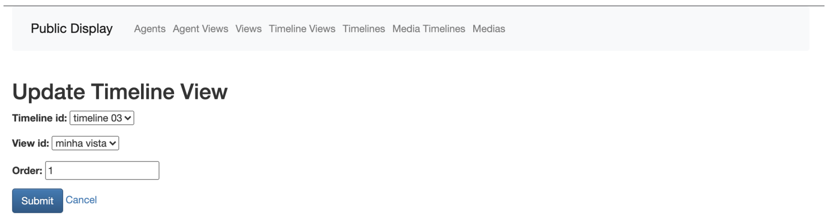
Timeline Views detail:

Timeline: timeline 03

View: minha vista

Order: 1

Figure A.14: Timeline View detail


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

Update Timeline View

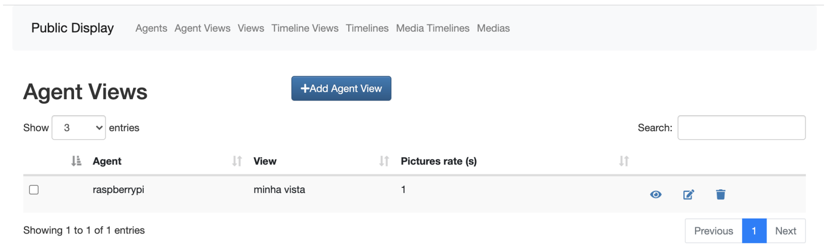
Timeline id: timeline 03

View id: minha vista

Order: 1

Figure A.15: Timeline View update

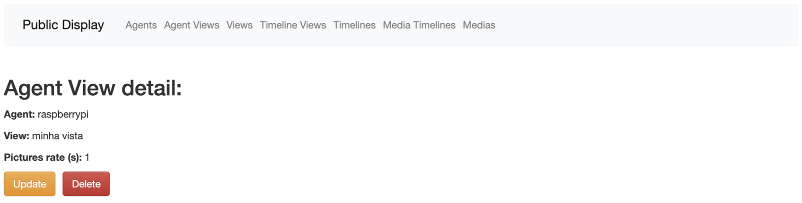
A.6 Agent Views


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias
Agent Views
+Add Agent View
Show 3 entries
Search:

Agent	View	Pictures rate (s)
raspberrypi	minha vista	1

Showing 1 to 1 of 1 entries

Figure A.16: Agent Views list


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

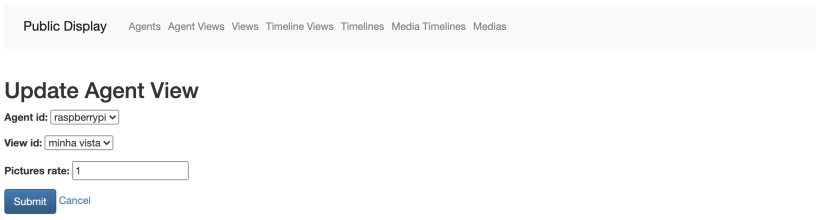
Agent View detail:

Agent: raspberrypi

View: minha vista

Pictures rate (s): 1

Figure A.17: Agent View detail


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

Update Agent View

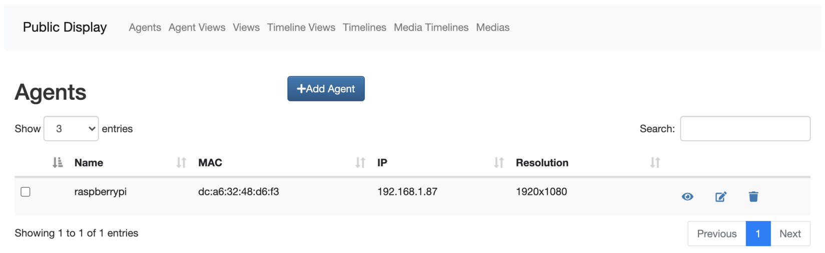
Agent id: raspberrypi

View id: minha vista

Pictures rate: 1

Figure A.18: Agent View update

A.7 Agents

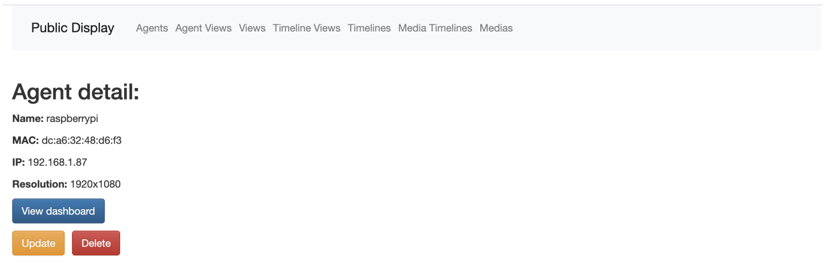

Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias
+Add Agent
Show 3 entries
Search:

Agents

	Name	MAC	IP	Resolution
☐	raspberrypi	dc:a6:32:48:d6:f3	192.168.1.87	1920x1080

Showing 1 to 1 of 1 entries

Figure A.19: Agents list


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

Agent detail:

Name: raspberrypi

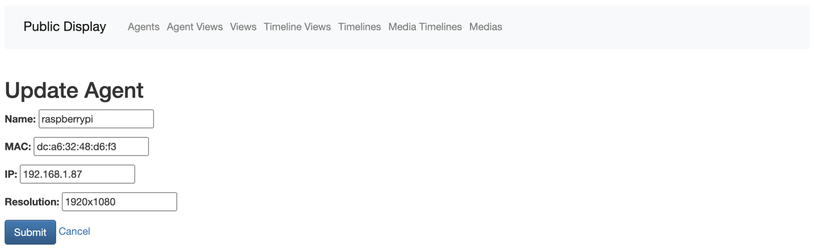
MAC: dc:a6:32:48:d6:f3

IP: 192.168.1.87

Resolution: 1920x1080

View dashboard
Update
Delete

Figure A.20: Agent detail


Public Display
Agents Agent Views Views Timeline Views Timelines Media Timelines Medias

Update Agent

Name: raspberrypi

MAC: dc:a6:32:48:d6:f3

IP: 192.168.1.87

Resolution: 1920x1080

Figure A.21: Agent update