



**Arménio
Ferreira Baptista**

**Gestão Digital de Múltiplos Monitores de
Publicidade**

**Digital Management of Multiple Advertising
Displays**



**Arménio
Ferreira Baptista**

**Gestão Digital de Múltiplos Monitores de
Publicidade**

**Digital Management of Multiple Advertising
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 Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e de Alina Liliana Trifan, Bolseira Pós-Doutoramento do Instituto de Engenharia Eletrónica e Informática de Aveiro da Universidade de Aveiro.

Dedico este trabalho à minha mãe pelo incansável apoio e encorajamento durante estes cinco anos.

o júri / the jury

presidente / president

Prof. Doutor Paulo Miguel de Jesus Dias

Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Luís Filipe Pinto de Almeida Teixeira

Professor Auxiliar do Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor António José Ribeiro Neves

Professor Auxiliar da Universidade de Aveiro

**agradecimentos /
acknowledgements**

Um agradecimento especial para os meus orientadores, Professor Doutor António Neves e Alina Trifan, por todo o apoio, disponibilidade e conhecimento que me proporcionaram e transmitiram, a qual nada disto seria possível sem eles.

À minha família o mais sincero obrigado pelo apoio e motivação que me deram, não só na elaboração desta tese, mas também ao longo do meu percurso universitário, sempre acreditando nas minhas capacidades e decisões. Aos meus colegas e sempre amigos, um agradecimento por todos os momentos que me proporcionaram e espírito de entreatajuda ao longo desta nossa etapa.

Palavras Chave

Adaptação de conteúdos, *Digital signage*, Gestão de conteúdos digitais, Multimédia, Plataforma baseada em núvem, Publicidade.

Resumo

A explosão tecnológica que temos experienciado na última década tem tido impacto sobre o setor do retalho de várias formas. Cativar o público-alvo através de estratégias de publicidade, no processo de retalho e aprimorando a sua experiência tem sido um desiderato nesta indústria há bastante tempo. A tecnologia recente possibilita seguir abordagens nunca antes vistas para atingir estes objetivos. Nesta dissertação, é apresentada uma solução baseada em várias estações autónomas (sejam estáticas, como monitores, ou até móveis, como robôs autónomos) que podem ser usadas em qualquer publicidade de conteúdos multimédia de uma ou várias entidades. A parte central da solução apresentada é um servidor Web capaz de armazenar os conteúdos carregados, que expõe uma página web para a sua gestão. Utilizadores registados podem gerir e distribuir os conteúdos pelos terminais conectados (ou agentes), sendo o único requisito uma ligação de rede entre o servidor e os agentes. Apresentamos, também, resultados deste sistema dentro de vários eventos de investigação que tiveram lugar no ambiente académico local. O sistema foi usado com o objetivo de automatizar a disseminação e publicitação de trabalhos de investigação locais. Por fim, é expectável extender o seu uso no setor do retalho numa tentativa de impactar a publicidade moderna.

Keywords

Advertising, Content adaptation, Cloud-based platform, Digital Contents Management, Digital signage, Multimedia.

Abstract

The technological boom that we have been experiencing in the last decade has impacted the retail sector in several ways. Captivating customers through smart advertising, engaging them in the retail process and enhancing their experience has been a long-time desideratum in this industry. Recent technology makes it possible to follow unprecedented approaches for achieving these goals. In this thesis, we present a solution based on a series of autonomous stations (either static, such as monitors, or mobile, such as autonomous robots) that can be used in any type of multimedia advertising across one or multiple entities. The main core of the presented solution is a Web Server that can store the uploaded contents, which exposes a web dashboard for their management. Registered users can manage and distribute the contents through the connected terminals (or agents), being the only requirement a network connection between the server and the agents. We present results of this system within several research events that took place within the local academic environment. The system was used with the aim of automatizing the dissemination and advertising of local research works. Ultimately, we expect to extend its use to the retail sector in an attempt to impact modern advertising.

Contents

Contents	i
List of Figures	iii
List of Tables	v
Glossary	vii
1 Introduction	1
1.1 Thesis structure	2
2 Related work	3
2.1 Similar systems	4
3 Architecture	7
3.1 Web Server	7
3.1.1 Flask	8
3.1.2 Django	8
3.1.3 Decision	8
3.2 Multimedia contents transformation	9
3.2.1 OpenCV	9
3.2.2 Python Imaging Library (PIL)	9
3.2.3 MoviePy	10
3.2.4 FFMpeg	10
3.2.5 Decision	10
3.3 Database	10
3.3.1 SQLite	10
3.3.2 MySQL	11
3.3.3 MPEG-21 format	11
3.3.4 Decision	11
3.4 Other support Python libraries	11

3.5	Files format support	12
3.5.1	Image formats support	12
3.5.2	Video formats support	13
3.5.3	Presentation formats support	13
3.6	Permissions	13
4	Digital Management of Multimedia Contents	15
4.1	Resources division	15
4.2	Users permissions	16
4.2.1	Resource level	17
4.2.2	Object level	17
4.3	Web server	18
4.3.1	Files upload and validation	18
4.3.2	Portable Document Format (PDF) transformations	18
4.3.3	Power Point transformations	19
4.3.4	Timelines creation	19
4.3.5	Views creation	19
4.4	Dashboard	20
4.5	Multimedia contents processing	21
4.6	Agents	23
4.6.1	Life cycle	23
4.7	Web Server and Raspberry Pi communication	25
4.8	Database implementation	26
5	Case studies	29
5.1	Case studies	29
5.1.1	Institute of Electronics and Informatics Engineering of Aveiro (IEETA) monitors	29
5.1.2	Students@DETI	30
6	Conclusion	33
6.1	Future work	33
	References	35

List of Figures

1.1	IEETA monitor demonstration.	2
2.1	Digital signage example in Times Square, New York.	3
2.2	Yodeck digital signage solution.	4
2.3	JCDecaux billboard.	5
3.1	System architecture.	7
3.2	Flask and Django popularity in StackOverflow questions.	9
4.1	Resources composition.	16
4.2	Users permissions.	16
4.3	User creation web page.	17
4.4	Modification of object level permissions of a Content.	18
4.5	Dashboard Views web page with an unconfigured View.	19
4.6	Dashboard home example.	20
4.7	Visualization of Contents, Timelines and Views.	21
4.8	Edition of a Timeline.	21
4.9	Pillarboxing and letterboxing effects.	22
4.10	Types of aspect ratios.	22
4.11	Monitor life cycle.	24
4.12	Raspberry Pi terminal output. It is visible, from the output, that the video was successfully downloaded.	25
4.13	Database object-relational mapping diagram.	27
5.1	Final result of a monitor in IEETA.	30
5.2	Demonstration and exhibition of the system at Students@DETI, using a vertical monitor.	31

List of Tables

3.1	Python libraries used in this work.	12
-----	---	----

Glossary

PDF	Portable Document Format	HTML	Hypertext Markup Language
OpenCV	Open Source Computer Vision Library	CSS	Cascading Style Sheets
JPEG	Joint Photographic Experts Group	sTIC	Serviços de Tecnologias de Informação e Comunicação
PNG	Portable Network Graphics	VPN	Virtual private network
PDF	Portable Document Format	IEETA	Institute of Electronics and Informatics Engineering of Aveiro
AVI	Audio Video Interleave	ACL	Access Control List
PIL	Python Imaging Library	RBAC	Role-based Access Control
SQL	Structured Query Language	HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator		
CSRF	Cross-site request forgery		

Introduction

One of the keys to the success of the retail industry passes by advertising to the general public, mainly by using marketing strategies [1]. The technological evolution, more and more, has an important role in the marketing and the advertise of products [2]. In [3], experimental results show that sales in supermarkets are enhanced when digital displays are used.

Additionally, the technological evolution also opened doors which would never be possible some decades ago, not only due to computing speed, but also due to cheaper hardware. This progress allows the companies to take a chance on digital advertising, by using digital components instead of physical posters.

Not only the digital advertising is much more appealing to the general public, making the impact of the contents much bigger, but also the maintenance of these terminals can be smaller than the physical ones.

This thesis emerged from the need to have a distribution system of multimedia contents in order to display them in a series of monitors, being the two main motivations the IEETA monitors used for research dissemination and the academic conferences hosted in the University of Aveiro which usually make use of monitors to display some contents and projects. These are the two main applications of the system, although, the system is portable to use in any department of the University due to the Eduroam network. Also, the system is useful to be used in any occasion needing the advertisement of contents.

In addition to the main motivations of this thesis, the applications of the system can reach the retail or any area needing advertisement.

Concerning the IEETA monitors, in Figure 1.1 is presented an application of the system in a monitor hosted in IEETA. The monitors are used to display contents to welcome the visitors of the institute. The contents were being displayed using a USB flash drive directly connected to each monitor, which by itself can be a headache to update or upload new contents, forcing the manual configuration of each monitor.

Regarding the conferences, the monitors are used to display some e-posters in the coffee break zone, allowing the visitors to read and present the papers between oral presentations.



Figure 1.1: IEETA monitor demonstration.

We also tested the use of an Android App to control the monitors, giving the users the possibility to interact with the posters, which is a valuable feature in these cases. The constant requests from the responsible of the events for a way to display contents in monitors lead us to this proposal.

Therefore, the main focus of this thesis is to propose a solution to control multimedia resources and display them using a unique platform, which provides the management and manipulation over the resources in order to produce a final content which fits the screen resolution of the monitor associated to a terminal, making the appropriate transformations, and distribute them across a network of terminals.

