Universidade de Aveiro
2021

Bruno Filipe
Maia Henriques

Gestão de Conteúdos Digitais em Múltiplos Monitores

Management of Digital Contents in Multiple Displays

Universidade de Aveiro
2021

**Bruno Filipe
Maia Henriques**

Gestão de Conteúdos Digitais em Múltiplos Monitores

Management of Digital Contents in Multiple Displays

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 Doutor António José Ribeiro Neves, Professor professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho à minha família e à minha namorada por todo o apoio durante o desenvolvimento desta dissertação.

agradecimentos / acknowledgements

Quero agradecer especialmente ao Professor Doutor António Neves e ao aluno de Doutoramento Daniel Canedo por todo o incentivo, disponibilidade e orientação que conduziram à realização deste trabalho. Agradeço também aos meus amigos João Quintanilha e Bernardo Ferreira que me acompanharam neste percurso, proporcionando memórias inesquéciveis e amizades que levarei para vida. Um agradecimento à minha família pela oportunidade de obter este grau académico e confiança depositada nas minhas capacidades para o atingir. E por fim, agradecer à minha namorada Raquel Albino pelo apoio incondicional que sem dúvida contribuiu para a conclusão desta dissertação.

Palavras Chave

Sinalização Digital, Visão de Computador, Extração de Dados, Deteção de Faces, Reconhecimento Facial, Extração de Emoções.

Resumo

Com a utilização generalizada de sistemas de disseminação de conteúdos digitais, surge a oportunidade de implementar soluções capazes de avaliar a reacção do público. Esta dissertação reflete a implementação de uma dessas soluções. Para isso, o desenvolvimento passou pela adaptação de um sistema de sinalização digital previamente funcional. Neste sentido, aos terminais de exposição de conteúdos, foram emparelhadas câmaras digitais de modo a permitir a captação de informação da área à frente destes. Com recurso a tecnologias de visão de computador, os terminais fazem, em tempo real, deteção de pessoas que apareçam no campo de visão das câmaras, sendo esta informação comunicada a um servidor para extração de dados. No servidor, são utilizados métodos para realização de reconhecimento de faces e emoções, e também é feita extração de dados indicadores da posição da cabeça, o que permite o cálculo de um coeficiente de atenção. Os dados são guardados numa base de dados relacional e podem ser consultados através de uma plataforma web, onde são apresentados associados aos contéudos correspondentes ao momento de captação e extração destes. Esta solução, permite, assim, a avaliação do impacto dos conteúdos digitais apresentados pelo sistema.

Keywords

Digital Signage, Computer Vision, Data Extraction, Face Detection, Face Recognition, Emotion Extraction.

Abstract

With the generalized use of systems for digital contents dissemination arises the opportunity for implementing solutions capable of evaluating audience reaction. This dissertation reflects the implementation of one of those solutions. To this end, the development involved adapting a previously functional digital signage system. In this sense, digital cameras were paired to the content display terminals in order to capture information from the area in front of them. Using computer vision technologies, the terminals detect, in real time, people who appear in the cameras' field of view, and this information is communicated to a server for data extraction. On the server, methods are used to perform face and emotion recognition, and also to extract data indicating the position of the head, which allows the calculation of an attention coefficient. The data is stored in a relational database, and can be consulted through a web platform, where they are presented associated with the contents corresponding to the moment of their capture and extraction. This solution thus allows the evaluation of the impact of the digital contents presented by the system.

Contents

List of Figures

List of Tables

Glossary

ACL	Access Control List
AVA	Anonymous Viewer Analytics
AVI	Audio Video Interleave
CSRF	Cross-site request forgery
CSS	Cascading Style Sheets
DOOH	Digital-Out-of-Home
HDMI	High-Definition Multimedia Interface
HOG	Histogram of Oriented Gradients
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IAF	Intelligent Advertising Framework
IEETA	Instituto de Engenharia Electrónica e Telemática de Aveiro
I/O	Input/Output
IoT	Internet of Things
JPEG	Joint Photographic Experts Group
MAC	Media Access Control
MP4	MPEG-4
MTV	Model-Template-View
MVC	Model-View-Controller
ONNX	Open Neural Network Exchange
OpenCV	Open Source Computer Vision
ORM	Object-Relational mapper
PDF	Portable Document Format
PIL	Python Imaging Library
PNG	Portable Network Graphics
PPT	PowerPoint
ResNet	Residual Neural Network
SQL	Structured Query Language
sTIC	Serviços de Tecnologias de Informação e Comunicação
VPN	Virtual Private Network
URL	Uniform Resource Locator

CHAPTER 1

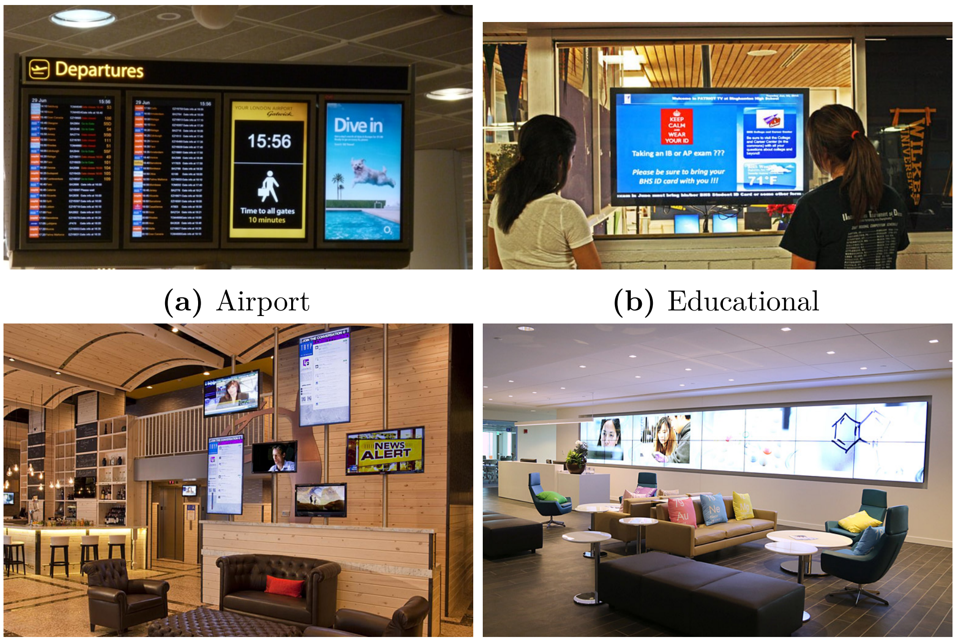
Introduction

Business and technology have always been tied together. Technology empowers business and business drives technological evolution. Digital Signage has improved businesses advertising strategies, since manufacturers and retailers have found that in-store marketing can affect shopper behavior powerfully [1]. Coming from paper posters and billboards, Digital Signage revolutionized the way businesses market their products.

Digital Signage is the concept of remotely managed digital displays, typically tied with sales, marketing and advertising in the form of Digital-Out-of-Home (DOOH). Any number of these displays can then be centrally managed and addressed individually for content dissemination [2]. These systems are also popular in other environments such as corporate, educational and hospitality, with a bigger focus in creating a more involving working environment, spreading information and setting the mood. Applications of this technology can now be found in most public places like shopping malls, airports, stations, retail stores, universities and hotels as illustrated in Figure 1.1.

Assembling a Digital Signage system has become more affordable, since the cost of the necessary equipment has fallen, such as LCD screens, servers and computers used to administer the content [4]. This leaves room to invest in methods for improving the reachability and effectiveness of these systems, which can be achieved with the introduction of Computer Vision technology [5].

Computer Vision is the field of computer science that focuses on replicating parts of the complexity of the human vision system and enabling computers to identify and process objects in images and videos in the same way that humans do [6]. With methods for face detection, face and emotions recognition and head pose tracking, it is possible to gather relevant information about the audience and even enable non-contact human computer interaction.


Departures
15:56
Time to all gates 10 minutes
Dive in
KEEP CALM AND WEAR YOUR ID
Taking an IB or AP exam ???
Please be sure to bring your BHS ID card with you !!!
NEWS ALERT

(a) Airport **(b)** Educational

(c) Hospitality **(d)** Corporate

Figure 1.1: Applications of Digital Signage solutions [3].

1.1 Motivation

Having information delivered in a digital way offers the possibility of tracking costumers exposed to specific contents [7]. Reporting the number of viewers in front of a display, when a content item is shown, is interesting in terms of adverstising penetration and impression counting [8]. Understading what incentivizes people to engage more with Digital Signage terminals, not only in real time, but also in later date, offers the means to reach target audiences in more effective ways, as well as to establish casual effects, as for example, whether that increased interaction results in an increase of sales. With the introduction of Machine Learning techniques on multi-view streaming systems it is possible to deliver dynamic content capabilities from accurate head-tracking data [9]. That being said, the motivation behind this dissertation was to develop a solution of Digital Signage that combines the display panels with digital cameras, making the solution a “two-way” system that has output aswell as input, enabling the delivery of digital contents using this human computer interaction approach.

1.1.1 Similar Systems

Some scientific work has been developed regarding this topic. In [10], the authors proposed an Intelligent Advertising Framework (IAF), which involves integration of Anonymous Viewer Analytics (AVA) and data mining technologies into a Digital Signage system in order to achieve targeted and interactive adverstising. Also, in [11], it was proposed a system that attempts to understand viewer’s involuntary behaviors in order to adapt the displayed contents by the Digital Signage system.

Some systems also served as inspiration to the implementation of this thesis's solution, and are presented in this section.

Signbox

Signbox [12] is a leading Digital Signage software company whose technology powers distributed media applications around the globe. It's product suite contains an audience measurement and analysis tool, which can be used with the Signbox Digital Signage system, called *signEye* [13].

The *signEye* system uses a digital camera in conjunction with the latest in facial recognition technology, to identify audience metrics that occur in the field of vision. The digital camera is embedded in the screen frame and will normally be located above or below a Digital Signage screen, so the recorded metrics are directly related to the content being shown on the digital signage screen in real time.

The system is capable of recording metrics such as age, gender, emotion, number of people that looked at the screen and the length of time they spent doing so.

signEye is the most similiar system compared the project of this dissertation. In Figure 1.2 is presented an illustration of the *signEye* module from Signbox.

Figure 1.2: Signbox's *signEye* module.

Seemetrix

Seemetrix [14] is an anonymous audience analytics solution consisting of client applications and a web dashboard. The application determines and tracks every person in front of the device estimating their gender, age group, emotions and dwell time.

The collected data is then sent to the web dashboard for analysis. The dashboard allows to monitor and control devices running Seemetrix application and information visualization regarding the obtained metrics. An illustration of the application running is provided in Figure 1.3.

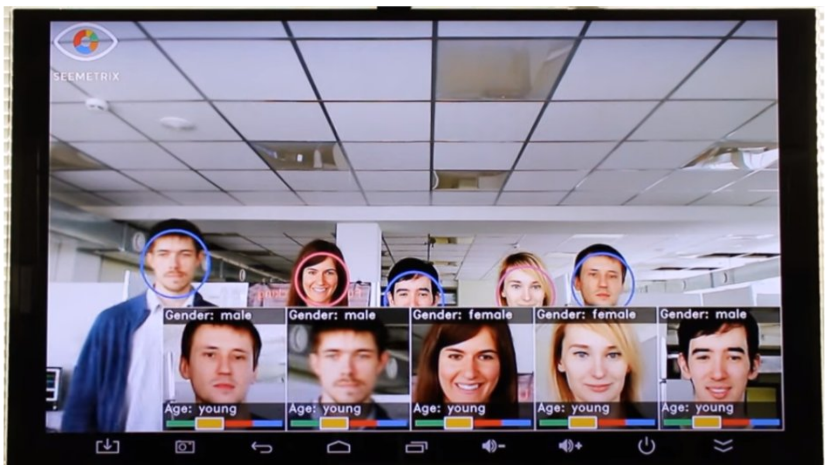


Figure 1.3: Seemetrix application for audience measurement.

1.2 Objectives

The objective of this dissertation concerned the development and integration of real-time audience assessment features into a Digital Signage system. This involved adding functionalities to several components of the system aimed at content adaptation according to audience behaviour. To better understand these developments, three areas of interest are defined below:

- Data Collection;
- Data Extraction;
- Data Mining.

For data collection, a functionality for continuous real-time face detection was implemented in the Content Display Agents. On the server side, processes for data extraction and mining were introduced. These processes contain methods for extracting data in order to perform face and emotions recognition and head pose estimation. Moreover, manipulation of this data aims to provide users of the system with information that facilitates an understanding of the impact of the digital contents being dessiminated by the system.

1.3 Dissertation Structure

This dissertation is organized in seven chapters with the present one, Introduction, as the first. The ones remaining are the following:

- **Chapter 2** - *Architecture*: definition of the architectural elements;
- **Chapter 3** - *Content Management*: discussion about the image processing behind content management;
- **Chapter 4** - *Content Display Agent*: presentation of the software solution of the content display and data collection agent;
- **Chapter 5** - *Data Extraction and Mining*: description of the processes for data extraction and mining;
- **Chapter 6** - *Results*: discussion of results obtained during tests of the system;
- **Chapter 7** - *Conclusions*: summary of the dissertation outcome.

CHAPTER 2

Architecture

The Digital Signage solution to serve as the starting point of this project already had a well-defined architecture that comprised three main groups: Web Server, Content Management Dashboard and Content Display Agents. This previous work was done by a colleague for his own dissertation [15].

This chapter provides insights on this architecture, going through all the components of the system, contextualizing their functionality. To begin, the system architecture is illustrated in Figure 2.1. Sections 2.1, 2.2 and 2.3 present the dependencies of the Web Server, the Content Management Dashboard and the Content Display Agents, respectively.

