FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Dashcam Video Streaming and Storage

José Costa

DISSERTATION

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Prof[a] Maria Teresa Andrade

July 21, 2019

Dashcam Video Streaming and Storage

José Costa

Mestrado Integrado em Engenharia Informática e Computação

July 21, 2019

Abstract

With the incessant growth of the Internet-of-Things paradigm where all sorts of devices, from the, now trivial, smartphone to smart thermostats and smart fridges, are connected and interact with each other, exchanging data, it's imperative to keep up with the current landscape. A particular device of growing interest and success is the dashboard camera or dashcam. A dashcam is a video recording device usually placed on the dashboard of vehicles, pointing forward, with the purpose of recording the journey to use the footage as proof for insurance companies in case of any incidents, among other motives. Purpose-built dashcams have a set of features that distinguishes them from the typical camera, hardware, and software-wise. Despite the, seemingly limitless, options in the market, the use of cloud storage and cloud connectivity, in general, can be a bit of a lackluster experience, with only a handful of devices even supporting those features. This is where this dissertation comes in. Proposed by WIT-Software, its main objective consists of the implementation of a proof of concept of a platform capable of interacting with different camera models and use the cloud features to allow users remote access, via mobile app, among other useful features. It integrates the Cloud features and the backend side with the already existing V-Auto app, developed by WIT-Software, that tracks the trips a user makes by using a device placed on their vehicle, and then presents that data in an appropriate way. The research surrounding the dashcam market and all the technologies relevant to the desired system helped to narrow the score of concepts and define what the platform will be in a more specific way. Simultaneously to the development of this system, there was a documentation of the process, along with the analysis, and consequent recommendation, of the best practices when it comes to cloud interaction in regards to video handling, storing, streaming, etc. The end result consists in a demonstrable, centralized and integrated system that uses the cloud features and interacts with the recording device and the mobile app.

Resumo

Com o crescimento incessante do paradigma Internet-of-Things onde todo o tipo de dispositivos, desde dos, agora triviais, smartphones até aos termostatos inteligentes e frigoríficos inteligentes, estão conectados e interagem uns com os outros, trocando dados, é imperativo manter-se a par do panorama atual. Um dispositivo particular de interesse e sucesso crescentes é a dashboard camera ou dashcam. Um dashcam é um dispositivo de gravação de vídeo normalmente colocado no tablier de um veículo, apontado para a frente, com o propósito de filmar a viagem para uso das filmagens como provas para companhias de seguros no caso de algum incidente, entre outros motivos. Daschams feitas para esse efeito têm uma série de funcionalidades que as distingue da câmara típica, em termos de hardware e software. Apesar das opções, aparentemente sem limite, no mercado, o uso de armazenamento na cloud e conectividade com a cloud no geral pode ser uma experiência pouco conseguida, com apenas um número escasso de dispositivos com suporte para estas funcionalidades. É aqui que esta dissertação entra. Proposta pela WIt-Software, o seu objetivo principal consiste na implementação de uma prova de conceito de uma plataforma capaz de interagir com modelos diferentes de camaras e usar as funcionalidades da cloud para permitir aos utilizadores acesso remoto, via aplicação mobile, entre outras funcionalidades úteis. Integra as funcionalidades Cloud e a lado backend com a app existente V-Auto, desenvolvida pela WIT-Software, que monitoriza as viagens de um utilizador usando um dispositivo no veículo e a apresenta da forma apropriada. A pesquisa à volta do mercado das dashcams e de todas as tecnologias relevantes para o sistema desejado ajudar a restringir a extensão dos conceitos e a definir o que a plataforma será duma maneira mais específica. Simultaneamente ao desenvolvimento deste sistema, houve uma documentação do processo, junto com a análise, e consequente recomendação, das melhores práticas no que toca a interação com a cloud no que toca a manuseamento de vídeo, armazenamento, streaming, etc. O resultado final consiste num sistema centralizado, integrado e demonstrável que usa funcionalidades cloud e interage com o dispositivo de gravação e com a aplicação mobile.

Acknowledgements

Despite the personal effort that was necessary throughout the entire process, it wouldn't have been possible without the help of several people.

First and foremost my supervisor, Prof. Maria Teresa Andrade, and everyone at WIT-Software, more specifically my coordinators, João Alves and Pedro Marques, and Miguel Goyanes.

And not forget my friends and family that kept me going through these few months with highs and lows but always focused and moving forward.

José Costa

"I'd been going to college for nine years,
and before I completed my dissertation, I quit."