This solution allows the control, maintenance, composition and division of the multimedia resources across the stations, displaying the information that the user selects into the different terminals. It also proposes a solution to control the permissions to the resources for each user logged in the platform.

1.1 THESIS STRUCTURE

This thesis is divided into six major chapters, Introduction and Conclusion included. The remaining chapters are the following:

- Related work - includes existent similar systems and approaches the term Digital Signage.
- Architecture - includes the main requisites of the system, some technological tools and decisions made regarding these tools.
- Digital Management of Multimedia Contents - includes the development process of the system.
- Case studies - includes results and case studies used to test the system.

Related work

The core concept of this distribution advertising system can be compared to some systems or approaches used in the present. In the past, most of the systems used a manual configuration to display the contents, being necessary for a person to move to the location and place the content in the display.

However, the technological evolution allowed the companies to adopt new and more advanced and appellative ways of advertising. Due to this growth, several areas and terms emerged, being the term Digital Signage the one that fits this proposal.

The term Digital Signage is widely used and is present in our day by day. It is defined as remotely managed digital display typically tied in with sales, advertising and marketing [4]. It can also be defined as a network of electronic displays that are centrally managed and individually addressable for the display of text, animated or video messages for advertising, information, entertainment and merchandising to targeted audiences.

There are several systems and products supporting this technology as we can see in most of the existent advertisement displays presented in a wide area of establishments, as illustrated in Figure 2.1, such as: airports, food chains, outdoor sites, shopping malls, etc.



Figure 2.1: Digital Signage example in Times Square, New York.

2.1 SIMILAR SYSTEMS

In this section, some similar systems are presented having into consideration the similarity and application of our proposed system. Although, every presented system belongs to companies specialized in building Digital Signage solutions as a product. Therefore, all these solutions are paid and the implementation behind them are unknown.

Yodeck

Yodeck [5] is the most similar system compared to our project. Their system consists on a software solution to manage a series of monitors using only their product: a Raspberry PI called "Yodeck Playbox", which brings a MicroSD card containing proprietary Yodeck software. The user only needs to buy the Raspberry PI and connect it to a monitor, create an account, upload the multimedia contents to the platform and associate these contents to a virtual monitor. Their solution divides the contents into 4 categories:

- Media - contents uploaded to the system, like image files, video files, documents or web site addresses.
- Playlists - set of media with a predefined duration to create an image slideshow or a video
- Widgets - small widgets that display useful info, like RSS feed, a clock, etc.
- Shows - set of media, playlists and widgets which can be associated to a virtual monitor (i.e. a Raspberry PI).



Figure 2.2: Yodeck digital signage solution [6].

Xarevision

Xarevision [7] is a leading company in technologies for retail and operates the biggest in-store digital network in Portugal, reaching over 40% of the active population. It administers broad digital networks of centrally managed displays and also create differentiated interactive media, like gesture recognition, georeferencing and individual addressing.

Their technologies apply mainly in the retail industry and offer three products:

- Queue management - this product offers management solutions to apply, for example, in supermarkets. They are present in almost big surfaces and are used to optimize waiting times and increase the customer service. Generally, it is composed of a ticket dispenser and a customer calling screen.

- New media - features like gesture recognition, georeferencing and individual addressing. And also the remote control and supervision of the distributed networks across the country.
- Digital Signage and Corporate TV - last but not least, the digital signage solution, which applies to this thesis. Their solution passes by supporting the client from the begging to the end of the implementation, providing the installation of hardware plus network management, supporting content creation and updating, as well as uninterrupted technical monitoring during the project's entire lifetime.

JCDecaux

JCDecaux [8] is probably the most recognized advertising company in the world in outdoor sites. Despite there isn't any specific detail about the development and implementation of their terminals, it is interesting to refer it because of the distributed advertising system they possess.

Their main products are the billboards presented in some bus stops, although they also have some digital products presented mainly in some shopping malls. This project aims to produce a distributed system similar to these digital billboards, being an autonomous one.

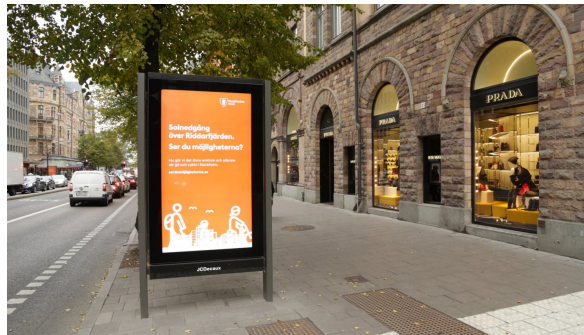


Figure 2.3: JCDecaux billboard [9].

Enplug

Enplug [10] is a digital signage software company that provides the disclosure of contents. It supports social contents, like Youtube and Instagram feeds, RSS feeds, digital menus, Google calendars, graphics and videos.

NoviSign

NoviSign [11] is a digital signage software company with a cloud-based solution with Windows, Android and Chrome OS players support. It has an online control dashboard - Studio, allowing the creation, edition and management of digital contents from any browser, as well as updating the screens in real time. It supports a variety of widgets, like text, images, video, RSS, games, polls, social widgets and many more. Also, it may be integrated with camera (face recognition), RFID reader, barcode and sensors for triggering ads/events/slides.

Mvix

Mvix [12] offers digital signage systems with a portable physical module, similar to a Raspberry Pi [13], which has a pre-loaded content management software capable of communication with a cloud-based server. Also, it offers a web-based software for creating, editing and updating digital signage templates. It includes content apps and widgets which enable users to display a large variety of content including HD videos and images, live web pages, RSS feeds, media animations and more. This digital signage solution also supports playlist management, comprehensive calendar-based scheduling, and a multi-role user management.

ScreenCloud

ScreenCloud [14] is a digital signage solution that can turn almost any screen to display multimedia contents. It offers compatibility with a large variety of devices, such as: Google Chromebit, Amazon Fire TV, Google Chromebox, Android TV, Apple Mac Mini, Mi Box, etc. But it also offers compatibility to display contents in iPads and tablets. The scheduling, management and upload of contents is made using a Web Browser.

Architecture

Having three main groups of agents (web server, terminals and control dashboard) with the need to communicate between them in order to control, manage and display the contents, the architecture illustrated in Figure 3.1 emerged.

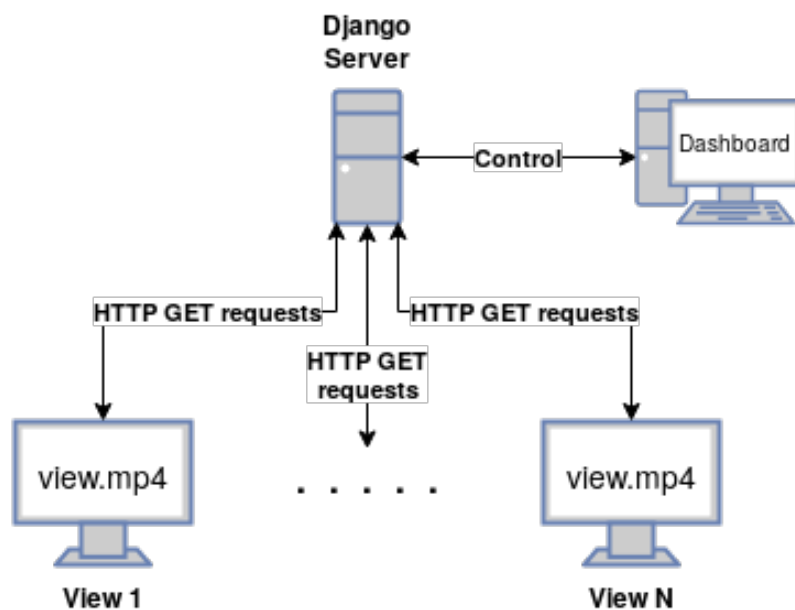


Figure 3.1: System architecture.

This architecture makes the web server the main core of our system, being the central point of communication and control.

3.1 WEB SERVER

When confronted with the development of a multimedia distributed system, we decided to use a web framework. The main advantage of this decision was the applications that a web server can have in our system, mainly the management of the multimedia resources by regular users of the system.

When building a web interface using a web framework, there are several aspects to count, like the database to use, the way to expose the interface, the session management or even the code reuse.

In order to expose the web interface and the need to implement a Web Server and also due to our experience with Python, it was considered the use of two different web frameworks, both free and open source.

3.1.1 Flask

Flask [15] is a lightweight and versatile web framework. Lightweight in the way that does not need much effort to expose a web interface and versatile in that it gives a lot of freedom to the users to adapt it to their needs and expand it through a series of available applications.

3.1.2 Django

Unlike Flask, Django [16] has a predefined design of development which the developers have to follow in order to deploy a web server. Django acts like a quick solution for web development [17], making use of its predefined design to deliver a fast way to develop the idea directly to the application.

Django's scalability allows the developers to add, remove, combine and share applications between different projects, making the process of web development pretty fast by reusing applications already developed.

Also, Django provides security mechanisms in order for the developers to focus more on implementation and less on security issues: SQL injection, cross-site request forgery, clickjacking and cross-site scripting [17], as well as an embedded administration dashboard to control the database models and users authentication.

3.1.3 Decision

Django is one of the most powerful web frameworks and by being released earlier than Flask (2005 and 2010, respectively), his popularity and community support is much higher, as illustrated in Figure 3.2.