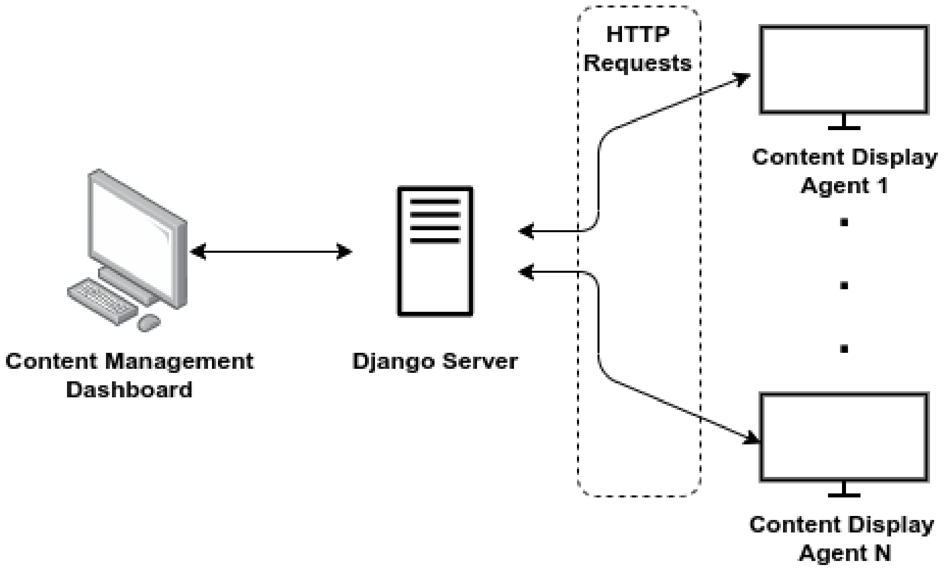


Figure 2.1: Architectural diagram of the system [15].

2.1 Web Server

As the central component of the system, the Web Server was implemented using Django [16], a high-level Python Web framework that encourages rapid development and clean, pragmatic design, allowing for fast application development from concept to completion, assuring security principles and providing quick and flexible scalability . This framework follows the philosophy of “batteries-included”, which means the inclusion of functionality for authentication, url routing, template engine, Object-Relational mapper (ORM) and database schema migrations [17].

The framework embraces the Model-View-Controller (MVC) design pattern. This entails separation of data presentation from user interactions logic handling. This being said, it implements its own logic under the MVC concept, since it considers handling the Controller a part of the pattern itself, thus making Django, mostly referred to as, a Model-Template-View (MTV) framework [18].

It also has large community support and provides built-in administration GUI that gives control over users authentication and database models and finally, facilitates end-to-end application development since it includes a lightweight web server, making it a great tool for full-stack software development.

2.1.1 Content Transformation

The system was designed to support three types of files and a couple of formats for each one. For images, the options are Joint Photographic Experts Group (JPEG) and Portable Network Graphics (PNG). For video, Audio Video Interleave (AVI) and MPEG-4 (MP4) are used. Lastly, for presentation files, Portable Document Format (PDF) and PowerPoint (PPT) are supported.

In order to achieve the necessary transformations of the multimedia contents uploaded to and manipulated in the system, several python libraries are used. These are listed in Table 2.1 coupled with a brief description for each one.

Library	Description
imghdr [19]	Determines the type of image contained in a file or byte stream. Used to validate image files extensions on upload (.JPEG or .PNG)
PyPDF2 [20]	Designed to operate over PDF files. Used to validate PDF files extensions on upload.
pdf2image [21]	Reads a PDF file and converts it into a PIL Image object. Used for PDF files conversion to images.
python-pptx [22]	Handler for Microsoft PowerPoint files that allows for creation and modification of presentations. Used for validation of uploaded PPT files.
ffmpy [23]	Command line wrapper for FFmpeg. Uses Python subprocess [24] module for command execution.

Table 2.1: Python libraries used in the server side [15].

FFmpeg

FFmpeg [25] is a multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play a wide variety of video and audio file formats. It allows a user to operate on audiovisual files directly [26]. This framework is a key dependency of this project since, when paired with the other libraries of the system, it fulfills its needs for content manipulation, be it, format transcoding, video editing and scaling among other features.

2.1.2 Data Extraction

According to the objectives of this dissertation, defined in Section 1.2, in the Web Server, were introduced data extraction capabilities concerning three main areas. The first one, face recognition, is the process of identifying and verifying the identity of individuals through photos or video. Algorithms extract specific and distinctive details about people's faces and convert them to mathematical representation to perform comparisons with other faces on record [27]. No image of the audience is saved, only anonymous descriptors that allow to differentiate detected subjects. The second one, emotion recognition, is the process of mapping facial expressions to identify emotions on a human face [28]. And the last one, head pose estimation, relates to the interpretation of facial landmarks in order to estimate the direction the head is oriented to [29].

In Chapter 5, it will be presented the implemented computer vision module that contains the solution associated with these capabilities, and in this Section, are listed three libraries used to fulfill that solution.

Dlib

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems [30]. Although it is written in C++ it has great compatibility with Python. Like OpenCV, it has a multitude of algorithms that bring capabilities for several fields. The one of most interest to this project is image processing, since Dlib has high performance face recognition features.

ONNX Runtime

ONNX Runtime is an open source project that is designed to accelerate machine learning across a wide range of frameworks, operating systems, and hardware platforms. It enables acceleration of machine learning inferencing across all of your deployment targets using a single set of API [31]. It brings this project the capability to identify emotions through facial expressions.

OpenCV

Open Source Computer Vision (OpenCV) is an open source computer vision and machine learning software library [32]. This library's algorithms are used to develop real-time computer vision applications, concept in which lie different fields. For this project, the capabilities associated with the field of image processing are once again the ones used the most, making use of methods for estimation of subjects head pose.

2.1.3 Database

With the need to store data associated with user information and permissions, multimedia contents and processed information derived from the collected data a database was prepared. Django officially supports several, such as PostgreSQL, MariaDB, MySQL, Oracle and SQLite [33]. In this project the last one, SQLite3, is used. This database is very popular and implements small, fast, self-contained, high-reliability, full-featured, Structured Query Language (SQL) database engine [34]. Another important advantage is that the database information is stored in a file, giving the system great portability.

2.1.4 Data Mining

As important as the processes for data collection and extraction was the process of data mining, since the aim of this solution was to provide users with information related to contents performance and impact on the audience that interacts with the Content Display Agents. This entails manipulations and calculations performed on the collected and extracted data.

Matplotlib

Matplotlib [35] is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is easy to use and makes possible the display of charts which help understand and interpret the data generated by the system.

2.2 Content Management Dashboard

This component is the access point for users to control and interact with the system. The webpage is served by Django and accomodates a simple interface that hosts a good range of user interactions. These range from user management, content upload and manipulation, real-time deployment and information visualization related to the generated data, firstly, from the cameras attached to each of the registered Content Display Agents and secondly, manipulated in the Web Server.

2.3 Content Display Agents

As the last component of the system we have the Content Display Agents. These are autonomous operating agents which are made up of a Raspberry Pi [36], coupled with a Pi Camera [37], connected to a display through High-Definition Multimedia Interface (HDMI). These agents act as the interaction point for subjects and display the multimedia contents generated and disseminated through the Content Management Dashboard, mentioned in Section 2.2. Along side the display process runs another process that continuously captures images of the area in front of the displays and detects subjects that may be engadging with the contents. More details on the software solution running on the Content Display Agent will be presented in Chapter 4.

The case study of this dissertation was to use the Digital Signage system at Instituto de Engenharia Electrónica e Telemática de Aveiro (IEETA) with the goal of disseminating science and good information. A deployment scenario example of the Content Display Agent, using a display in the insitute, is shown in Figure 2.2. In Chapter 6 will be presented the experimental results for this case study.

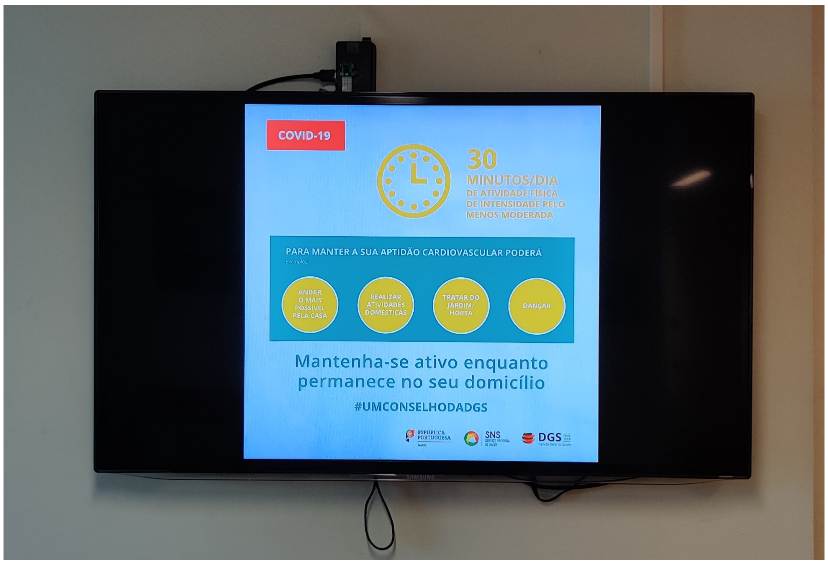

COVID-19
30
MINUTOS/DIA
DE ATIVIDADE FÍSICA DE INTENSIDADE PELO MENOS MODERADA
PARA MANTER A SUA APTIDÃO CARDIOVASCULAR PODERÁ
ANDAR O MAIS POSSÍVEL PELA CASA
REALIZAR ATIVIDADES DOMÉSTICAS
TRATAR DO JARDIM/HORTA
DANÇAR
Mantenha-se ativo enquanto permanece no seu domicílio
#UMCONSELHODADGS
SNS
DGS

Figure 2.2: Content Display Agent setup at one of IEETA's displays.

2.3.1 Hardware

Raspberry Pi, a UK based foundation, provides affordable single board computers in order to enable people all over the world to experience computing and digital technologies [38].

The use of these computers allowed for easy deployment and setup of the Content Display Agents. Raspberry Pi have shown that cheap, yet serviceable computer boards are the perfect platform for interfacing with many different devices making it very suitable for applications in Internet of Things (IoT) context [39].

Raspberry Pi 3 Model B+

For this project, version 3 of the Raspberry Pi single board computer was used, as illustrated in Figure 2.3. With sufficient processing power for the Python programme it needs to run, a chip for wireless LAN connections and varied Input/Output (I/O), including full size HDMI, DC power input and CSI camera port, allowing for a connection to the Raspberry Pi camera, thus making it a good fit to act as an agent of the system.

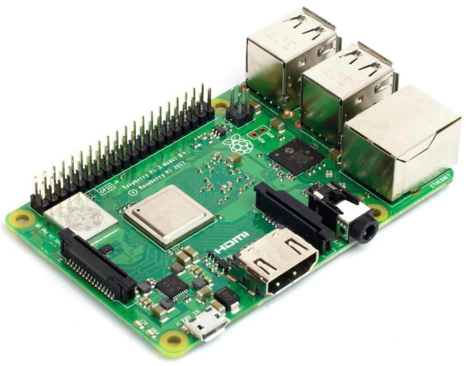


Figure 2.3: Raspberry Pi single board computer [40].

Raspberry Pi Camera Module V2

The camera module is easilly attachable and has an 8 megapixel sensor. The camera specifications provided a capable tool for this project, allowing for capture of high definition video and photographs. In Figure 2.4 is possible to see a captured image.

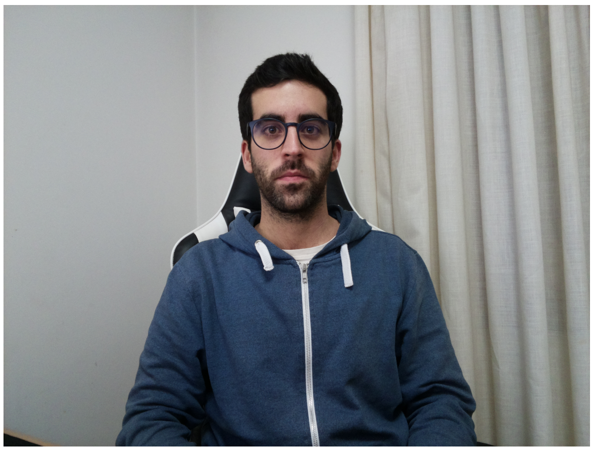


Figure 2.4: Image captured with the Pi Camera.

2.3.2 Software

The intended functionalities for the Content Display Agent software solution required some dependencies in order to fulfill the needs for displaying multimedia contents and capture the subject engadgement. In Table 2.2 are listed some of the libraries, with respective descriptions, and after that are presented the most important ones.

Library	Description
screeninfo [41]	Module for fetching location and resolution of connected physical screens. Used to fetch the Content Display Agent screen resolution.
netifaces [42]	Module for fetching connection interfaces. Used to find suitable MAC address of the Content Display Agent.
requests [43]	Module for HTTP communication. Used to perform HTTP GET and HTTP POST requests between Content Display Agents and the Web Server.

Table 2.2: Python libraries used in the Content Display Agent [15].

OMXPlayer

For the content display OMXPlayer [44] was used. It is a command-line media player that comes pre-installed in the Raspeberry Pi operating system and is able to play back popular audio and video file formats [45].

Pi Camera

The Pi Camera Python package [46] was needed in order to capture frames from the Raspberry Pi camera module. This package provides an interface to the camera module, allowing to connect to it and initialize the camera with the desired configurations.

Dlib

Dlib was used in the software of the Content Display Agent for its image processing capabilities for detecting faces. An example of a detection is provided in Figure 2.5. Only images where a positive detection is made are sent to the Web Server. More details of the implementation will be presented in Chapter 4.

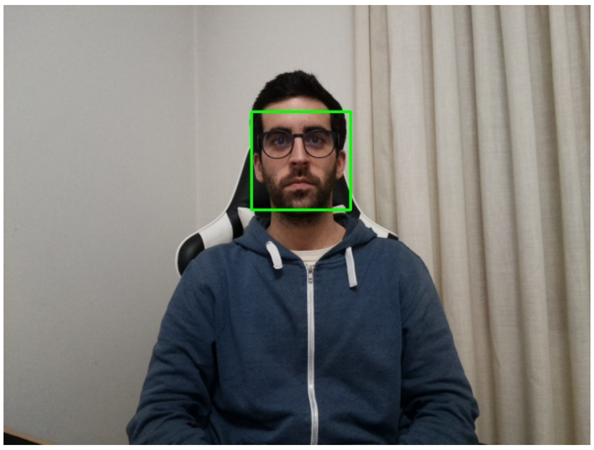


Figure 2.5: Face Detection using Dlib.

CHAPTER 3

Content Management

The purpose of this chapter is to address the system features for content management. It focuses on two of the architectural components of this project, the Web Server (Section 2.1) and the Content Management Dashboard (Section 2.2).

To start, Section 3.1 presents the division of resources. In Section 3.2 the permissions strategy used in the system is introduced. Sections 3.3 and 3.4 detail the content transformation processes and content management dashboard. Following, in Section 3.5, it is presented the database diagram and, to finish, in Section 3.6, it is where the improvements and new developments are described.

3.1 Resources Division

This Digital Signage solution gives freedom to the users to create and dispose multimedia contents into any order, and consequently display them [15].

To do this, three different resource types were defined to be the footprint for content organization in the system. Those types are the following:

- **Contents** - base resource that represents items uploaded by users that can be of any of the types metioned in Section 2.1.1;
- **Timelines** - composition of Contents with predefined sequence;
- **Views** - composition of Timelines with predefined sequence.

For visualization of this structure an illustration is provided in Figure 3.1.

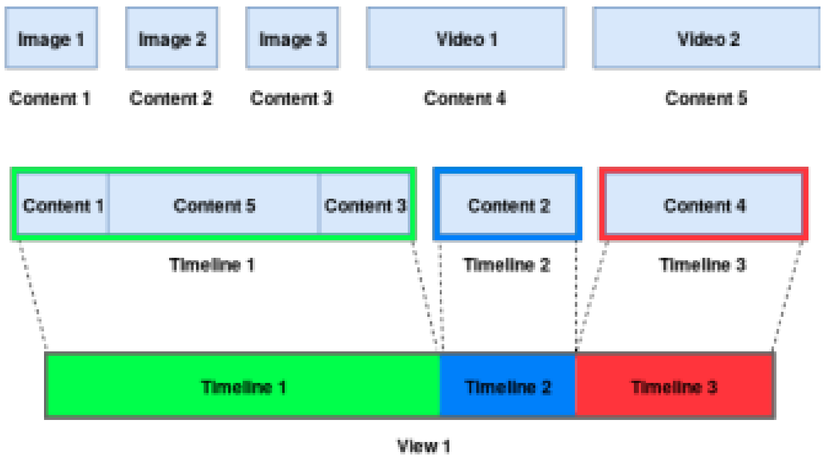


Figure 3.1: Diagram for composition of the system resources [15].

3.2 Permissions Strategy

The solution takes advantage of Django containing a built-in authentication system [47] that couples authentication and authorization. This way, to access the system users require authentication with the corresponding credentials, also, since the system can be accessed by multiple users it is important that management of multimedia resources is restricted to the respective owners, hence the use of permissions. The system is comprised of two permission levels: role based permissions and user permissions, as illustrated in Figure 3.2.

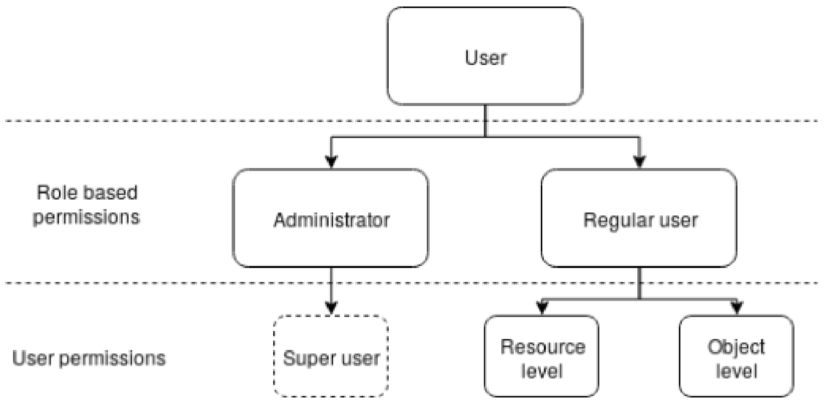


Figure 3.2: System Permissions diagram [15].

3.2.1 Role Based Permissions

As shown in Figure 3.2, the system contains two types of users. The first one, Administrator, is a user with the reponsability of managing users access and permissions. For this, an Administrator user has access to a restricted area called *Users* that provides control for all of this. In addition, this user has no restrictions when it comes to multimedia resources management, being able to add, edit and delete any kind.

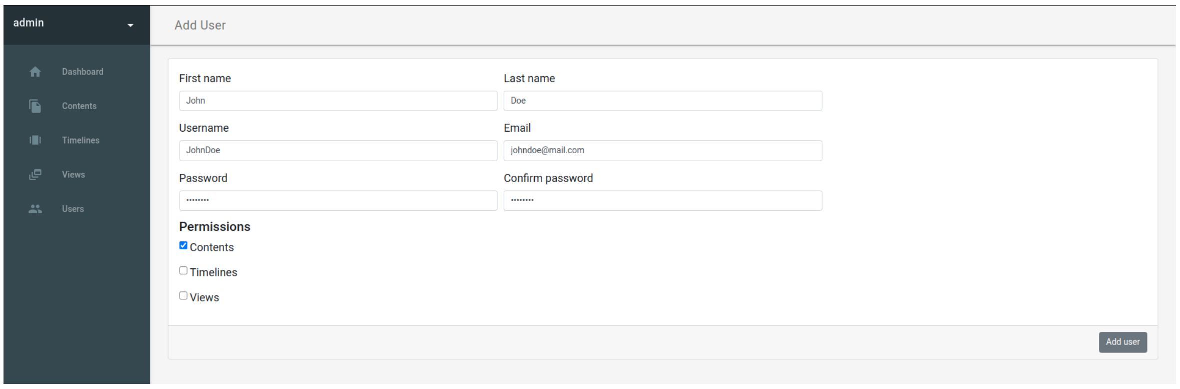
The second user type, Regular User, lives inside the system restricted by the permissions set by Administrator Users.

3.2.2 User Permissions

For a Regular user the system is designed to have two more permission levels: Resource level and Object level.

The Resource level refers to the permissions for access to the areas of the Content Management Dashboard that relate to each of the three resource types described in Section 3.1, Contents, Timelines and Views.

The attribution of these permissions is done at the creation or edition of a user in the restricted area for user management. Both the user creation and edition forms have controls for the insertion of user basic information as well as a checkbox group for permission definition. An example is provided in Figure 3.3.



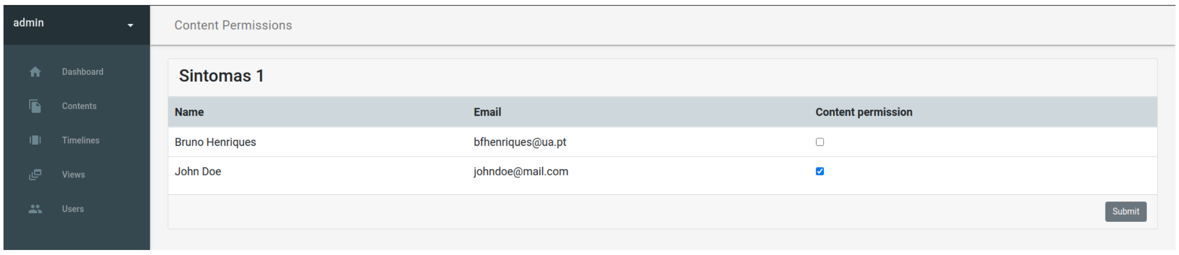
First name	Last name
John	Doe
Username	Email
JohnDoe	johndoe@mail.com
Password	Confirm password
••••••••	••••••••

Permissions
[✓ Contents]
Timelines
Views

Figure 3.3: Content Management Dashboard page for user creation.

When it comes to the Object level it refers to the access to specific multimedia resources. Following the principle of an Access Control List (ACL) the permissions are associated with the object and the list contains all users permissions to access it.

Although by default the user has access only to the objects he has created [15], this can be changed by an Administrator user by accessing the resource permissions page. In every resources list (Content, Timeline and View), each entry representing a resource object contains a padlock button (example in Figure 3.8) that redirects to the previously mentioned page. Figure 3.4 presents an illustration of this process.



Sintomas 1

Name	Email	Content permission
Bruno Henriques	bfhenriques@ua.pt	☐
John Doe	johndoe@mail.com	☑

Figure 3.4: Content Management Dashboard page for resource level permission handling.

3.3 Content Transformation

The process of content transformation can be summarized into three steps and those are listed below:

1. Content Upload;
2. Timeline Creation/Edition;
3. View Creation/ Edition.

3.3.1 Content Upload

As the first step this one is the most straight forward. Using Django's File Upload [48] files are saved to the Web Server, followed by the validation of file extension support. To finish the upload process a second validation is done according to the file format using the Python libraries presented in Table 2.1.

The page of the Content Management Dashboard that allows for content upload is illustrated in Figure 3.5.

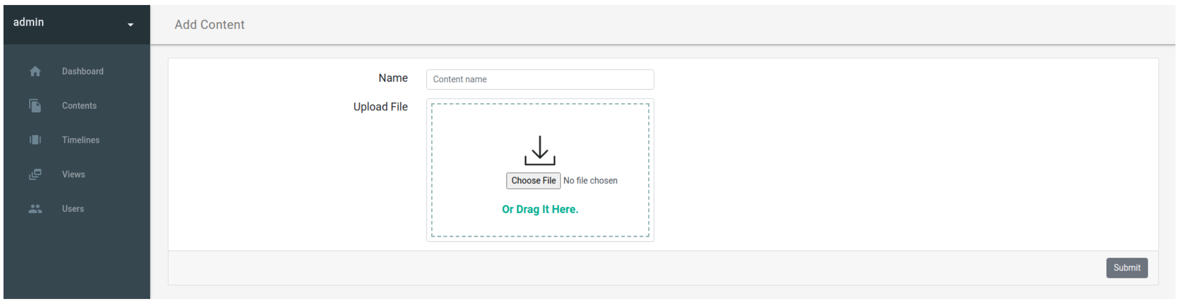


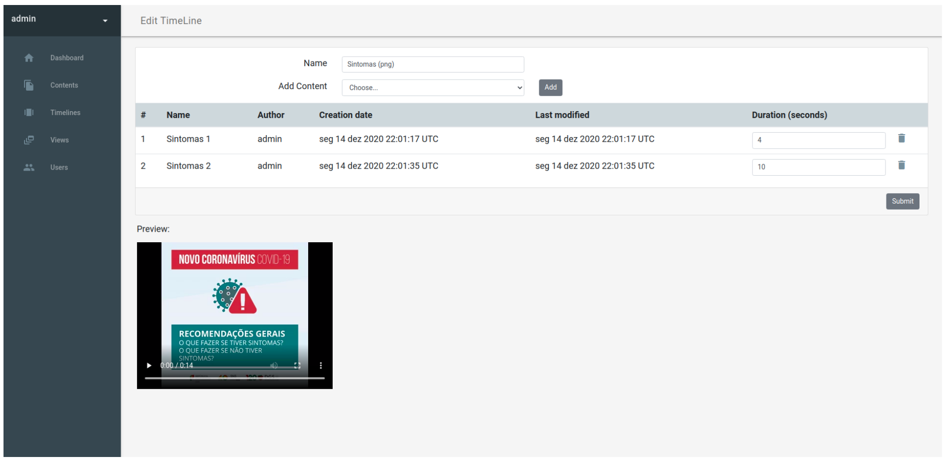
Figure 3.5: Content Management Dashboard page for Content upload.

3.3.2 Timeline Creation/Edition

It is in this step that users are allowed to combine Contents. In the Timeline creation (or edition) page a form is displayed containing controls for insertion of a name for the timeline, a dropdown button where it is possible to choose the Contents to be part of the Timeline and a table where the added Contents are listed and can be ordered by a dragging action. Each entry of the table provides an input for definition of Content display duration.

In the specific cases of Contents of presentation format (PDF and PPT) this duration value is used for the display time of each slide or page, amounting to a total duration of *duration * number of slides*. Since FFmpeg requires files in image or video format, these presentation format files have to be converted. For PDFs the conversion process starts by creating an image for each slide and registering the names for each of them in a text file. At this point FFmpeg can read the text file and create the video file. As for the PPTs, they have to first be converted to PDF using LibreOffice's command line converter [49] and only then be subject to the process mentioned before.

In Figure 3.6 it is provided and illustration of the Timeline edition page.



Name: Sintomas (png)
Add Content: Choose...

#	Name	Author	Creation date	Last modified	Duration (seconds)
1	Sintomas 1	admin	seg 14 dez 2020 22:01:17 UTC	seg 14 dez 2020 22:01:17 UTC	4
2	Sintomas 2	admin	seg 14 dez 2020 22:01:35 UTC	seg 14 dez 2020 22:01:35 UTC	10

Submit
Preview:
NOVO CORONAVÍRUS COVID-19
RECOMENDAÇÕES GERAIS
O QUE FAZER SE TIVER SINTOMAS?
O QUE FAZER SE NÃO TIVER SINTOMAS?
0:00 / 0:14

Figure 3.6: Content Management Dashboard page for Timeline edition.

3.3.3 View Creation/Edition

View creation is an automatic process triggered by the registration of a Content Display Agent in the system (more details presented in Section 4.1.2). When this happens the View is initialized as unconfigured, having no name and no Timelines associated. In order to have multimedia contents displaying in the Content Display Agent the View has to be configured first. To do this, similarly to the Timeline creation page, the View edition page displays a form with controls for the name, a dropdown button to choose Timelines and a table listing the added ones with the same dragging action to set the order.

On View form submission, with the help of FFmpeg, the MP4 file creation process is started. When finished, the View video file will be available for the corresponding Content Display Agent to fetch it and start displaying.

The page for View edition is identical to the page for Timeline edition, presented in Figure 3.6.

3.4 Content Management Dashboard

As previously mentioned in Section 2.2, Django is capable of exposing a webpage, for which in this project, the implementation was done using Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and Javascript [50].

After a successfull log in, the user lands in the main page of the Content Management Dashboard, as illustrated in Figure 3.7.