Both web frameworks have a steady growth and offer a great option for building a web platform. Since the main focus of the system is the manipulation of the multimedia contents, we wanted a web framework with a straight-forward way solution.

Django provides this solution, having an obvious way to develop a web server, allowing us to focus more on the main goal. Also, the administration dashboard proved to be very useful to test the system and the documentation is a plus.

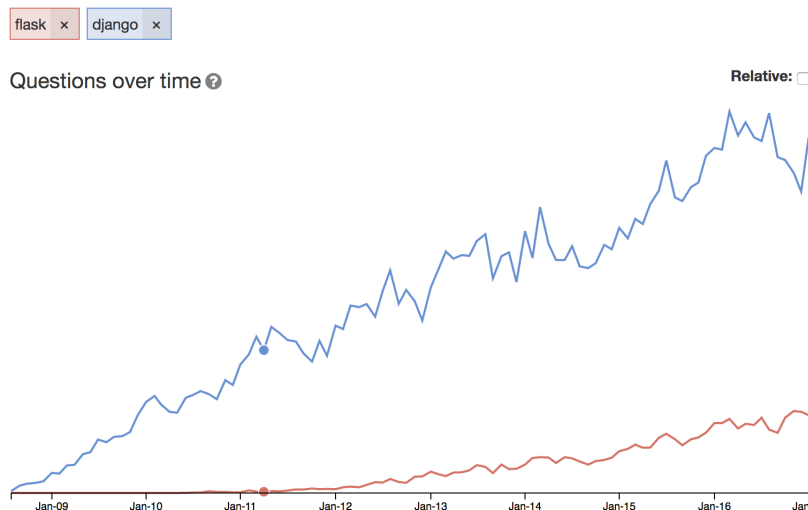


Figure 3.2: Flask and Django popularity in StackOverflow questions [18]

3.2 MULTIMEDIA CONTENTS TRANSFORMATION

Developing a system with the capability to transform the uploaded multimedia contents requires the use of libraries capable of manipulating these contents, mainly through the manipulation of frames from the contents (i.e. images, videos, PDFs).

The purposes of this manipulation is the need to convert the files between formats to a predefined one, better fit the resolution and aspect ratio of the frames to the monitor and to merge all the contents into one final video to be displayed in the monitors.

In order to do this, we considered the following described during the next sections.

3.2.1 OpenCV

Open Source Computer Vision Library (OpenCV) is an open source computer vision and machine learning software library [19]. It has a large amount of implemented algorithms in several areas, although the part that concerns our project is the image and video manipulation. It allows us to transform the frames of the images or videos, create videos from several contents and save the final content into many formats.

3.2.2 Python Imaging Library (PIL)

Python Imaging Library (PIL) [20] is a Python library with image processing capabilities and a large image formats compatibility, being the last commit dated to 2011. This library became deprecated mainly because of the Python version 3 incompatibility.

Although, Alex Clark and some contributors forked the PIL library and extended some functionalities in order to create Pillow [21], providing the compatibility with Python version 3.

The main reason that made us exclude the usage of this library was it only supports image files and, to use it, we would need another library to manipulate the videos.

3.2.3 MoviePy

MoviePy [22] is a Python library with video processing capabilities. It allows cutting, concatenations, title insertions, video compositing, video processing and creation of custom effects.

This library would fit our project but, due to the missing transformation of the video frames (i.e. resizing and padding), it was eventually excluded for most of the tasks. However, it was used to validate the uploaded videos.

3.2.4 FFmpeg

FFmpeg is the leading multimedia framework to handle audio and video files, able to decode, encode, transcode, mux, demux, stream, filter and play [23] almost any multimedia content. The FFmpeg command line tool is a very fast video and audio converter which is capable of receiving several input files (or streams) and apply the desired transformations. It also accepts image files as input, which is an important aspect in our system.

When applied to our system, it can be very useful in the transformation of the contents, not only by supporting a very large number of formats and codecs, but also applying padding and scaling effects and concatenation.

3.2.5 Decision

After analyzing the mentioned tools, we opted to use FFmpeg. Besides the support of almost every video and image format, it also allows the concatenation, padding and scaling effects with the use of a command.

For example, the process of fitting a certain resolution is automatic, being only need to specify the desired resolution and it handles and transforms the contents to better fit the desired one. Also, the software allows to fully customize the desired operation.

The process can be somehow slow if the amount of frames and their transformations involves a lot of steps, however, the easy to use execution and the automatic process makes it worth it.

3.3 DATABASE

Our system needs to store data associated to the multimedia contents, but also the informations about the users and respective permissions. As the information of the multimedia contents are associated to each other, we decided to use a relational database management system (RDBMS), which facilitates fetching data due to the relations of the objects.

Due to our experience using Structured Query Language (SQL), we opted for the use of a database engine with embedded SQL support. So, we considered the following relational database engines, all supported by Django.

3.3.1 SQLite

SQLite [24] is a database engine with embedded SQL with great speed and reliability. It focuses in keeping a tradeoff between memory usage and speed. This database engine differs

mainly from the others by the way the database is stored - in a file. This characteristic enables the portability of the database only by copying the file to other file system.

3.3.2 MySQL

MySQL [25] is also a database engine with embedded SQL, although MySQL provides a database server in order to operate over the data. The high usage of this database makes it a good option because of the support it can offer.

3.3.3 MPEG-21 format

ISO/IEC 21000 (MPEG-21) [26] defines an open framework for multimedia delivery and consumption, with both the content creator and content consumer as focal points. The vision for MPEG-21 is to define a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities.

The MPEG-21 Multimedia Framework is based on two essential concepts: the definition of a fundamental unit of distribution and transaction (the Digital Item) and the concept of Users interacting with Digital Items. The Digital Items can be considered the "what" of the Multimedia Framework (e.g. a video collection, a music album) and the Users can be considered the "who" of the Multimedia Framework.

It is interesting to refer this framework due to the application it has in this thesis. The framework provides mechanisms to identify Digital Items, IP related to the Digital Items and Description schemes, using identifiers to link the Digital Items, which defines the Digital Item Declaration (DID).

3.3.4 Decision

The usage of MPEG-21 was discarded mostly due to the fact that we wanted a video as a final product and MPEG-21 is used mostly to define mechanisms to store contents. So, the decision of which database to use was not one of the most important decisions to take due to the portability that Django offers. However, we chose to use SQLite3 because of the database being saved in a file, making the debug and test of the system easier and the needlessness of installing a database engine in the server.

3.4 OTHER SUPPORT PYTHON LIBRARIES

When using Python as a programming language, one of the main advantages are the support libraries associated, easy to install and to use. In the following table (Table 3.1) are listed the most relevant Python libraries used in the project. Each one of them have a specific task in the system.

Python Library	Description
imghdr [27]	Determines the type of image contained in a file or byte stream. It is used in this project to validate when a file with an image extension is uploaded (.jpeg or .png).
PyPDF2 [28]	Designed specially to operate over a PDF file. It is used in this project to validate when a file with PDF extension is uploaded.
pdf2image [29]	Reads a PDF file and converts it into a PIL Image object. It is used to convert the PDF files to image, to be later converted in a slideshow video.
python-pptx [30]	It is an handler for Microsoft PowerPoint files (.pptx), that allows to create and modify the presentations. Although, in our project, it is only used to validate the uploaded files.
ffmpy [31]	It is a command line wrapper to FFmpeg [23]. It uses the Python subprocess [32] module to execute the commands.
screeninfo [33]	It is a module to fetch the location and the resolution of the physical screens connected. In our project, it is used to fetch the resolution of the screens connected to the Raspberry Pi devices.
netifaces [34]	It is a module to fetch all the interfaces connected to the local device. So, it is used in this project to find a suitable MAC address of the Raspberry PI.
requests [35]	It is a module designed to send HTTP requests. In our system, it is useful to send the HTTP GET and HTTP POST requests from the Raspberry PI to the server.

Table 3.1: Python libraries used in this work.

3.5 FILES FORMAT SUPPORT

One of the essential requisites of the system is the capacity to upload, store and transform the files to be displayed later. The compatibility of the system with the formats of the files needs to be considered, not only to validate the uploaded file, but also to assure the compatibility with the library chosen to operate over these files.

We considered the upload of three types of files: image, video and presentation files. These types grant a large range over the needs of the users by creating slideshows with predefined duration with the images and presentations and also by concatenating videos, creating the final video product.

3.5.1 Image formats support

For images, we opted for two of the most used image formats:

- Joint Photographic Experts Group (JPEG) [36] (.jpeg) - JPEG is a lossy based image format, but also supports lossless compression.
- Portable Network Graphics (PNG) [37] (.png) - PNG is a lossless based image format.

Both formats are used at a worldwide scale and cover a wide range of contents.

3.5.2 Video formats support

For videos, the system supports two video formats, being both a multimedia container format which may contain both audio and video data:

- Audio Video Interleave (AVI) [38] (.avi) - AVI is a multimedia container format created by Microsoft.
- MP4 [39] (.mp4) - MP4 is also a multimedia container and one of the most used video formats.

All the video codecs supported by FFmpeg can be used (ex. H.264 [40], H.265 [41], just to name a few).

3.5.3 Presentation formats support

In the case of the presentation formats, we decided to support PDF and PowerPoint files (.ppt and .pptx extensions) due to the large usage. The developed system supports files with multiple pages (or slides).

3.6 PERMISSIONS

When confronted with the idea of multiple user access and different contents uploaded from each user, the first barrier to break down was how to block certain users from changing the contents from others. Secondly, how to define these permissions and who should change these permissions.