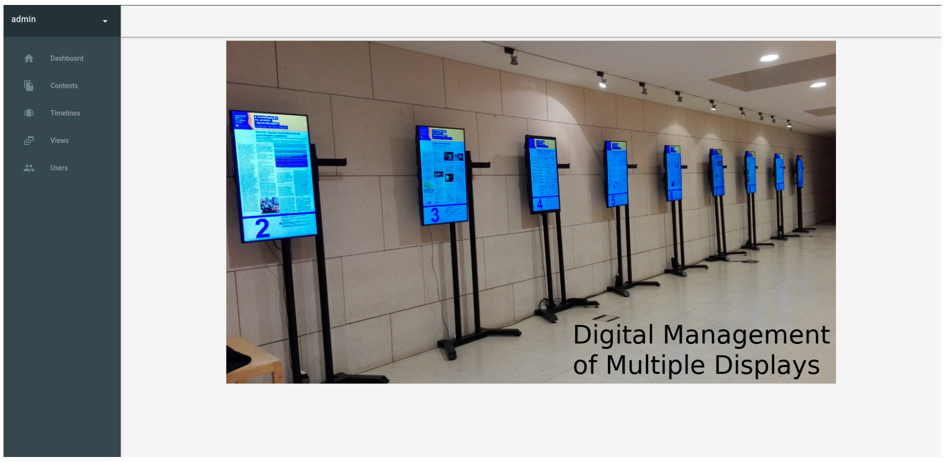
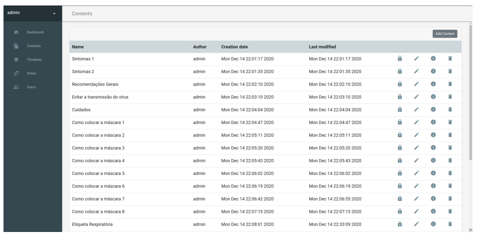


Figure 3.7: Content Management Dashboard initial page.

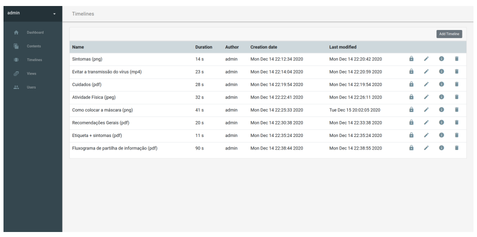
Also in Figure 3.7, it is possible to see the interface contains top and left navigation bars. The first, allows users to access the profile page and to logout. The second one, allows for navigation to the areas corresponding to each resource type introduced in Section 3.1, as well as the user management area. These navigation links are displayed according to the user permissions described in Section 3.2.2.

Clicking on the navigation links the user is taken to the resources listing pages. Examples are provided in Figures 3.8 and 3.9.



Name	Author	Creation date	Last modified
Sintomas 1	admin	Mon Dec 14 22:01:17 2020	Mon Dec 14 22:01:17 2020
Sintomas 2	admin	Mon Dec 14 22:01:35 2020	Mon Dec 14 22:01:35 2020
Recomendações Gerais	admin	Mon Dec 14 22:02:10 2020	Mon Dec 14 22:02:10 2020
Evitar a transmissão do vírus	admin	Mon Dec 14 22:03:10 2020	Mon Dec 14 22:03:10 2020
Cuidados	admin	Mon Dec 14 22:04:04 2020	Mon Dec 14 22:04:04 2020
Como colocar a máscara 1	admin	Mon Dec 14 22:04:47 2020	Mon Dec 14 22:04:47 2020
Como colocar a máscara 2	admin	Mon Dec 14 22:05:11 2020	Mon Dec 14 22:05:11 2020
Como colocar a máscara 3	admin	Mon Dec 14 22:05:20 2020	Mon Dec 14 22:05:20 2020
Como colocar a máscara 4	admin	Mon Dec 14 22:05:43 2020	Mon Dec 14 22:05:43 2020
Como colocar a máscara 5	admin	Mon Dec 14 22:06:02 2020	Mon Dec 14 22:06:02 2020
Como colocar a máscara 6	admin	Mon Dec 14 22:06:19 2020	Mon Dec 14 22:06:19 2020
Como colocar a máscara 7	admin	Mon Dec 14 22:06:42 2020	Mon Dec 14 22:06:55 2020
Como colocar a máscara 8	admin	Mon Dec 14 22:07:15 2020	Mon Dec 14 22:07:15 2020
Etiqueta Respiratória	admin	Mon Dec 14 22:08:01 2020	Mon Dec 14 22:33:09 2020

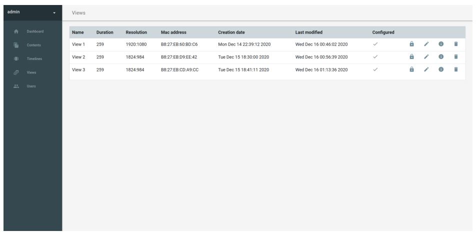
(a) Contents Listing



Name	Duration	Author	Creation date	Last modified
Sintomas (png)	14 s	admin	Mon Dec 14 22:12:34 2020	Mon Dec 14 22:20:42 2020
Evitar a transmissão do vírus (mp4)	23 s	admin	Mon Dec 14 22:14:04 2020	Mon Dec 14 22:20:59 2020
Cuidados (pdf)	28 s	admin	Mon Dec 14 22:19:54 2020	Mon Dec 14 22:19:54 2020
Atividade Física (jpeg)	32 s	admin	Mon Dec 14 22:22:41 2020	Mon Dec 14 22:26:11 2020
Como colocar a máscara (png)	41 s	admin	Mon Dec 14 22:25:33 2020	Tue Dec 15 20:02:05 2020
Recomendações Gerais (pdf)	20 s	admin	Mon Dec 14 22:30:38 2020	Mon Dec 14 22:33:38 2020
Etiqueta + sintomas (pdf)	11 s	admin	Mon Dec 14 22:35:24 2020	Mon Dec 14 22:35:24 2020
Fluxograma de partilha de informação (pdf)	90 s	admin	Mon Dec 14 22:38:44 2020	Mon Dec 14 22:38:55 2020

(b) Timelines Listing

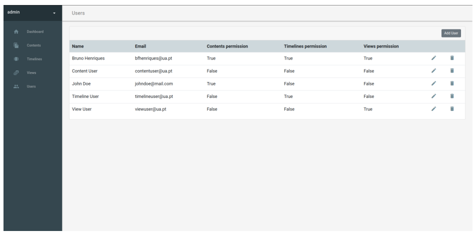
Figure 3.8: Content Management Dashboard resource listing pages (1).



Views

Name	Duration	Resolution	Mac address	Creation date	Last modified	Configured
View 1	259	1920:1080	B8:27:EB:60:BD:C6	Mon Dec 14 22:39:12 2020	Wed Dec 16 00:46:02 2020	✓
View 2	259	1824:984	B8:27:EB:D9:EE:42	Tue Dec 15 18:30:00 2020	Wed Dec 16 00:56:39 2020	✓
View 3	259	1824:984	B8:27:EB:CD:A9:CC	Tue Dec 15 18:41:11 2020	Wed Dec 16 01:13:36 2020	✓

(a) Views Listing



Users

Name	Email	Contents permission	Timelines permission	Views permission
Bruno Henriques	bfhenriques@ua.pt	True	True	True
Content User	contentuser@ua.pt	False	False	False
John Doe	johndoe@mail.com	True	False	False
Timeline User	timelineuser@ua.pt	False	True	False
View User	viewuser@ua.pt	False	False	True

(b) Users Listing

Figure 3.9: Content Management Dashboard resource listing pages (2).

3.5 Database

The database of this project capitalizes on Django's database-access API. The database tables are mapped from Python subclasses of Django's Model class [51]. A diagram of the structure, showing the database tables and their respective fields, is provided in Figure 3.10.

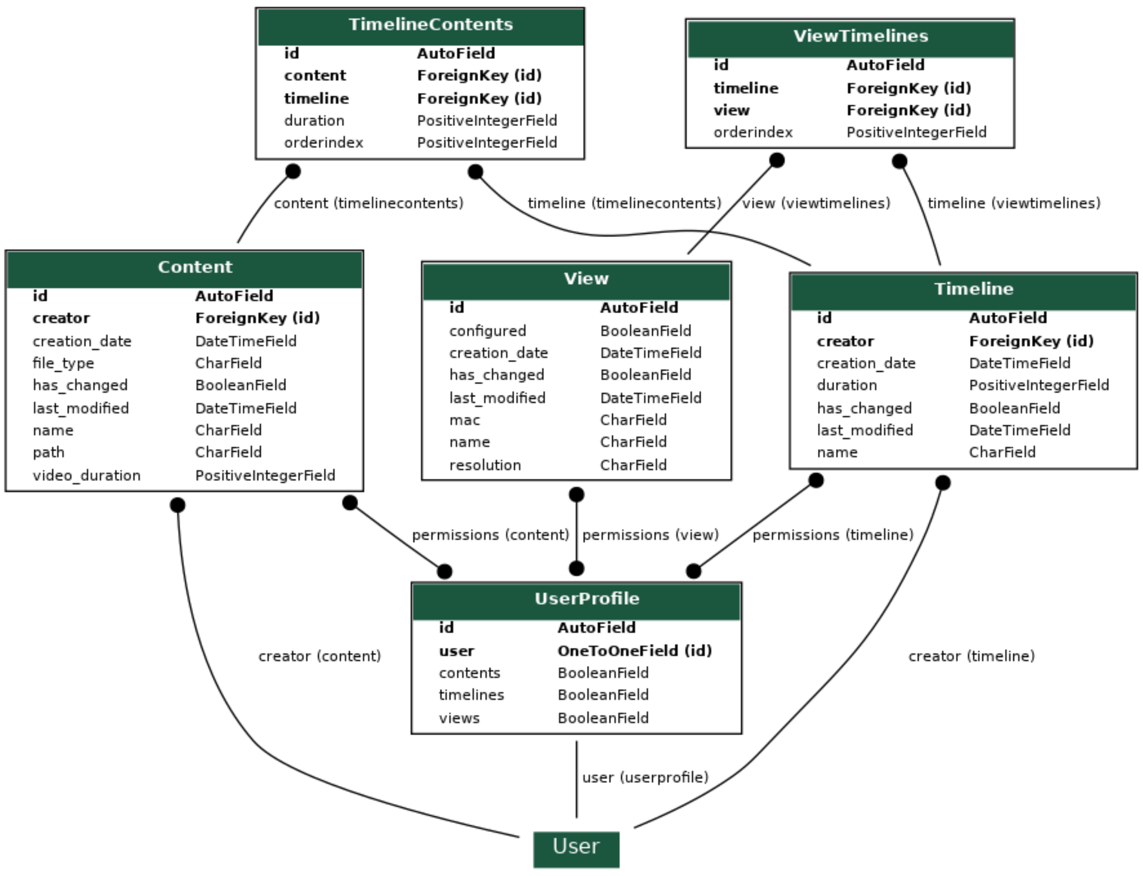


Figure 3.10: Diagram of the database structure.

3.6 Improvements and Developments

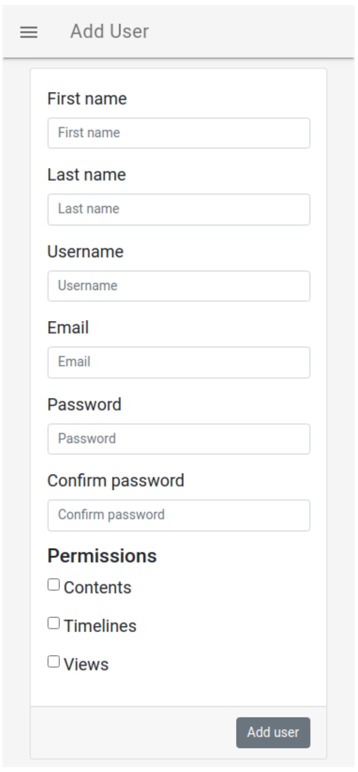
In order to start the development process, some requirements were established. The first one was implementation of design improvements to the webpage, concerning a cleaner look and responsive capabilities. Other than that, it was important to include the ability to preview the multimedia resources in the system, making clear to the user what they really are working with.

When it comes to the objectives of this dissertation it was required the introduction of a new page, accessible from every resource list (Contents, Timelines and Views), where it is possible to display the data generated associated with each resource, process that is explained in more detail in Chapter 5.

3.6.1 Design

As a way of making the webpage cleaner Bootstrap's [52] Card [53] class was used. This is a flexible and extensible content container that includes options for headers and footers, which help organize the pages. An illustration of this can be found in Figure 3.4 where the option for the card header is active and displays the content name, in the card body lives the information of interest, in this case, the users permissions for resource access, and at the bottom in the footer, is the button for submission of the performed changes.

For content display in web pages Bootstrap also provides control over building a grid layout using columns [54]. In order to address the requirement of responsive capabilities these columns were reimplemented using column classes with extensions for activation in specific display sizes. To see this in action, Figure 3.11, showing a screenshot of a mobile display, provides a comparison to Figure 3.3, which shows a screenshot of a desktop display.



Add User

First name

First name

Last name

Last name

Username

Username

Email

Email

Password

Password

Confirm password

Confirm password

Permissions

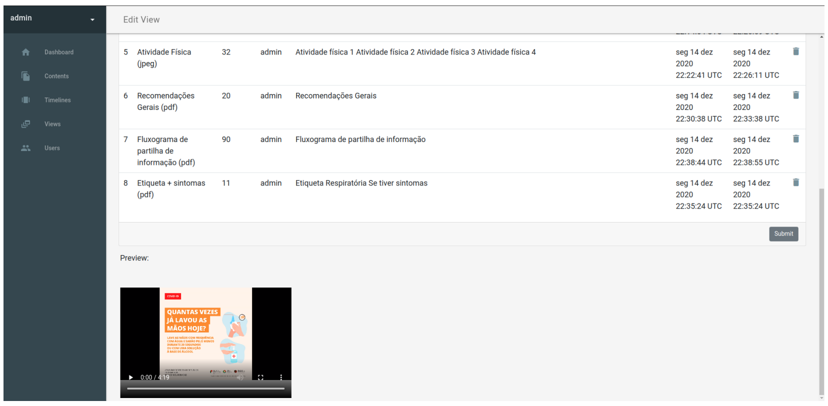
- [ ] Contents
- [ ] Timelines
- [ ] Views

Add user

Figure 3.11: Content Management Dashboard page for user creation on a mobile device screen.