The first approach was to define roles for each user, with each role having certain predefined permissions, following a Role-based Access Control (RBAC) [42]. The second approach was to define permissions over the objects created, following an Access Control List (ACL). This approach will be later explained in the Section 4.2.

Digital Management of Multimedia Contents

This chapter exposes the solution of the system and the decisions made during the development and implementation, focusing also on the technical part. This chapter is based on the article that resulted from the development of this thesis [43].

4.1 RESOURCES DIVISION

One of the first challenges of this project was how to divide the uploaded multimedia contents and how to associate them to different monitors. Another challenge was how to reuse these contents or set of contents in order to create different sequences.

So, in order to organize these contents, we decided to create three different components, each one with a defined purpose:

- Contents - base element of the multimedia resources, which is basically an item uploaded by a user (i.e. a video, image, presentation, etc.).
- Timelines - set of Contents with a predefined sequence, much similar to a video composed of different Contents. If the Content is an image or a presentation, the user may define the duration for each image or slide to be presented in the Timeline.
- Views - set of Timelines with a predefined sequence to be displayed. This component is associated to a physical terminal and represents it in the system. The only way to create a View is with the connection of a terminal.

This division between the resources gives freedom to the users to create and dispose multimedia contents into any order and consequently display them. It also allows the users to reuse the Timelines in different Views without the need to recreate them again, as illustrated in Figure 4.1.

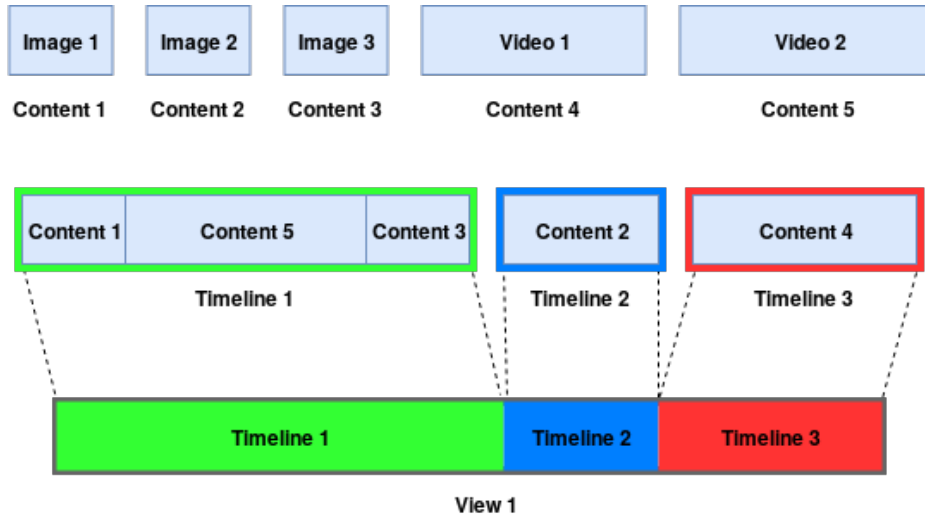


Figure 4.1: Resources composition.

4.2 USERS PERMISSIONS

Having a system with access for a variety of users needs a control policy in order to keep the privacy and discretion of the multimedia contents between users, blocking the modification of resources not owned by that user.

In order to restrict the improper access of the users to the multimedia resources, an authentication system is proposed with different credentials and permission levels for the users. Therefore, before accessing the control dashboard, the system asks for the authentication credentials of the user. Django [16] has an user authentication system, which facilitated the implementation of the authentication system.

The system has two main levels of permissions: role based permissions and users permissions, illustrated in Figure 4.2.

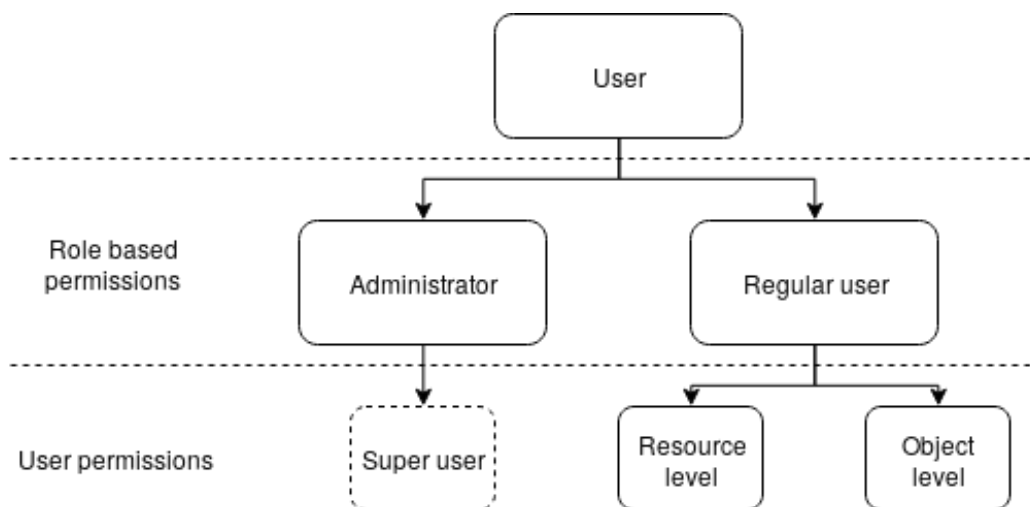


Figure 4.2: Users permissions.

The role based permissions (Section 3.6) has two roles: administrator and regular user.

The administrator has access and control over every resource and access to a section in the dashboard that allows the management of the regular users (creation, edition, deletion and permissions). So, the system has two types of users (roles):

- **Administrator:** user responsible to manage the users access to the dashboard and their permissions for each resource. Additionally, the manager can add, edit and delete any resource, having no restrictions for his actions.
- **Regular User:** user with permissions predefined by the administrator and only with access to the resources predefined by those permissions.

There is a sub-level of permissions for regular users with two distinct levels: resource level and object level. The resource level refers to the different resources presented in Section 4.1. The object level refers to an object, which is a resource (i.e. Content, Timeline or View). The administrator manages these permissions through the web dashboard.

4.2.1 Resource level

The resource level has three different types of permissions: Contents, Timelines and Views. When creating an user, the administrator fills out a form with some basic information associated to the user, as illustrated in Figure 4.3. Note that this form has three checkbox fields, each one for a resource (Content, Timeline and View). These checkboxes are the permissions at resource level of the user being created and will define which resources the user can create and edit.

Only the allowed resources will be displayed in the left navigation bar of the interface of the corresponding user. That is, a user with no permissions over Contents, will only have view access to the Timelines and Views tab in the left navigation bar.

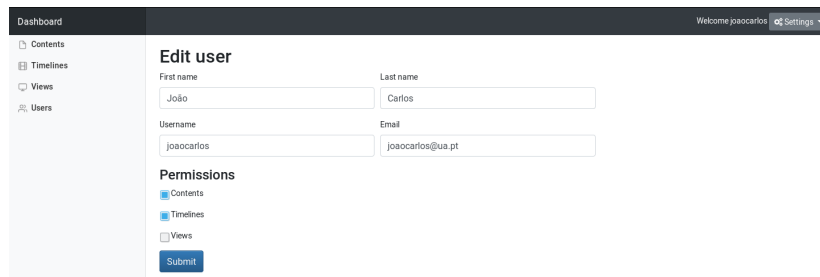


Figure 4.3: User creation web page.

4.2.2 Object level

The object level refers to which objects from the resources the user can access. By default, the user only has access to the objects he has created, although, the administrator can give access to a certain object to an user, following an ACL(Section 3.6). This list of permissions is associated to an object and has the permissions of each user to that object.

In Figure 4.7, there is a padlock for each object of the table. This padlock redirects to a page where the administrator edits the permissions over the respective object, as in Figure 4.4. This page lists all users with resource level permission over the resource of the object, so the administrator can grant access to that object.

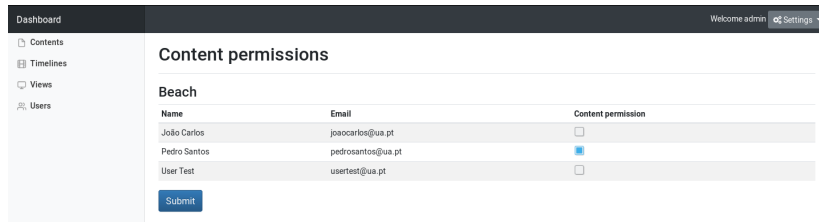


Figure 4.4: Modification of object level permissions of a Content.

This permission is particularly useful if some resource needs to be complemented by another content from a different user.

The users with Timelines and Views permissions can view all the underlying resources in order to create a Timeline or View, that is:

- To create a View, the user can use any of the underlying Timelines.
- To create a Timeline, the user can use any of the underlying Contents.

These blocking permissions allow, not only the creation of users to a specific task, but also provide the cooperation of users to a final multimedia product. As an example of specific tasks we can think of giving Content and Timeline permissions (Resource level) to a designer responsible to provide multimedia contents to the system or giving View permissions to a user responsible to the monitors inside of a specific building.

4.3 WEB SERVER

This chapter explains the main processes that the Web server executes and the workflows it uses to achieve the final product.

4.3.1 Files upload and validation

When a file is uploaded with the File Upload of Django, the web server saves the file and starts the validation of it. It starts by validating the extension of the file by checking if the extension is supported. Depending if the extension is validated or not, the Web server validates the uploaded file according to its format.

- Image format - for image validation, the `img_hdr 3.1` Python library is used.
- Video format - for video validation, the `moviepy 3.2.3` Python library is used.
- PDF format - for PDF validation, the `PyPDF2 3.1` Python library is used.
- PPT format - for PPT validation, the `catppt [44]` Linux command is used.
- PPTX format - for PPTX validation, the `python-pptx 3.1` Python library is used.

If the file is valid, the web server returns the acknowledge to the user. Otherwise, the file is deleted and an error is returned to the user.

4.3.2 PDF transformations

In order to create a video from a PDF file and due to the requirements of FFmpeg, which needs files in image or video format, the web server converts every slide of the PDF to an image file. These images are saved in a directory and all the names are saved in a text file.

FFmpeg reads the text file and creates the video with the specified duration by the user for each slide. In the end, all the images are deleted and the video file is saved.

4.3.3 Power Point transformations

The conversion process of the Power Point to video is an extension of the PDF transformation(Section 4.3.2). The Power Point file is converted to PDF with the command line LibreOffice [45] converter and then it follows the PDF transformation.

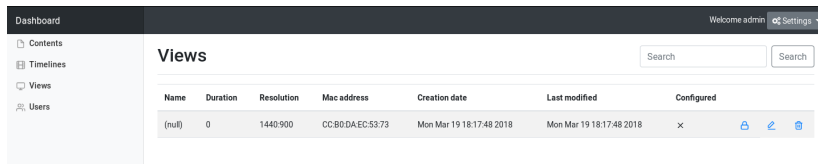
4.3.4 Timelines creation

The creation of a Timeline is straightforward and intuitive to the users. The process passes by accessing the Timeline tab and clicking the Add button. To create a Timeline, the user only needs to input a name for the Timeline and the Contents associated to it.

After the submission, the web Server starts the creation of the preview of the Timeline with a predefined 400x300px video resolution. This preview will be available in the edition of the Timeline after the process is completed.

4.3.5 Views creation

The process of creating a View is made automatically with the connection of a Raspberry Pi, being portable to another platform but optimized to Raspbian. The View will be displayed in the dashboard with an empty name and with a false configured flag, as in Figure 4.5. In other words, the View has to be configured in order to display contents and to distinguish one from the others.



The screenshot shows a web dashboard with a sidebar on the left containing navigation links for Contents, Timelines, Views, and Users. The main content area is titled 'Views' and includes a search bar. Below the search bar is a table with the following data:

Name	Duration	Resolution	Mac address	Creation date	Last modified	Configured
(null)	0	1440/900	CC:8D:DA:EC:53:73	Mon Mar 19 18:17:48 2018	Mon Mar 19 18:17:48 2018	✕

Figure 4.5: Dashboard Views web page with an unconfigured View.

Upon the configuration of a View, if it has Timelines associated, the server begins the process to create the MP4 file associated to the view. This file is compressed using the H.264 standard [40], encoded with YUV420 at 25 frames per second, as all underlying videos of the Timeline.

To better fit the resolution associated to the monitor, all the Contents are adapted to this resolution. That is, when a View is configured with Timelines associated, the system goes through all the Contents associated to these Timelines and makes the changes needed to fit the screen resolution, Content by Content.

To create, manipulate and merge these files, we chose the FFmpeg library since it has compatibility with all major video and image formats. The FFmpeg library adapts the Contents using mostly padding and resize transformations. When iterating over the frames of the Contents, the FFmpeg library resizes the frames which have different size from the resolution and applies padding to keep the Contents aspect ratio. This process will be explained in detail in Section 4.5.

4.4 DASHBOARD

As explained in Section 4.3, the system makes use of the Django web Framework to serve the website.

The dashboard was designed with the Resources and Permissions in mind in order to facilitate the interaction of the users. So, the dashboard gives control over four main resources: Contents, Timelines, Views and Users, having a navigation bar with these resources, as illustrated in Figure 4.6.



Figure 4.6: Dashboard home example.

As noticeable in Figure 4.6, the dashboard has a top navigation bar, which has a dropdown button so the user can edit his personal informations or logout, and a left navigation bar with four possible navigations. This left navigation bar only shows the possible navigations to which the user has access. Only the users with permissions over the resources can access and control the respective resource and only the administrator can edit these permissions and have access to the Users page.

When accessing a resource from the left Navigation bar, one of the pages presented in Figure 4.7 (except User) is displayed. This page allows the visualization over the existing Contents, Timelines or Views that the user has access. It also allows to create (except for Views as explained in Section 4.3), edit or delete a resource and, if the user is the administrator, to edit the permissions (object level) of the users to that resource.

The dashboard was built using three known technologies: Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript. This triad allows to create interfaces using HTML for structure, CSS for styling and JavaScript to control the flow of some elements.

When editing a resource, a form is displayed so the user can edit the data associated to the resource. In the case of Contents, the form contains two fields: the name of the Content and a file upload button.

However, when editing a Timeline or a View (Figure 4.8), a table with the Contents or Timelines associated to the resource, respectively, is displayed. This table allows the addition of objects from the dropdown button. Moreover, the objects from the table can be dragged and dropped into the intended order. When editing a Timeline, the table also has a duration input field for each image or presentation Content. By filling in this field the user can specify the intended duration for each image or slide that will be part of the Timeline. In addition,

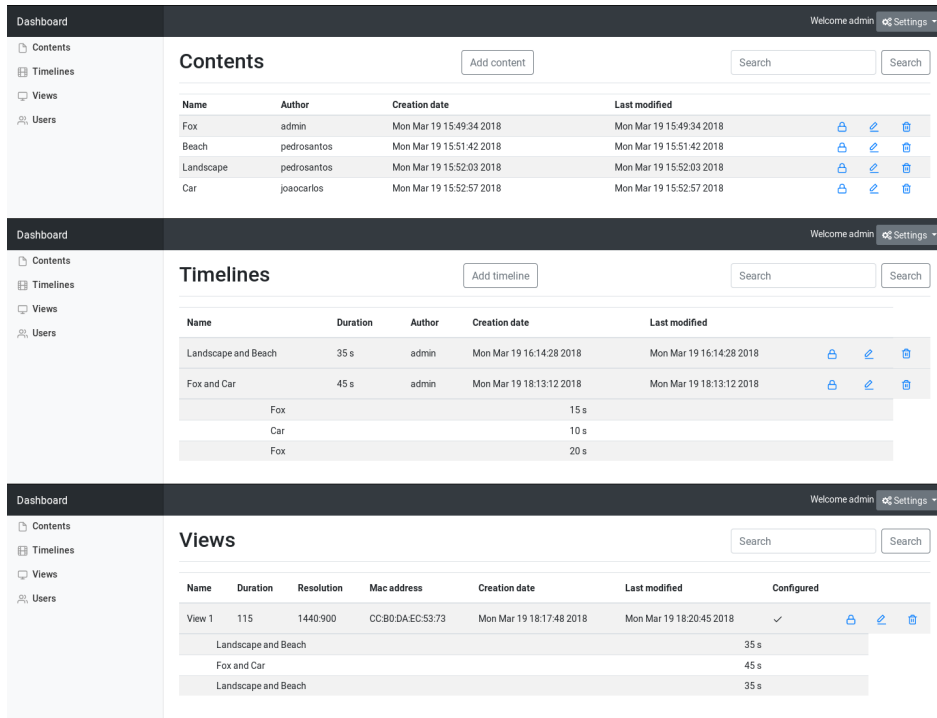


Figure 4.7: Visualization of Contents, Timelines and Views.

a preview window displaying the video reproduced from the last Timeline submission is available.

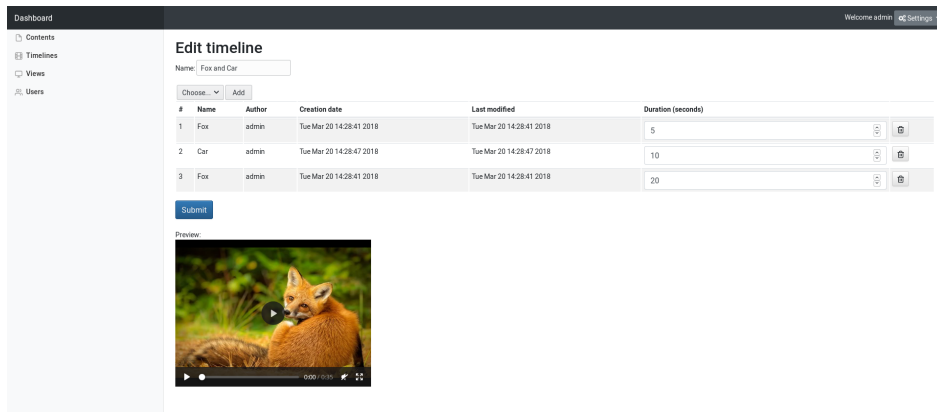


Figure 4.8: Edition of a Timeline.

4.5 MULTIMEDIA CONTENTS PROCESSING

As present in Section 3.5, the system supports three types of files: audio, video and presentation.

One of the main cores of this project is the process behind the transformation of the contents. The transformation process is mainly made by the FFmpeg library [23].

But, before explaining the FFmpeg transformations, it is important to refer 3 effects: letterboxing, pillarboxing and windowboxing [46]. These effects result mainly when some content is being displayed but the aspect ratio of the content and the monitor is different, which

usually results from a resize transformation followed by image padding (padding introduces new pixels around an image, like a border), as in Figure 4.9.