3.6.2 Resource Preview

Controls for preview of the multimedia resources in the system were added. For Contents, a preview button was added to the Content edit page that, when clicked, opens the respective file in a new tab of the browser. For Timelines and Views, the preview implementation passed by adding a video control in the respective resource type edit page, as illustrated in Figure 3.12 for the generated MP4 file from View configuration.



5	Atividade Física (jpeg)	32	admin	Atividade física 1 Atividade física 2 Atividade física 3 Atividade física 4	seg 14 dez 2020 22:22:41 UTC	seg 14 dez 2020 22:26:11 UTC
6	Recomendações Gerais (pdf)	20	admin	Recomendações Gerais	seg 14 dez 2020 22:30:38 UTC	seg 14 dez 2020 22:33:38 UTC
7	Fluxograma de partilha de informação (pdf)	90	admin	Fluxograma de partilha de informação	seg 14 dez 2020 22:38:44 UTC	seg 14 dez 2020 22:38:55 UTC
8	Etiqueta + sintomas (pdf)	11	admin	Etiqueta Respiratória Se tiver sintomas	seg 14 dez 2020 22:35:24 UTC	seg 14 dez 2020 22:35:24 UTC

Figure 3.12: View Preview.

3.6.3 Content Transformation

During the development process, it was detected that conversion of PDF files was generating video files with longer duration than expected. For example, if a PDF file had 3 pages, and the defined time for each page was 10 seconds, while the generated video should have 30 seconds total duration, it would have 33 seconds and the extra 3 seconds would be a continuation of the last page. It was important to do this correction because, when processing a viewer detection, calculations are made to determine which multimedia content was displaying at that exact moment (more on this matter in Chapter 5).

That being said, after converting a PDF to a video file, FFmpeg's method *ffmpeg _extract _subclip()* was used. This trimming action generates a final video file with expected duration.

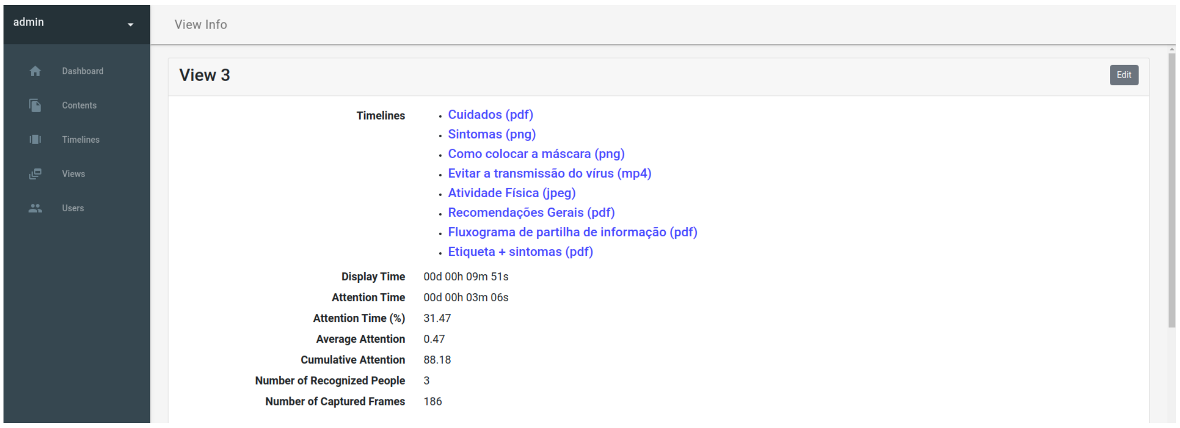
3.6.4 Knowledge Discovery

This requirement for the Content Management Dashboard was of great relevance to this dissertation's objectives, since it was about providing access to information about the multimedia contents "performance", aiming for an understanding of the impact, of the multimedia contents, on the subjects who visualize them. It is possible to visualize information about Contents, Timelines and Views, in the respective info page, accessible from each entry in the resources list pages.

Taking as an example a View info page, illustrated in Figure 3.13, it is possible to navigate to its corresponding edit page from the header of the card, since users migth want to perform some changes after interpretation of the information. On the card body, at the top, are referenced the lower layer multimedia resources that are part of this View, which are the Timelines that compose it, linking to their respective info page. This is also the case for Timelines, but not for Contents, since these are the introductory resource type. Having these references provides easy navigation through the resource tree.

After that, information like Display Time, Attention Time, Average and Cumulative Attention values, number of detected subjects and the number of total captured frames is presented. The reasoning for the information fields provided, and the implementation behind them, is explained in more detail in Chapter 5.

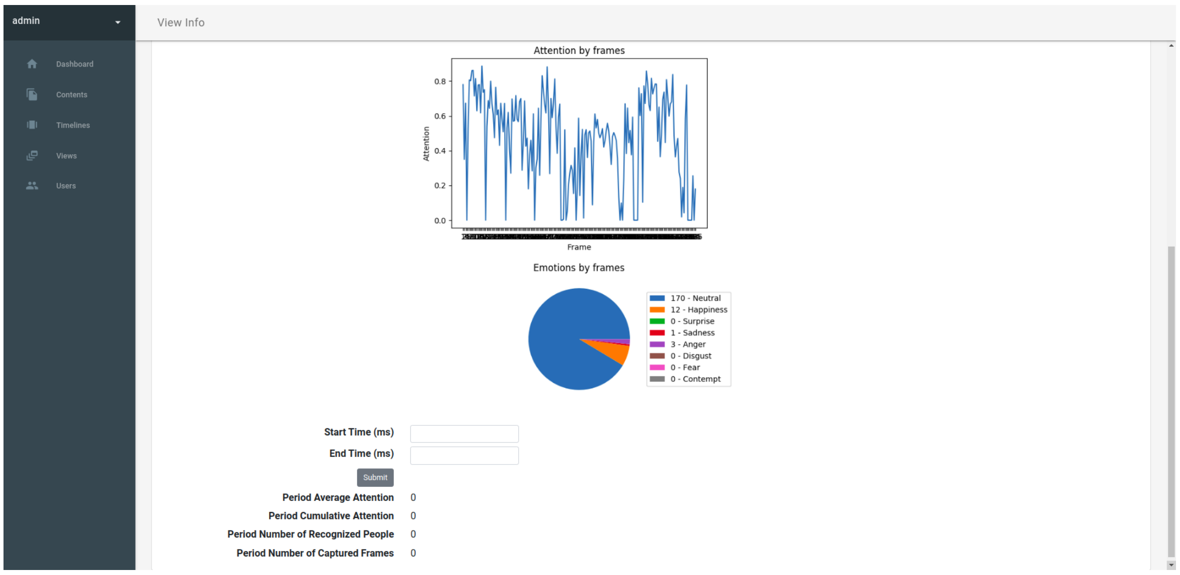
It is also possible to access data in a graphical way. For that, a line chart of the atention values per captured frame over time, and a pie chart of the number of times emotions from a specific set were recognized, are presented, as shown in Figure 3.14. At the bottom can be found a simple form where, with the input of specific start and finish timestamps, the information for that specific period of time is provided.



View 3

Timelines	• Cuidados (pdf) • Sintomas (png) • Como colocar a máscara (png) • Evitar a transmissão do vírus (mp4) • Atividade Física (jpeg) • Recomendações Gerais (pdf) • Fluxograma de partilha de informação (pdf) • Etiqueta + sintomas (pdf)
Display Time	00d 00h 09m 51s
Attention Time	00d 00h 03m 06s
Attention Time (%)	31.47
Average Attention	0.47
Cumulative Attention	88.18
Number of Recognized People	3
Number of Captured Frames	186

Figure 3.13: Content Management Dashboard page for View information (1).



Start Time (ms)	
End Time (ms)	
Period Average Attention	0
Period Cumulative Attention	0
Period Number of Recognized People	0
Period Number of Captured Frames	0

Figure 3.14: Content Management Dashboard page for View information (2).

CHAPTER 4

Content Display Agent

The purpose of this chapter is to provide a detailed description of the software running in the terminal agents metioned in Section 2.3. To start, in Section 4.1, it will be addressed the part of the solution regarding the display of contents, presenting it's features and life cycle. After that, in Section 4.2, the sofware implementation in charge of collecting data from the Pi Camera will be introduced.

4.1 Content Display

The Content Display Agent's software started as a single thread Python program. When an agent is properly set up, the program is set to run on boot and permanently until a shutdown occurs. It requires an active network connection to be able to establish a connection to the Web Server (Section 2.1) over Hypertext Transfer Protocol (HTTP). On program start, a series of steps are in place to ensure the agent functions as intended, meaning, being able to register in the Web Server, in order to have its associated View configured, and consequently be able to fetch the corresponding generated MP4 video file. When the download is complete, a video player process is started and, in order to keep the multimedia contents updated, the agent periodically checks for view updates on the server, downloading the new video file if necessary. A diagram representation of this cycle is provided in Figure 4.1.

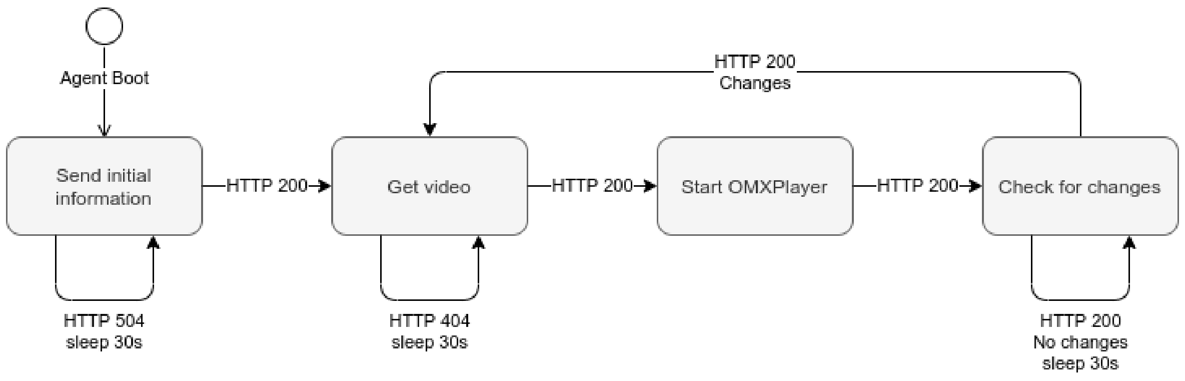


Figure 4.1: Life cycle of the Content Display Agent content display process.

4.1.1 Web Server Communication

Requests to the Web Server are directed at URL endpoints exposed by Django. Through its life cycle, the software required three URLs, which are listed below:

- *login/*;
- *monitor/new_monitor/* - Section 4.1.2;
- *monitor/check_for_changes/* - Section 4.1.5.

To have protected connections to the server a request to the *login* URL asks for a valid Cross-site request forgery (CSRF) token. Only with this token can the remaining GET and POST requests be acknowledged by the server.

4.1.2 Registration

Agent registration requires the MAC address of the device, obtained with netifaces [42], and the screen resolution of the connected display, obtained using screeninfo [41]. In the Web Server, the *monitor/new_monitor/* URL, receives this information to serve as unique identification for each of the registered Content Display Agents and to correctly generate the digital contents. The request response contains the server path for the corresponding video file.

4.1.3 Content Download

Video file download is done with an HTTP GET request to the previously obtained path. Successful download creates a local file and in case of download failure, for example when View configuration hasn't been done yet, which means the video file doesn't exist, the request is repeated periodically until it is possible to perform a successfull download.

If on program start, the video file already exists locally from a previous boot, video playback is started right away, skipping the initial download. If during the agent's downtime changes were performed to the View, update of the video file will be done the next time the agent checks for View updates, as explained in Section 4.1.5.

4.1.4 Video Playback

Having the local video file available, video playback is started by launching an OMXPlayer process. The player takes as input the video file path and some attribute flags to set the intended behaviour. Using *-g*, the player is enabled for log generation, in order to have clean display of the contents, the background is set to black with *-b* and the status information is hidden with *-no-osd*. Finally, *-loop* is used to keep the video playing repeatedly.

4.1.5 View Update

Since the Content Display Agents can be running for long periods of time, to avoid restarting the agent in order to fetch the latest version of the video file, it periodically polls the Web Server, checking for updates on the View. Polling is done every 60 seconds and, if the server reports View changes, the process for content download is triggered and performed as described in 4.1.3, followed by a reload of the OMXPlayer process.

4.2 Data Collection

In sight of the objectives presented in Section 1.2, data collection is the first process of the audience assessment functionality of the system.

As metioned in Section 4.1, the software was implemented to run on a single thread, thus making the introduction of multi-threading the first goal of the implementation, since both the content display process and the data collection process require blocking code to perform it's continuous life cycle, and the objective was to have them run side by side.

The new process is in charge of connecting to the Pi Camera and continuously capture images of the area in front of the display, while performing real-time face detection and reporting to the Web Server when atleast one detection is made.

4.2.1 Multi-Threading

Implementation of multi-threading was done using the *threading* module [55] included in Python. The creation of custom threads required the definition of subclasses of the *Thread* class overriding the constructor *___init___(self, [args)* and the *run(self, [args])* method. Two subclasses were created, one for the content display process and another for the data collection process. With code readability and organization in mind, in the *run(self, [args])* method of each thread only lives the code responsible for the life cycle of each process and the supporting variables and methods are part of a third class representing a shared space. This class is passed as an argument to both threads, which allows for information sharing between them. In Figure 4.2 is provided a diagram illustration of the main thread of the software.

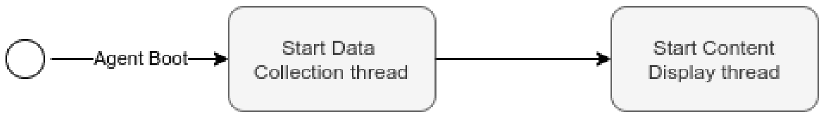


Figure 4.2: Improved main thread of the Content Display Agent software.

4.2.2 Content Display Thread

This thread was the re-implementation of the life cycle detailed in Section 4.1 with the addition of two features. The first, had to do with reporting to the Web Server that the digital contents of the View have started playing in the agent. This was done after every OMXPlayer process start or reload, by calling the method *report _view _start()*, which performs an HTTP POST request to a new URL endpoint exposed by the server, *monitor/view _start*. In this URL, the server expects the payload of the request to contain as data the absolute datetime in milliseconds of when the player was started.

The second, is meant to synchronize the start of image captures in the data collection thread. Using the *Event* class provided by the *threading* module, it is possible to have communication between the threads in the form a flag. Again, right after the start of the OMXPlayer process, this event flag is *set()* to *True* by the content display thread. In this way, the data collection thread can check the flag value and decide to perform an image capture or not.

4.2.3 Data Collection Thread

The data collection thread starts by establishing a connection to the Pi Camera. Then, it enters a continuous loop where it checks the *Event* flag value to ensure captures are only done when the digital contents are displaying. If they are, it checks for the presence of subjects in the camera field of view. When a detection occurs, the captured image is reported to the Web Server for extraction of data. In Figure 4.3, an illustration of this life cycle is provided.

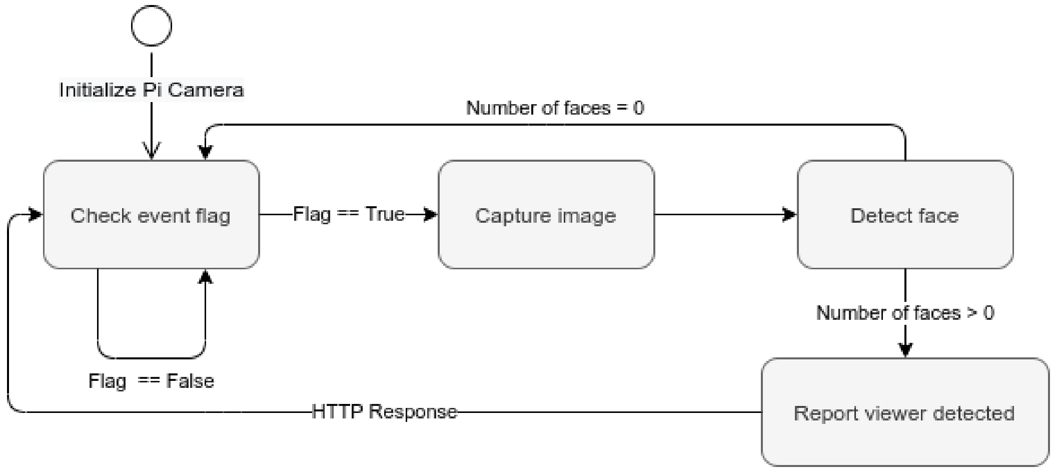


Figure 4.3: Content Display Agent data collection thread.

4.2.4 Image Capture

Using the dependency presented in Section 2.3.2, the Pi Camera is initialized with a resolution of *640 * 480* pixels at a framerate of 30 frames per second. Higher resolutions proved demanding on the modest computing power of the Raspberry Pi, increasing the time needed to perform face detection.

Calling the *capture()* method captures an image to a *numpy* array object which is then passed through the face detection algorithm. The algorithm takes about a second to run, so, even though the camera runs at 30 frames per second, the agent is only capable of processing captures at 1 frame per second.

4.2.5 Face Detection

The regions of interest in an image are the faces of subjects engadging with the displaying contents. Face detection is performed using Dlib's pre-trained Histogram of Oriented Gradients (HOG) based face detector [56]. HOG is a feature descriptor technique that counts occurences of gradient orientation. The algortithm extracts the bounding boxes of faces and returns a list of objects equal in length to the number of faces detected in the image. The length of this list serves as the validation required for acknowledgement of successful detections in the agent and contributes for close to real-time detections. Full face detection is done server side, more details will be presented in Chapter 5.

4.2.6 Detection Report

If one face is detected by the face detection algorithm, the captured image is encoded to JPEG through OpenCV's *imcode()* method and reencoded to text. At this point, the thread reaches the last step of its life cycle and an HTTP POST request to the URL *monitor/viewer _detected* is performed. Its payload contains the absolute datetime in millisseconds of the moment of image capture, the relative time in seconds of the position in the video, possible to get from OMXPlayer when the detection occurs, and finally, the enconded image. Completing this step restarts the agents's capture life cycle.

CHAPTER 5

Data Extraction and Mining

In order to achieve this project's main objective, concerning the integration of audience assessment features into the Digital Signage solution, the concepts of data extraction and data mining were introduced. Data extraction is the process which allows the system to retrieve data from the image sources provided by the data collection process [57]. Data mining is the process that makes the system capable of finding patterns and correlations within the extracted data, in order to understand relevant information [58].

Section 5.1 presents the proposed metrics generated by the system. Section 5.2 presents the flow of data through the system. The introduction of the computer vision module is provided in Section 5.3. To finish, the changes to the database models required for storage of the mined data are presented in Section 5.4.

5.1 System Generated Data

The aim of this computer vision solution is to mine quantitative data in order to evaluate the impact of the multimedia contents on the subjects. This requires for the proposed implementation to keep track of a set of metrics of interest. These are meant to help measure engadgement time and quality, and are listed below:

- *Display Time* - the total time of content display;
- *Attention Time (and Percentage)* - time of subject engagement;
- *Average and Cumulative Attention Coefficient* - level of engagement (value ranges from 0 to 1);
- *Emotions* - occurrences of recognized emotions;
- *Number of Recognitions* - number of different recognized subjects.

5.2 Data Flow

The flow of data through the system starts on the Content Display Agents, where it is collected, it then reaches the Web Server where it is processed and stored. Communication between these components triggers several actions that are of interest to the system when it comes to the system data it proposes to generate.

Every time a Content Display Agent is started it notifies the Web Server in order to synchronize the information about the contents display time. During its life cycle, presented in Figure 4.1, the agent periodically polls the server for changes, and when it does, the content display time is incremented by the difference of time to the last poll, which the server also keeps track of. Moreover, when a subject detection is done at the agent, it promptly reports to the server, sending the captured image and timestamps.

Reaching the server, the data extraction process is started. The implementation took into consideration that the time it takes for this process to end and return an HTTP response affects the agent's capture rate. With this in mind, a *Thread* is started to run the method *process _viewer()* and perform the intended processing without holding up the requests response. A diagram illustrating this process is provided in Figure 5.1.

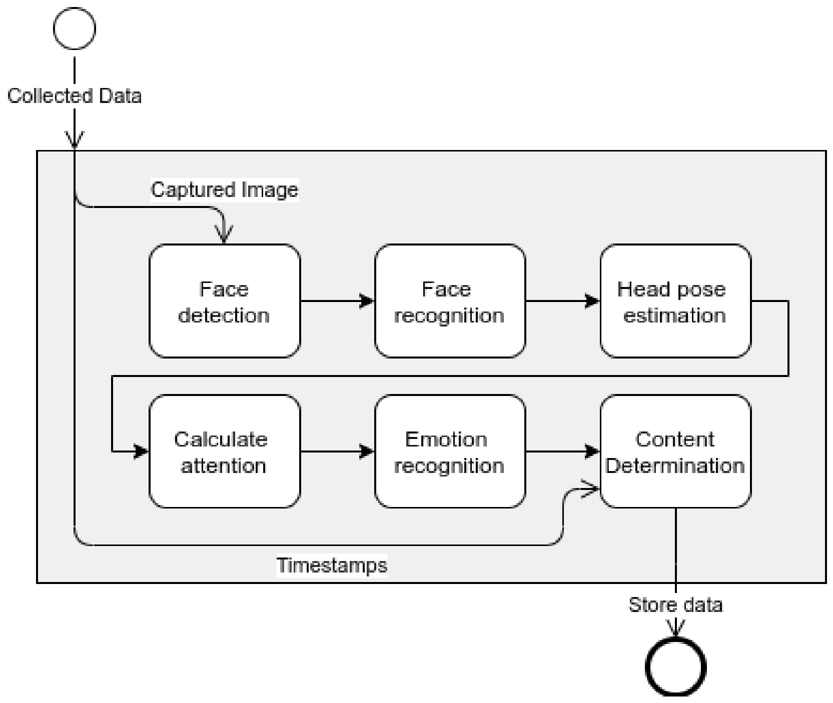


Figure 5.1: Flow of the data extraction and mining processes.

Section 5.3 will describe the first 5 steps of the process. For the last step, Content Determination, the system uses the timestamps provided by the agent to determine which Content, Timeline and View was displaying at the time of subject detection. On successful determination of each of the resources, the extracted information is stored in each of the corresponding database entries.

5.3 Computer Vision Module

This module holds all the computer vision capabilities of the system. It implements methods for face detection, face recognition, head pose estimation, attention calculation and emotions recognition. In Figure 5.2, some of the extracted information can be seen drawn over a captured image.

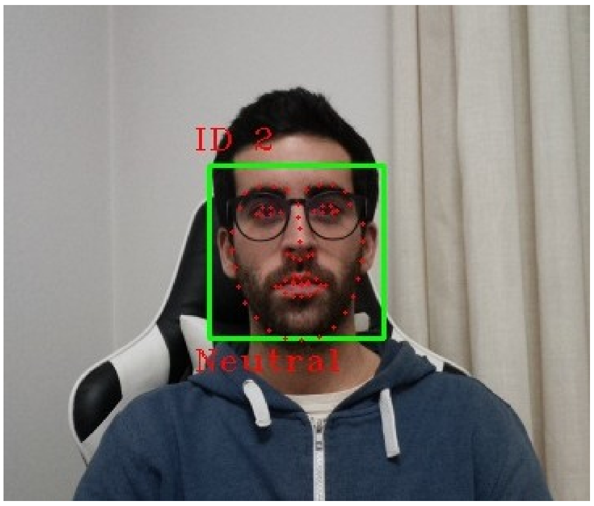


Figure 5.2: Image with drawn annotations for the recognized subject ID, face bounding box, facial landmarks and recognized emotion.

5.3.1 Face Detection

Using DLib's HOG based frontal face detector the system is able to extract the bounding boxes of the subjects faces present in the captured image. After that, going through the boundind boxes list, DLib's shape predictor method, previously loaded with a pre-trained 68 point face landmarks model [59], is used to detect facial landmarks such as eyes, eyebrows, nose, mouth and jawline that are present inside the bounding box.

5.3.2 Face Recognition

To start, the face of the subject is extracted from the image as a *numpy* array, using the prediction obtained with the face detection process and running DLib's get face chip method. This gives the system an object that is a definition of the candidate to be passed through the recognition process.

Previously loading DLib's face recognition model method with a pre-trained Residual Neural Network (ResNet) model [59], that converts human faces into 128D vectors, allows the system to compute face descriptors of the candidate using the *numpy* array of the face.

With the candidates descriptors, the system loops through the existing face descriptors, which are stored in a file on the Web Server, in order to find the best comparison. If the best comparison is within the system recognition confidence value, the counter for captured frames associated with this candidate is incremented. Otherwise, a new ID and its corresponding descriptor is added to the descriptors file.

5.3.3 Head Pose Estimation

Taking the detected facial landmarks, the system is able to estimate the head pose of the subject. Using OpenCV's *solvePnP* method it calculates the rotation and translations vectors that transform the 2D facial landmarks into 3D coordinates. It is able to decompose a projection matrix created with the previously calculated vectors in order to extract Euler Angles that describe the orientation of the face.

5.3.4 Attention Calculation

From the Euler Angles the system takes the pitch and yaw values to calculate the subjects attention. For this, the system assumes that frontal orientation implies that subjects are paying attention, providing an higher value for the attention coefficient. Deviations from the frontal orientation decrease the attention coefficient value, and if they are over 30 degrees, they aren't considered, since this can mean that subjects aren't looking at the display, and also due to the face detection process difficulties to detect faces after this point.

5.3.5 Emotions Recognition

Using ONNX Runtime, which supports Open Neural Network Exchange (ONNX) open standard format machine learning models, loaded with a pre-trained model for emotions, the system is able to infer which emotion a subject is displaying [60]. The emotions are categorized in a set and are the following: Neutral, Happiness, Surprise, Sadness, Anger, Disgust, Fear, Contempt.

5.4 Database Models

Storage of the extracted information is done for every resource type of the system, Contents, Timelines and Views. In order to properly store it some changes had to be done to the existing database models. These changes affect multiple database models, and those are listed below:

- Content;
- Timeline;
- View;
- TimelineContents.

In Figure 5.3, it is presented an illustration of the database object-relation mapping diagram containing the new fields added to the previously listed models. After that, Table 5.1 highlights the added fields, and provides a description of their purpose as well as the models they are included in.