Figure 4.9: Pillarboxing and letterboxing effects [47].

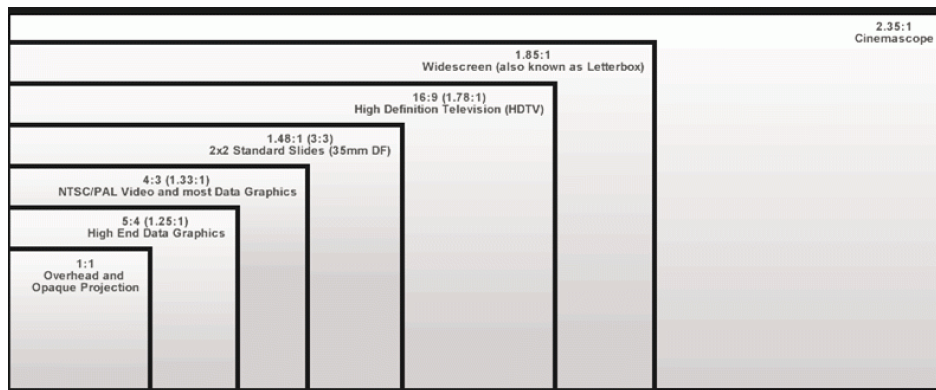


Figure 4.10: Types of aspect ratios.

These effects occur when the aspect ratios of the contents and the display is different. In Figure 4.10 the most common aspect ratios are presented.

Letterboxing consists in the transformation of frames with widescreen aspect ratio (16:9) to a NTSC/PAL video ratio (4:3) while preserving the frames original aspect ratio. This transformation consists of an image padding transformation both on top and bottom of the frames.

On the contrary, the pillarboxing effect consists in the transformation of a standard-width video format into a widescreen aspect ratio by applying image padding into the frames both on left and right.

Windowboxing consists of the combination of both effects: letterboxing and pillarboxing. This is noticeable when the frames of a video are centered in the screen with a padding effect all around them. This happens when the resolution of the screen is bigger than the frames and no resize transformation is used.

However, the windowboxing effect never occurs because the frames are always resized to a resolution, making the final frame equal the vertical or horizontal size of the resolution.

Using FFMpeg with the arguments:

- `scale and force_original_aspect_ratio`

- `pad`

makes it possible to apply the intended transformations to the frames.

The `scale` parameter allows to specify the scale resolution to apply into the frames, while using the `force_original_aspect_ratio` allows to maintain the original aspect ratio of the images. This transformation will upscale the frames, if the resolution of the screen is bigger than the frames, and downscale, in the opposite situation. The `pad` parameter allows to apply padding to the frames after the scale transformation. When the frames of the Contents have a different aspect ratio of the screen, the letterboxing and pillarboxing are perceptible.

4.6 AGENTS

The need to display the contents in the monitors and the need to have a terminal capable to communicate with the Web Server lead us to choose a Raspberry Pi [13] to integrate the distributed system.

This equipment has enough processing capabilities to display the contents with a low need for space area and low energy dependency. It also has a 802.11b/g/n/ac networking interface to communicate over Wi-Fi in order to download the contents from the Web Server.

To test the system, we used the Raspberry Pi Model 3B+, which needs a proper power plug to connect the power supply, otherwise the Raspberry Pi will shut down on boot. We also tested the system using a Raspberry Pi Model 3B and, on the contrary of Model 3B+, it does not need a proper power supply, that is, it only needs a Micro-USB connector as power supply (present in almost every LCD or monitor).

4.6.1 Life cycle

Concerning the development of the Raspberry Pi system, the communication with the Web Server is made over HTTP requests and, as most of the case studies were in the Eduroam network (which does not allow peer-to-peer communication), all the communication requests must come from the Raspberry Pi to the Web Server and not vice-versa.

This lead us to create a life cycle of the system, using a Python script to control the display and the communication. When the Raspberry Pi starts, the script runs on boot and goes through a series of steps in order to register in the server (if it is the first time connecting) and download the video.

This life cycle, as illustrated in Figure 4.11, is divided in four major states.

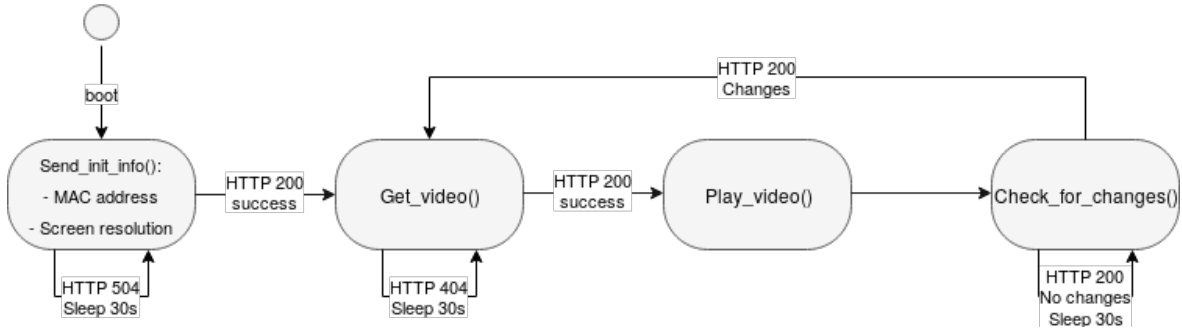


Figure 4.11: Monitor life cycle.

State 1 - send_init_info()

In this state, the script tries to fetch some data to send it to the Web Server, namely:

- MAC Address - using the netifaces [34] Python library to fetch it, this information allows the server to distinguish uniquely the different terminals and allows the user to know which View is associated to a terminal.
- Screen resolution - using the screeninfo [33] Python library to fetch it, it is used to adapt the Contents associated with some View to its screen resolution.

To fetch the MAC address, the script tries to fetch the address associated to the Ethernet interface and if it fails, tries to fetch the address associated to the 'wlan0' interface, which is the default wireless interface. If both cases fail, it tries to fetch the address associated to another wireless interface present in the list of interfaces.

To fetch the screen resolution, screeninfo [33] fetches the size and location of every physical screen connected. The script chooses the screen resolution of the default monitor. If it fails, the script waits for a period of time and tries to fetch it again, that is, when a monitor is duly connected.

After this, it sends the information to the Web Server with a HTTP Post Request. The server, after receiving this information, checks if the MAC address already exists and returns the path in the server to the video corresponding to that View.

State 2 - get_video()

The second state after fetching the path to the corresponding video, is responsible to download the video from the Web Server with a HTTP GET request and store it in the file system of the Raspberry Pi. If it fails (if the View wasn't configured yet or the server is down), the terminal waits thirty seconds by default, and tries to download it again. This loop is repeated until the video successfully downloads.

State 3 - play_video()

The third state is only responsible to launch the process with the video player in order to play the video in the monitor.

To play the video, the Omxplayer is used, which is a command line OMX player specifically made for the Raspberry Pi's GPU [48]. The player is launched with the following command line flags:

- `-g` - used to generate the log file.
- `-b` - used to set background to color black.
- `-no-osd` - used to hide some status information on the screen.
- `-loop` - used to repeat the video indefinitely.

State 4 - `check_for_changes()`

The fourth and final state is repeated infinitely and is responsible to poll the web Server, sixty seconds by default, to check for any changes in the View. If any changes are detected, the script jumps back to State 2 to download the video and continue the cycle, as in Figure 4.11.

If by any case the Internet connection is broken, it continues to poll the server until a response is obtained. While this happens, the previously downloaded video continues to be played, which means that the video is always displayed, even with no Internet connection, after a successful download.

In Figure 4.12, the terminal output of a Raspberry Pi, since the connection with the server until the video is successfully downloaded, is presented. As this was just a test, the execution of the program stops when the video is downloaded. In a normal situation, the execution continues with State 4.

```

File Edit View Search Terminal Help
pi@rpi:~/monitor$ ls
requirements.txt run.py startup.sh
pi@rpi:~/monitor$ python3 run.py
14:58:03.876 - Monitor Logger - INFO - Initializing monitor
14:58:03.880 - Monitor Logger - INFO - Sending initial info to server
14:58:05.298 - Monitor Logger - INFO - Video unavailable. Retrying in 30s...
14:58:37.294 - Monitor Logger - INFO - Video unavailable. Retrying in 30s...
14:59:13.344 - Monitor Logger - INFO - Video download complete
14:59:13.345 - Monitor Logger - INFO - Running
pi@rpi:~/monitor$ ls
Monitor.log requirements.txt run.py startup.sh view.mp4
pi@rpi:~/monitor$

```

Figure 4.12: Raspberry Pi terminal output. It is visible, from the output, that the video was successfully downloaded.

4.7 WEB SERVER AND RASPBERRY PI COMMUNICATION

The communication between the web server and the Raspberry Pi is made with HTTP GET and HTTP POST Requests from the Raspberry Pi to the web server.

Since the system was designed to be deployed in the Eduroam network and as it does not allow peer-to-peer communication, all the requests must come from the agents to the server.

The Web Server exposes three specific Uniform Resource Locator (URL) for the terminals to make the requests:

- `login/` - used to authenticate.
- `monitor/new_monitor/` - used to register in the system or/and fetch the URL from the corresponding video.
- `monitor/check_for_changes/` - used to check for changes in the View.