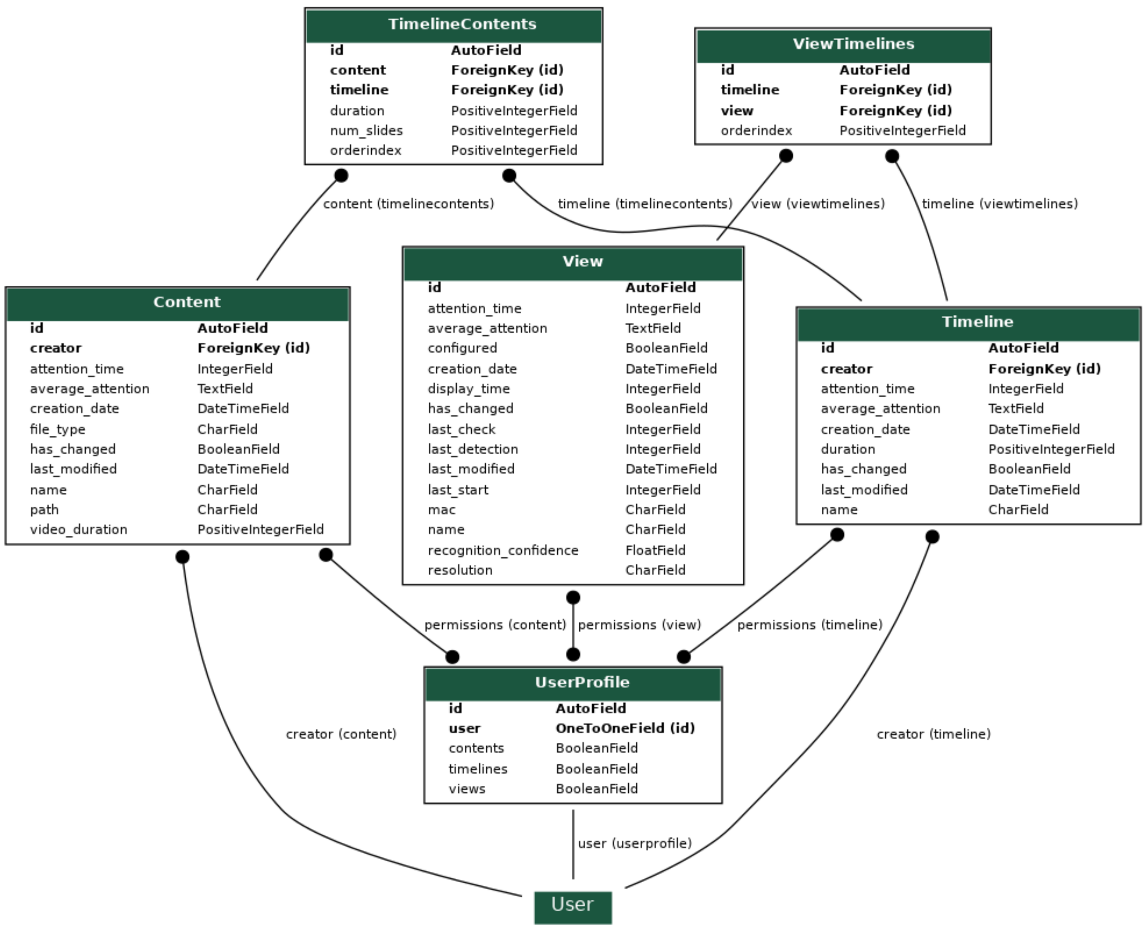


Figure 5.3: Diagram of the database structure showing the new fields added to tables Content, View, Timeline and TimelineContents.

Field	Description	Model(s)
attention_time	Stores the total seconds of subject engagement.	Content, Timeline, View
average_attention	Stores a dictionary of relevant data. See Snippet 1.	Content, Timeline, View
display_time	Stores the total seconds of exposure.	View
last_check	Stores a datetime in milliseconds of the last access to the model.	View
last_detection	Stores a datetime in milliseconds of the last subject detection.	View
last_start	Stores a datetime in milliseconds of the last View start.	View
num_slides	Stores the number of slides or pages of contents of the presentation formats (PDF and PPT).	TimelineContents
recognition_confidence	Stores the confidence coefficient for the face recognition process.	View

Table 5.1: New database fields and their description.

```
{
    key: {
            average_attention: [],
            cumulative_attention: 0,
            emotions: {
                        "Anger": 0,
                        "Comtempt": 0,
                        "Disgust": 0,
                        "Fear": 0,
                        "Happiness": 0,
                        "Neutral": 0,
                        "Sadness": 0,
                        "Surprise": 0,
                },
                emotions_list: [],
                frames: 0
                timestamps: []
    }
}
```

Snippet 1: Example of an initialized key-value pair (key is a subject ID and value is a dictionary containing the extracted data).

CHAPTER 6

Results

After having obtained a functional solution satisfying the identified requirements, it was necessary to perform some tests to assess its efficiency. Accordingly, a demonstrative scenario was conceived and an experiment planned and carried out. This chapter presents such experiment and provides an analysis of the obtained results. Section 6.1 describes the deployment scenario. Section 6.2 adressess the preparation of content for the experiment. The guidelines for the participants are presented in Section 6.3. To finish, in Section 6.4, it will be presented a discussion over the experiment results.

6.1 Deployment Scenario

The location of the experiment was IEETA. The institute building is on University of Aveiro campus and contains offices and coworking open spaces used by faculty and students. Spread across the area are three monitors used to display multimedia contents, allowing for information dessimination about the departments project's or events. Figure 6.1 presents an illustration of the monitors.

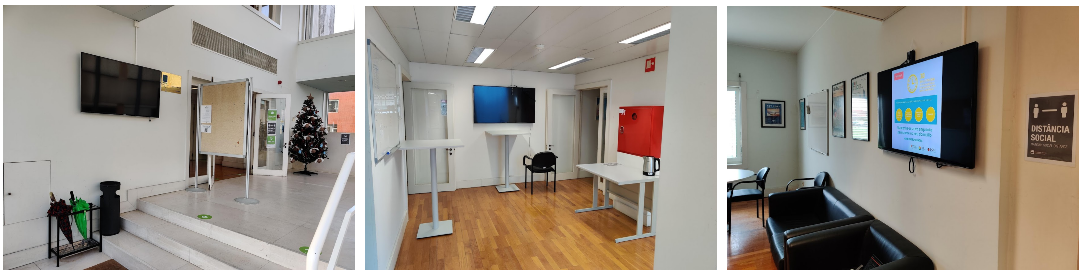


Figure 6.1: IEETA's three monitors.

Connected to these monitors are three Raspberry Pi 3 Model B+ [36] equipped with a Pi Camera [37] and setup to run the software implemented as described in Chapter 4. Figure 6.2 provides an illustration of the setup.

The Django Web Server was deployed to a Virtual Machine hosted by Serviços de Tecnologias de Informação e Comunicação (sTIC) which makes connections possible from any terminal inside the Eduroam network or over Internet connection using a Virtual Private Network (VPN).

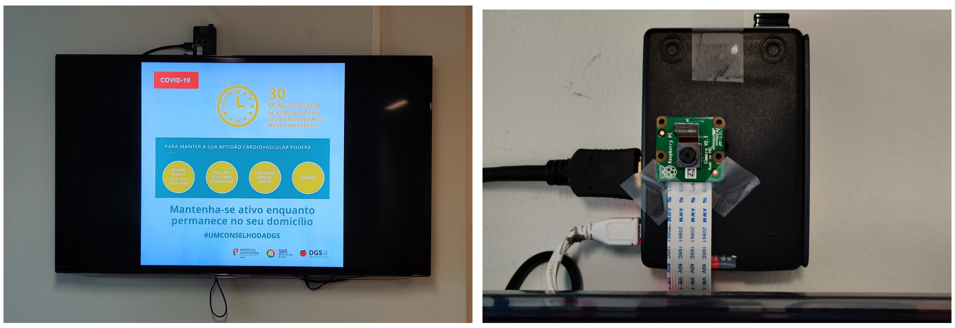

COVID-19
30
PARA MANTER A SUA APTIDÃO CARDIOVASCULAR PODERÁ
Mantenha-se ativo enquanto permanece no seu domicílio
#UMCONSELHODADGS
DGS

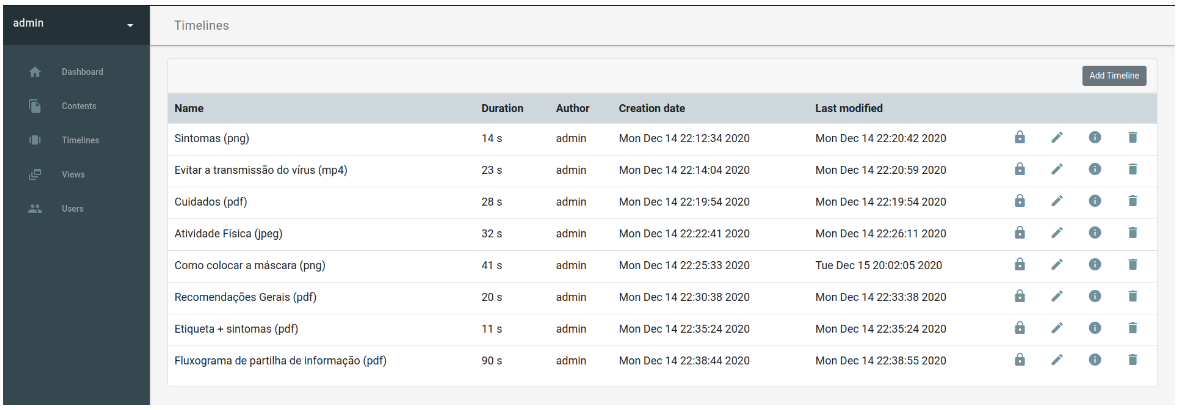
Figure 6.2: Content Display Agent setup at a IEETA monitor.

6.2 Contents

The choice of multimedia contents wasn't to relevant when it comes to the evaluation of the solution. That being said it was a requirement and for that reason and the fact that at the time of this experiment the world was going through the COVID-19 pandemic, it felt positive to populate the system with multimedia contents regarding this subject. As it is commonly understood, having access to accurate and up-to-date information is very important to keep people safe and fight the disease and for this the multimedia contents were collected from the portuguese Health Ministry's COVID-19 website. As there were plenty of individual contents uploaded and most of them are intended to be used together, they fall right into the digital signage solution capabilities. A list of the generated Timelines is presented and Figure 6.3 shows these in the Content Management Dashboard.

- Sintomas (two PNG files);
- Evitar a transmissão do vírus (one MP4 file);
- Cuidados (pdf);
- Atividade física (four JPEG files);
- Como colocar a máscara (8 PNG files);
- Recomendações gerais (one PDF file);
- Etiqueta Respiratória e Sintomas (two PDF files);
- Fluxograma de partilha de informação (one PDF file);

Every View was setup with all the timelines in the same order.



Name	Duration	Author	Creation date	Last modified
Sintomas (png)	14 s	admin	Mon Dec 14 22:12:34 2020	Mon Dec 14 22:20:42 2020
Evitar a transmissão do vírus (mp4)	23 s	admin	Mon Dec 14 22:14:04 2020	Mon Dec 14 22:20:59 2020
Cuidados (pdf)	28 s	admin	Mon Dec 14 22:19:54 2020	Mon Dec 14 22:19:54 2020
Atividade Física (jpeg)	32 s	admin	Mon Dec 14 22:22:41 2020	Mon Dec 14 22:26:11 2020
Como colocar a máscara (png)	41 s	admin	Mon Dec 14 22:25:33 2020	Tue Dec 15 20:02:05 2020
Recomendações Gerais (pdf)	20 s	admin	Mon Dec 14 22:30:38 2020	Mon Dec 14 22:33:38 2020
Etiqueta + sintomas (pdf)	11 s	admin	Mon Dec 14 22:35:24 2020	Mon Dec 14 22:35:24 2020
Fluxograma de partilha de informação (pdf)	90 s	admin	Mon Dec 14 22:38:44 2020	Mon Dec 14 22:38:55 2020

Figure 6.3: Experiment Timelines in the Content Management Dashboard.

6.3 Experiment Guidelines

To better understand the performance of the solution some guidelines were set. Only in this way it is possible to gauge if the obtained results coincide with the expectations. The experiment was designed to last 6 minutes and to have 3 different stages. In each of the stages 4 participants were asked to behave in a certain way.

- **Stage 1** (0 - 2min) - Direct engagement;
- **Stage 2** (2 - 4min) - Sporadic engagement;
- **Stage 3** (4 - 6min) - Engagement with a mask on.

6.4 Results Discussion

For this analysis the data to be considered relates to the View resource level. Having timestamps associated to each entry of extracted data allowed to isolate the data corresponding to the experiment time that is equal to 6 minutes. In Table 6.1 are presented the audience assessment metrics obtained by the system.

	Global	**Stage 1**	**Stage 2**	**Stage 3**
Number of Frames	423	162	136	125
Number of Subjects	5	4	4	5
Attention Time	7m 3s	2m 42s	2m 16s	2m 5s
Attention Coefficient	0.59	0.62	0.51	0.65
Cumulative Attention	251.51	100.66	69.2	81.65

Table 6.1: Data generated by the system for the experiment.

Starting with the number of frames and focusing on the stages, its possible to see that they are registered as expected, with the highest being Stage 1, and the lowest Stage 3, even if not by the amount one would imagine, especially due to the use of the mask. This means the system is capable of detecting subjects even with a part of the face covered, but the success rate drops. That being said, looking at the number of detected subjects, some more insight to this matter can be found. For Stage 1 and 2, 4 different subjects were detected, as expected, but in Stage 3 the system registered a new subject. The introduction of the mask proves challenging for the face recognition algorithm, since it removes some of the facial features extracted by the system.

Since the attention time amounts to the total sum of time of engadgement by all of the subjects, it's no surprise this time exceeds the display time, since there were 4 participants.

When comparing the attention coefficient and cumulative attention values, these are reported as expected, with Stage 2 having the lowest value in both. Furthermore, Stage 3, while reporting a higher value for the attention coefficient when compared with Stage 1, has a lower value for the cumulative attention. This is understandable due to the lower number of captured frames explained by the success rate drop mentioned above.

In Figure 6.4 is provided a chart illustration of the attention coefficient values over the time of the experiment.

In order to help visualize the relevant parameters such as attention coefficient values and emotions some charts were generated. Examples of those charts are provided in Figure 6.4 and .

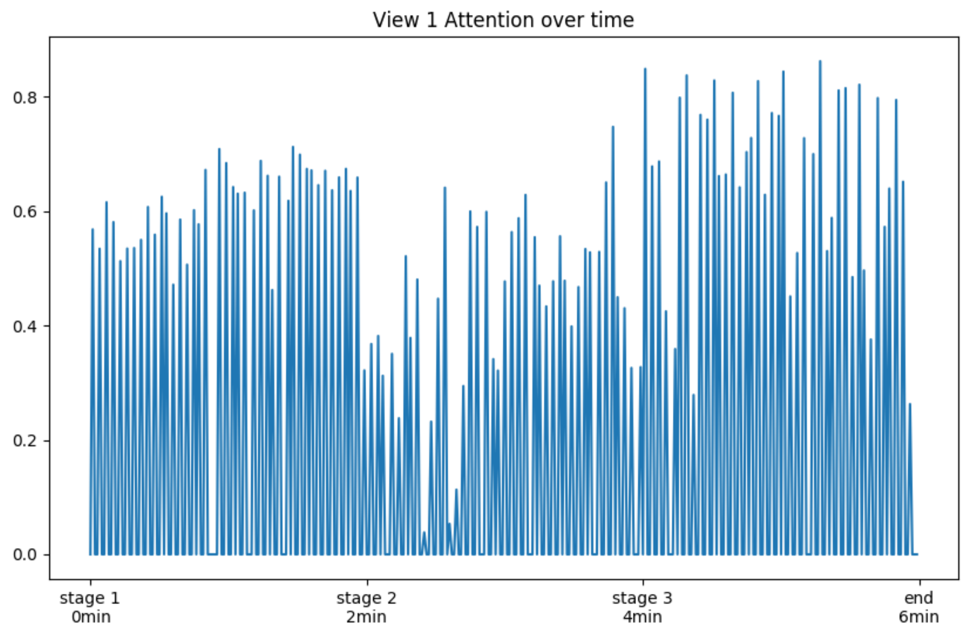


Figure 6.4: Recorded attention values over time for the experiment.