Every request to the Web Server requires the terminal to log into the system in order to get the Cross-site request forgery (CSRF) token, provided by Django, to protect against Cross Site Request Forgeries attacks.

A CSRF hole is when a malicious site can cause a visitor’s browser to make a request to the server that causes a change on the server. The server thinks that because the request comes with the user’s cookies and that the user wanted to submit that form [49].

According to the Django documentation: This type of attack occurs when a malicious website contains a link, a form button or some JavaScript that is intended to perform some action on your website, using the credentials of a logged-in user who visits the malicious site in their browser [50].

The CSRF middleware and template tag of Django provides automatic mechanisms to protect against this type of attacks.

4.8 DATABASE IMPLEMENTATION

Django provides an easy-to-use database-access API. The API, by creating Django Models [51] like a Python class, maps each model to a single database table. This process abstracts the creation of the database tables from the developer and is the same for any database engine.

Regarding the application of Django Models in our project, we tried to make use of the authentication system that Django provides, making an extension of the existing User model by adding three boolean fields: one for each of the resources of Section 4.1, representing the permissions at resource level of Section 4.2.1. These extended models are the users that have access to the dashboard.

Also, we created three models for each resource and two more to link the resources related to each other, as illustrated in Figure 4.13.

The diagram, created using the Django Extensions collection [52], illustrates the tables existent in the database and their relations. There are three groups of relations to highlight:

- Firstly, the relations between the User table and the Content and Timeline tables. These relations define the creators of the objects and are “one to many” relations because a user may have many objects but an object only has one creator.
- Secondly, the relations between the UserProfile table and the Content, View and Timeline tables. These relations are used to define the ACL permissions over the objects. They are “many to many” relations because a user may have permission over a variety of objects and an object may have different users with access to itself.
- Lastly, the relations between the Content, View and Timeline and TimelineContents and ViewTimelines. These relations define the Contents of a Timeline and the Timelines of a View. As the Timelines may have many Contents and the Views may have many Timelines, the relation is “many to many”.

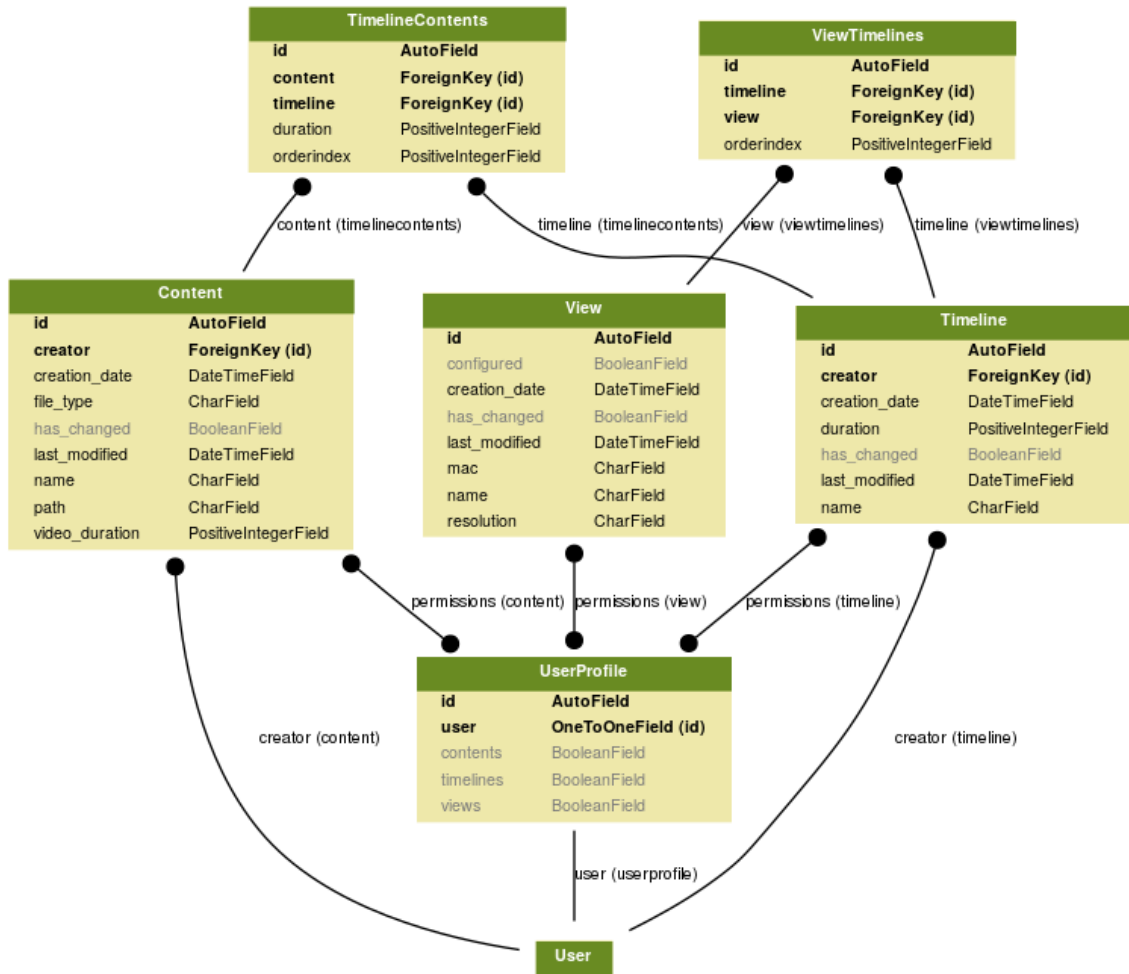


Figure 4.13: Database object-relational mapping diagram.

Case studies

This chapter presents the main results obtained by using the the system we proposed. To test the system, we used a virtual machine provided by Serviços de Tecnologias de Informação e Comunicação (sTIC) of the University of Aveiro as the Web Server. This makes the server available to all the terminals inside the Eduroam network, making the system portable and usable in any department and the access to the dashboard possible in any local with Internet connection (using Virtual private network (VPN)). The server has the Hypertext Transfer Protocol (HTTP) Port (number 80) open in order to enable requests from other hosts inside the Eduroam network.

5.1 CASE STUDIES

The system was tested in two environments: IEETA and Students@DETI [53]. Both environments require the dissemination of multimedia contents. The following sections explain the procedure and details about each one.

5.1.1 IEETA monitors

Since the application of monitors across IEETA, the display of multimedia contents used a USB flash drive to store the contents and connect it to the monitor. The process of updating the contents passed by removing the USB flash drive and manually update the contents using for example a computer.

Then, the need to automatize and speed up this process contributed to the emergence of this thesis.

In Figure 5.1, we can see a real Full HD monitor (1920x1080 px of resolution and a 1.78:1 aspect ratio) with a Raspberry PI single board connected to the developed system displaying the final output: the video stream for that View.

The building has three monitors spread across the area. As use case, we used the three monitors connected with a Raspberry Pi and to test the monitors we used the following Contents.



Figure 5.1: Final result of a monitor in IEETA.

- 7 PDFs with 15 seconds of display.
- 3 Power Point's with 5 seconds for each slide.
- 2 videos with MP4 format.

All of these Contents were composed in one Timeline, resulting in a video with 1 hour and 51 minutes. The system was updating automatically the video if any changes were made only with a delay (the time of creation of the video on the server and the time of polling and download).

5.1.2 Students@DETI

Every year, there is an exhibition hosted in the Electronics, Telematics and Informatics Department of University of Aveiro with projects from teachers and students. This year, we deployed our system in three spots of the event in order to test and advertise it.

The system displayed multimedia contents uploaded by the organizer of the event with contents relevant to the event, mainly PDF's, Power Points and videos.

Two of the three spots were using a projector to display the multimedia contents into a wall, one in the entrance of the event and the other one in a hall. These two spots were displaying three PDF Contents with 15 seconds for each slide converted to a Timeline.

The other spot was a smaller monitor in a table where the project was in exhibition. The monitor was displaying some posters from the different projects present in the event. As the posters had a vertical orientation, we changed the orientation of the Raspberry Pi and the server made the changes automatically to better fit the monitor. Figure 5.2 monitor was displaying a Timeline with 15 PDF posters with 10 seconds for each one.

We only noticed one problem due to the weak signal of the wireless network, making the agent unable to communicate with the server, but with some adjusts the signal was reached and everything ran as expected.

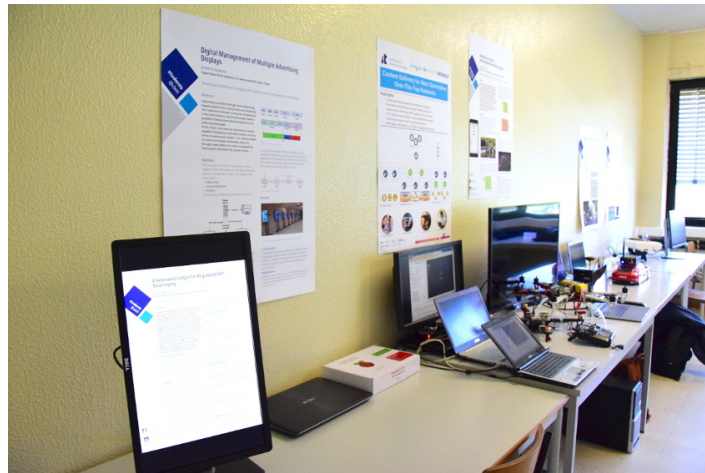


Figure 5.2: Demonstration and exhibition of the system at Students@DETI, using a vertical monitor.