Looking at the data regarding the extracted emotions, overall the system mainly recognized the Neutral emotion, only recording one occurence of a different emotion, Sadness. This can relate to the multimedia contents not being prone to generating reactions from the subjects, but more testing would be necessary to take better conclusions. An illustration of the recorded emotions over the time of the experiment is presented in Figure 6.5.

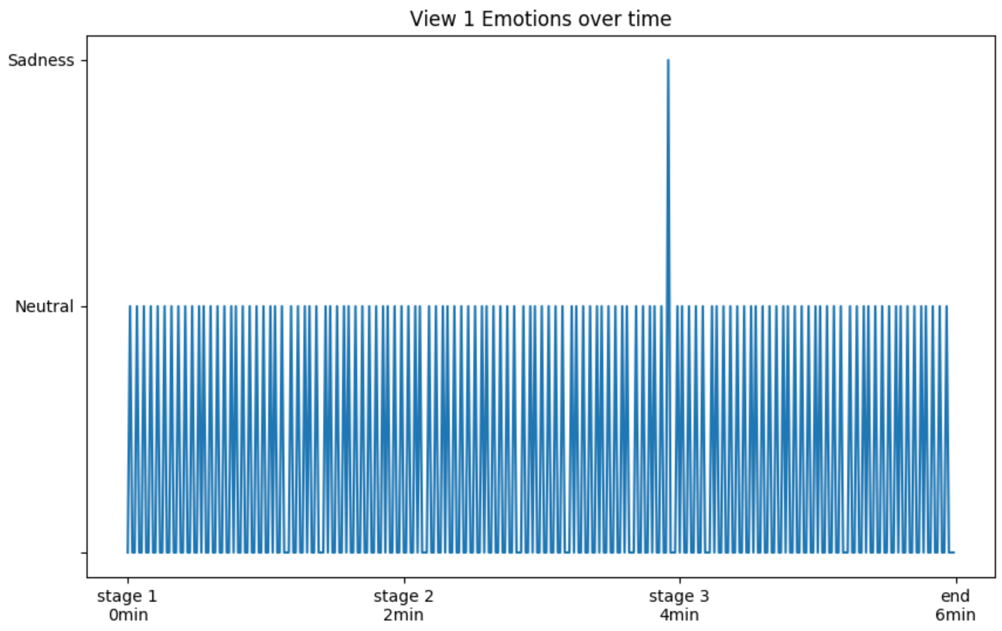


Figure 6.5: Recorded emotions over time for the experiment.

CHAPTER 7

Conclusions

This final chapter aims to provide an overview of the developed solution. The main objective of this dissertation was the implementation and integration of audience assessment features on an existing digital signage system. This required developments in all the components of the project architecture.

Starting with improvements to the Content Management Dashboard, the developments focused on providing responsive design that enables a better user experience when accessing the platform using a mobile device, followed by the introduction of a new page related to every multimedia resource uploaded to and manipulated by the system, which allows for users to review system generated information about the multimedia contents and also the introduction of a preview feature for all of these same resources.

Moving to the Content Display Agents, the implemented software solution was adjusted to handle the new data collection process. For this process, the agent was coupled with a digital camera that allows for capturing the area in front the agents' display. Throughout the time that the agent is displaying multimedia content, it continuously analyses the feed captured by its camera in an attempt to detect viewers, using face detection techniques. When such detection happens, it notifies the server, triggering the operation of the computer vision and data mining algorithms running in the server.

Finally, the Web Server, received a new module that contains all the computer vision features for data extraction and mining. These involve face detection algorithms for facial features extraction, face recognition algorithms for subject identification, emotions recognition for understanding the emotional impact of the multimedia contents, and finally, head pose calculation for measuring the level of engadgement by the subjects.

After all the developments, an experiment was planned and carried out in order to evaluate the correct functioning of the system. All the objectives for this dissertation's project were met. That being said, future work can be done to improve the system, especially relating to the manipulation of the data generated by the system in order to provide users with stronger conclusions about the impact of the multimedia contents. At this point, the system only outputs data for the user himself to interpret and act upon. In the future it could be possible that the system is able to autonomously decide what to adjust in the multimedia contents in real-time.

To conclude, while the system was designed to generate content-centric data, it could easilly be made to generate user-centric data. With this approach it would be possible to track subject interactions over time and understand which mutlimedia contents captured their individual interest.

References

[1] R. R. Burke, “Behavioral effects of digital signage”, *Journal of Advertising Research*, vol. 49, no. 2, pp. 180–185, 2009, ISSN: 0021-8499. DOI: 10.2501/S0021849909090254. eprint: http://www.journalofadvertisingresearch.com/content/49/2/180.full.pdf. [Online]. Available: http://www.journalofadvertisingresearch.com/content/49/2/180.

[2] J. Schaeffler, *Digital Signage: Software, Networks, Advertising and Displays: A Primer for Understanding the Business*. Taylor Francis, 2012, pp. 1, 37, ISBN: 9781136031533. [Online]. Available: https://books.google.pt/books?id=9ZUrt7-igxQC&dq.

[3] *Smarter solutions digital signage products*, https://www.smarter-solutions.ltd.uk/products/digital-signage/, Accessed: December 18, 2020.

[4] *Computer vision and digital signage*, https://core.ac.uk/download/pdf/11678579.pdf, Accessed: January 19, 2021.

[5] R. Ravnik and F. Solina, “Interactive and audience adaptive digital signage using real-time computer vision”, *International Journal of Advanced Robotic Systems*, vol. 10, no. 2, p. 107, 2013. DOI: 10.5772/55516. eprint: https://doi.org/10.5772/55516. [Online]. Available: https://doi.org/10.5772/55516.

[6] *Everything you ever wanted to know about computer vision.* https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e, Accessed: January 19, 2021.

[7] *Article on audience tracking as a trend in advertising*, https://www.knowthis.com/advertising/advertising-trends-audience-tracking/, Accessed: December 18, 2020.

[8] M. Mikusz, A. Noulas, N. Davies, S. Clinch, and A. Friday, “Next generation physical analytics for digital signage”, in *Proceedings of the 3rd International on*

Workshop on Physical Analytics, ser. WPA '16, Singapore, Singapore: Association for Computing Machinery, 2016, pp. 19–24, ISBN: 9781450343282. DOI: 10.1145/2935651.2935658. [Online]. Available: https://doi.org/10.1145/2935651.2935658.

[9] T. Costa, M. Andrade, and P. Viana, "Predictive multi-view content buffering applied to interactive streaming system", 2019. [Online]. Available: https://doi.org/10.1049/el.2019.0713.

[10] P. Tian, A. V. Sanjay, K. Chiranjeevi, and S. M. Malik, "Intelligent advertising framework for digital signage", in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2012, pp. 1532–1535, ISBN: 9781450314626. [Online]. Available: https://doi.org/10.1145/2339530.2339773.

[11] K. Nagao and I. Fujishiro, "Understanding viewers' involuntary behaviors for adaptive digital signage", in *Proceedings of the ACM Symposium on Applied Perception*, ser. SAP '13, Dublin, Ireland: Association for Computing Machinery, 2013, p. 145, ISBN: 9781450322621. DOI: 10.1145/2492494.2501903. [Online]. Available: https://doi.org/10.1145/2492494.2501903.

[12] *Signbox official website*, https://signbox.tv, Accessed: January 19, 2021.

[13] *Signbox's signeye*, https://signbox.tv/digital-signage-products/digital-signage-facial-recognition, Accessed: January 19, 2021.

[14] *Seemetrix official website*, https://seemetrix.net/, Accessed: January 19, 2021.

[15] A. F. Batista, "Digital management of multiple advertising displays", pp. 1, 28, 2018. [Online]. Available: http://hdl.handle.net/10773/25886.

[16] *Official django website*, https://www.djangoproject.com/, Accessed: December 19, 2020.

[17] *Full stack python website*, https://www.fullstackpython.com/django.html, Accessed: December 19, 2020.

[18] *Article on mvc design pattern in django*, https://medium.com/shecodeafrica/understanding-the-mvc-pattern-in-django-edda05b9f43f, Accessed: December 19, 2020.

[19] *Python imghdr*, https://docs.python.org/3/library/imghdr.html, Accessed: December 19, 2020.

[20] *Python pypdf2*, http://mstamy2.github.io/PyPDF2/, Accessed: December 19, 2020.

[21] *Python pdf2image*, https://github.com/Belval/pdf2image, Accessed: December 19, 2020.

[22] *Python python-pptx*, https://github.com/scanny/python-pptx, Accessed: December 19, 2020.

[23] *Python ffmpy*, https://github.com/Ch00k/ffmpy, Accessed: December 19, 2020.

[24] *Python subprocess*, https://docs.python.org/3/library/subprocess.html, Accessed: December 19, 2020.

[25] *Ffmpeg official website*, https://ffmpeg.org/, Accessed: December 19, 2020.

[26] D. Rodriguez, "Introduction to audiovisual transcoding, editing, and color analysis with ffmpeg", *The Programming Historian*, vol. 7, 2018. DOI: 10.46430/phen0077.

[27] *Article on face recognition in law enforcement*, https://www.eff.org/pages/face-recognition, Accessed: December 19, 2020.

[28] *Article on trends in face recognition technology*, https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/biometrics/facial-recognition, Accessed: December 19, 2020.

[29] D. Canedo, "Focus estimation in academic environments using computer vision", 2019. [Online]. Available: https://doi.org/10.1007/978-3-030-31332-6_54.

[30] *Dlib official website*, http://dlib.net/, Accessed: December 20, 2020.

[31] *Onnx runtime official website*, https://www.onnxruntime.ai/about.html, Accessed: December 20, 2020.

[32] *Opencv official website*, https://opencv.org/about/, Accessed: December 20, 2020.

[33] *Django databases documentation*, https://docs.djangoproject.com/en/3.1/ref/databases/, Accessed: December 19, 2020.

[34] *Sqlite official website*, https://www.sqlite.org/index.html, Accessed: December 19, 2020.

[35] *Matplotlib official website*, https://matplotlib.org/, Accessed: December 20, 2020.

[36] *Raspberry pi 3 model b+*, https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/, Accessed: December 20, 2020.

[37] *Raspberry pi camera module v2*, https://www.raspberrypi.org/products/camera-module-v2/, Accessed: December 20, 2020.

[38] *Raspberry pi official website*, https://www.raspberrypi.org/about/, Accessed: December 20, 2020.

[39] M. Maksimovic, V. Vujovic, N. Davidovic, V. Milosevic, and B. Perisic, "Raspberry pi as internet of things hardware: Performances and constraints", 2014. [Online]. Available: https://www.researchgate.net/publication/272175660_Raspberry_Pi_as_Internet_of_Things_hardware_Performances_and_Constraints.

[40] *Raspberry pi model 3 b+*, https://www.raspberrypi.org/poducts/raspberry-pi-3-model-b-plus/, Accessed: January 27, 2020.

[41] *Python screeninfo*, https://github.com/rr-/screeninfo, Accessed: December 19, 2020.

[42] *Python netifaces*, https://github.com/al45tair/netifaces, Accessed: December 19, 2020.

[43] *Python requests*, https://github.com/psf/requests, Accessed: December 19, 2020.

[44] *Omxplayer github*, https://github.com/popcornmix/omxplayer, Accessed: December 20, 2020.

[45] *Raspberry pi documentation on omxplayer*, https://www.raspberrypi.org/documentation/raspbian/applications/omxplayer.md, Accessed: December 20, 2020.

[46] *Pi camera python package*, https://picamera.readthedocs.io/, Accessed: January 27, 2020.

[47] *Django's authentication system documentation*, https://docs.djangoproject.com/en/3.1/topics/auth/, Accessed: December 26, 2020.

[48] *Django's file upload documentation*, https://docs.djangoproject.com/en/3.1/topics/http/file-uploads/, Accessed: December 26, 2020.

[49] *Libreoffice official website*, https://www.libreoffice.org/, Accessed: December 26, 2020.

[50] *Javascript official website*, `https://www.javascript.com/`, Accessed: December 26, 2020.

[51] *Django models documentation*, `https://docs.djangoproject.com/en/3.1/topics/db/models/`, Accessed: December 26, 2020.

[52] *Bootstrap official website*, `https://getbootstrap.com/`, Accessed: December 27, 2020.

[53] *Bootstrap's card class documentation*, `https://getbootstrap.com/docs/5.0/components/card/`, Accessed: December 27, 2020.

[54] *Bootstrap's columns class documentation*, `https://getbootstrap.com/docs/5.0/layout/columns/`, Accessed: December 27, 2020.

[55] *Python documentation for threading module*, `https://docs.python.org/3/library/threading.html`, Accessed: December 22, 2020.

[56] *Blog post about dlib's hog based face detection*, `https://medium.com/mlcrunch/face-detection-using-dlib-hog-198414837945`, Accessed: January 27, 2020.

[57] *Blog post about data extraction*, `https://www.alooma.com/blog/what-is-data-extraction`, Accessed: December 23, 2020.

[58] *Blog post about data mining*, `https://www.sas.com/pt_pt/insights/analytics/data-mining.html`, Accessed: January 27, 2020.

[59] *Pre trained models for use with dlib*, `https://github.com/davisking/dlib-models`, Accessed: January 27, 2020.

[60] E. Barsoum, C. Zhang, C. C. Ferrer, and Z. Zhang, "Training deep networks for facial expression recognition with crowd-sourced label distribution", in *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, ser. ICMI '16, Tokyo, Japan: Association for Computing Machinery, 2016, pp. 279–283, ISBN: 9781450345569. DOI: `10.1145/2993148.2993165`. [Online]. Available: `https://doi.org/10.1145/2993148.2993165`.