Conclusion

The main goal of this thesis was to propose a solution to manage multiple multimedia contents in order to display them in multiple monitors. The system we have presented, as a whole, is operational and ready to manage the upload of multimedia contents and display them. It implies the existence of a computer to host the server (with proper image and video processing capability), monitors to display the contents, single boards to connect to the monitor (like a Raspberry Pi) and, of course, network connection between the server and the terminals.

Overviewing the system and the results in a whole, we can assume the objectives mentioned in the Introduction were generally accomplished. However, there are many aspects to improve and features to add.

The case studies gave us feedback to guide the system to meet the needs and requirements in the practice. Several other events have been organized at the University of Aveiro and one of the requirements was the user interaction with the displays, allowing to skip or pause the presentations.

Nevertheless, the events where the system operated were a success and the further work on this project can improve these features and give a more complete solution to meet these kind of requirements.

6.1 FUTURE WORK

A global overview over the results highlights some features that we intend to improve, as future work. Such features are:

- The aspect of the dashboard that needs to be more appealing and intuitive to the user. It could display for example, snapshots of the Contents and an easier way to upload them (like drag and drop upload).
- A more intuitive way to organize the Contents, Timelines and Views. As the system grows and the number of users and events too, a way to organize these resources is needed, for example by creating a tag for each resource. This way, we could filter the resources by event, building, etc.

- A more advanced Timeline editor in order to give the user a greater control over the sequence of Contents.
- The capability to fetch information in real time about the monitor in order to show the status in the dashboard; Moreover, making it possible to run certain commands over the monitor, like turn on and turnoff would be a desired improvement of this work.
- Support the change of the orientation of the display, for example to use a monitor with a vertical orientation without changing the orientation manually in the agent.

In the future, as the system improves, we intend to expand the supported formats of Contents, like supporting audio files. This enhancement of the system would eventually imply a reformulation of Timelines creation in the dashboard.

Another interesting feature that we plan to develop is the support for the interaction between the target audience and the system, either by a physical contact or even by voice or gesture control. This interaction would allow the user to pause and skip contents, for example.

References

- [1] A. Di Rienzo, F. Garzotto, P. Cremonesi, C. Frà, and M. Valla, «Towards a smart retail environment», in *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, ACM, 2015, pp. 779–782.
- [2] N. I. Bruce, B. Murthi, and R. C. Rao, «A dynamic model for digital advertising: The effects of creative format, message content, and targeting on engagement», *Journal of Marketing Research*, vol. 54, no. 2, pp. 202–218, 2017.
- [3] A. L. Roggeveen, J. Nordfält, and D. Grewal, «Do digital displays enhance sales? role of retail format and message content», *Journal of Retailing*, vol. 92, no. 1, pp. 122–131, 2016.
- [4] J. Schaeffler, *Digital signage: Software, networks, advertising, and displays: A primer for understanding the business*. Taylor & Francis, 2012, ISBN: 9781136031533. [Online]. Available: <https://books.google.pt/books?id=9ZUrt7-igxQC>.
- [5] *Yodeck official website*. <https://www.yodeck.com/>, Accessed: 2018-05-30.
- [6] *Yodeck digital signage solution*. <https://www.yodeck.com/wp-content/uploads/2015/12/yodeck-04-2.png>, Accessed: 2018-06-08.
- [7] *Xarevision*, <https://www.xarevision.pt/>, Accessed: 2018-07-30.
- [8] *JCDecaux official website*. <http://www.jcdecaux.com/>, Accessed: 2018-05-30.
- [9] *JCDecaux billboard*. http://www.jcdecaux.com/sites/default/files/styles/left/jcd_asset_small/4885/public/assets/image/2016/10/jcdecaux-stockholm.jpg, Accessed: 2018-05-30.
- [10] *Enplug*, <https://www.enplug.com/>, Accessed: 2018-06-30.
- [11] *NoviSign*, <https://www.novisign.com/>, Accessed: 2018-06-30.
- [12] *Mvix*, <http://www.mvixusa.com/systems/>, Accessed: 2018-06-30.
- [13] *Raspberry PI official website*. <https://www.raspberrypi.org/>, Accessed: 2018-05-22.
- [14] *ScreenCloud*, <https://screen.cloud/>, Accessed: 2018-06-30.
- [15] *Flask official website*. <http://flask.pocoo.org/>, Accessed: 2018-05-22.
- [16] *Django official website*. <https://djangoproject.com>, Accessed: 2018-05-22.
- [17] *Advantages and disadvantages of Django*. <https://hackernoon.com/advantages-and-disadvantages-of-django-499b1e20a2c5>, Accessed: 2018-06-05.
- [18] *Flask and Django popularity in StackOverflow questions*. https://s3.amazonaws.com/codementor_content/2017-Feb/Flask+vs+Django+Popularity, Accessed: 2018-06-05.
- [19] *OpenCV official website*. <https://opencv.org/>, Accessed: 2018-05-22.
- [20] *Python Imaging Library official website*. <http://www.pythonware.com/products/pil/>, Accessed: 2018-05-22.
- [21] *Pillow*, <https://github.com/python-pillow/Pillow>, Accessed: 2018-05-22.

- [22] *MoviePy*, <https://zulko.github.io/moviepy/>, Accessed: 2018-05-22.
- [23] *FFmpeg official website*. <https://www.ffmpeg.org/>, Accessed: 2018-05-29.
- [24] *SQLite official website*. <https://www.sqlite.org/index.html>, Accessed: 2018-05-28.
- [25] *MySQL official website*. <https://www.mysql.com/>, Accessed: 2018-05-28.
- [26] «Information technology – Multimedia framework (MPEG-21) – Part 3: Digital Item Identification; Information technology – Multimedia framework (MPEG-21) – Part 1: Vision, Technologies and Strategy; Information technology – Multimedia framework (MPEG-21) – Part 2: Digital Item Declaration», International Organization for Standardization, Standard, 2003, 2004 and 2005.
- [27] *Python imghdr*, <https://docs.python.org/3.4/librar/imghdr.html>, Accessed: 2018-05-28.
- [28] *Python PyPDF2*, <http://mstamy2.github.io/PyPDF2/>, Accessed: 2018-05-28.
- [29] *Python pdf2image*, <https://github.com/Belval/pdf2image>, Accessed: 2018-05-29.
- [30] *Python python-pptx*, <https://github.com/scanny/python-pptx>, Accessed: 2018-05-29.
- [31] *Python ffmpeg*, <https://github.com/Ch00k/ffmpeg>, Accessed: 2018-05-29.
- [32] *Python subprocess*, <https://docs.python.org/3/library/subprocess.html>, Accessed: 2018-05-29.
- [33] *Python screeninfo*, <https://github.com/rr-/screeninfo>, Accessed: 2018-05-29.
- [34] *Python netifaces*, <https://github.com/al45tair/netifaces>, Accessed: 2018-05-29.
- [35] *Python requests*, <https://github.com/requests/requests>, Accessed: 2018-05-29.
- [36] G. K. Wallace, «The jpeg still picture compression standard», *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.
- [37] C. Wilbur, «Png: The definitive guide», *Journal of Computing in Higher Education*, vol. 12, no. 2, pp. 94–97, 2001.
- [38] *Audio Video Interleave (AVI)*, [https://msdn.microsoft.com/en-us/library/windows/desktop/dd318187\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd318187(v=vs.85).aspx), Accessed: 2018-05-28.
- [39] *MPEG-4*, <https://mpeg.chiariglione.org/standards/mpeg-4>, Accessed: 2018-05-28.
- [40] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, «Overview of the h. 264/avc video coding standard», *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.
- [41] V. Sze, M. Budagavi, and G. J. Sullivan, «High efficiency video coding (hevc)», *Integrated Circuit and Systems, Algorithms and Architectures. Springer*, vol. 39, p. 40, 2014.
- [42] R. S. Sandhu, *Role-based Access Control*. 1998, vol. 46, pp. 237–286, ISBN: 0120121468.
- [43] Baptista, Arménio and Trifan, Alina and Neves, António, «Digital Management of Multiple Advertising Displays», *International Conference on Autonomic and Autonomous Systems*, 2018.
- [44] *catppt command*, <https://linux.die.net/man/1/catppt>, Accessed: 2018-07-01.
- [45] *LibreOffice*, <https://www.libreoffice.org/>, Accessed: 2018-07-01.
- [46] C. Poynton, *Digital video and hd: Algorithms and interfaces*. Elsevier, 2012.
- [47] *Pillarboxing and letterboxing effects*. <https://cdn2.desu-usergeneratedcontent.xyz/g/image/1517/46/1517465156572.jpg>, Accessed: 2018-06-06.
- [48] *Omxplayer Github page*. <https://github.com/popcornmix/omxplayer>, Accessed: 2018-05-31.
- [49] *Cross-site request forgery*, <https://www.squarefree.com/securitytips/web-developers.html>, Accessed: 2018-07-02.

- [50] *Django's Cross Site Request Forgery protection*. <https://docs.djangoproject.com/en/2.0/ref/csrf/>, Accessed: 2018-05-31.
- [51] *Django Models documentation*. <https://docs.djangoproject.com/en/2.0/topics/db/models/>, Accessed: 2018-06-05.
- [52] *Django Extensions collection*. <https://github.com/django-extensions/django-extensions>, Accessed: 2018-07-02.
- [53] *Students@DETI website*. <http://studentsandteachersdeti.web.ua.pt/>, Accessed: 2018-07-09.