Robert Weinberg

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

IoT	Internet-of-Things
RFID	Radio-Frequency Identification
CDN	Content Delivery Network
WAN	Wide Area Network
OBD	On-Board Diagnostic
API	Application Programming Interface
JPEG	Joint Photografic Experts Group
MPEG	Moving Pictures Experts Group
DCT	Discrete Cosine Transform
AVC	Advanced Video Coding
HEVC	High Efficiency Video Coding
GPS	Global Positioning System
SD	Secure Digitial
Soft AP	Software Access Point
WPA	Wi-Fi Protected Access
HAS	HTTP Adaptive Streaming
ISP	Internet Service Provider
MPD	Media Presentation Description
DB	Database
PoC	Proof of Concept
AWS	Amazon Web Service
RTP	Real-Time Transport Protocol
RTMP	Real-Time Messaging Protocol
HLS	HTTP Live Streaming
DASH	Dynamic Adaptive Streaming over HTTP
DBMS	Database Management System
MVP	Model-View-Presenter
MVC	Model-View-Controller
JVM	Java Virtual Machine

Chapter 1

Introduction

1.1 Overview

This chapter provides context for the dissertation work, as well as an explanation of the motivation behind it and the goals achieved in its duration. It introduces the concept of IoT and IoT devices. It goes a bit more in-depth into a particular type of device, Dashcams, and the way they work and their presence in the modern world. It also provides a brief description of the hosting company, WIT-Software.

A dissertation report structure is also given to clarify the other chapters included in the present document.

1.2 Context

Lately, there has been a resurgence in the tech market and in the scenario of modern wireless telecommunications when it comes to a relatively new concept, the Internet-of-Things paradigm. The core idea of this concept is the constant presence, around us, of a variety of objects – such as RFID tags, sensors, actuators, mobile phones, etc. – which are able to interact with each other and cooperate among themselves for the same end goal [AIM10].

In this context, one particular device of growing success, among the innumerable types, from smartphones to smart lightbulbs, fridges and other home appliances, is the Dashboard Camera (or Dashcam). Dashcams are cameras equipped in the front and sometimes rear, of a vehicle, with the ability to record video continuously. Some cameras can feature multiple recording modes, as well as a built-in GPS that displays vehicle speed and location. Virtually any portable camera or recording device could be used as a dashcam, but purpose-built devices typically run on a voltage that allows them to be hard-wired into a vehicle's electrical system, the device automatically starts recording whenever the vehicle is being driven, and old data is overwritten as new data is recorded, thus making a more efficient use of the, most commonly used, SD memory card. Other features include G-sensors that are triggered in case of impact, notifying the user or the camera

to record a separate, event tagged, video that will not be deleted, and the ability to record while parked.[Bla19][Lau18]

Dashcams are generally used to provide evidence for insurance companies in the case of an accident or similar event and all-around protection and safety. Legality can be a complicated subject though, with several EU countries, like Austria or Luxembourg, not allowing its use, and others, like Spain or Serbia, allowing it. In many cases, video from dashcams can also be used in court.[Smi18]

The company where the development process took place is WIT-Software. WIT-Software develops software for telecommunications operators of several continents such as the Vodafone Group (Europe), Deutsche Telekom (Germany), Reliance Jio (India), KDDI, Softbank, NTT Docomo (Japan), Singtel (Singapore), Telstra (Australia), Unitel (Angola), Eir (Ireland), Telecom Italia (Italy), Orange (France), Telefónica (Spain), TeliaSonera (Sweden), Belgacom (Belgium), Post Luxembourg (Luxembourg), Bell (Canada), Century Link (USA) and Everything Everywhere (RU)[WS19]. Its software is present in 40 countries and it has offices in Portugal and the UK with its developing centers located in Coimbra, Porto, Leiria, and Aveiro. The many software solutions WIT develops include IoT solutions for motor vehicles that allow the monitoring and aggregation of data about journeys, in which it would be useful to explore the viability of adding video information as well.

1.3 Motivation and Goals

With this reality in mind, WIT-Software proposed this dissertation intending to develop a proof of concept of a centralized dashcam integrated system, which would be agnostic to the type of camera and that could be used by a telecommunications operator as a central video management and streaming service, independently from camera model/brand.

To achieve this, it was necessary to study how connectivity works in the most used cameras in the market, as well as use cases and features offered by these cameras and their interfaces, those being mobile apps, a web interface or an API integrated with other services. During the dissertation, a prototype that interacts with footage representative of the typical dashcam was implemented. This prototype contains a backend system that receives and manages the footage and a frontend application, developed in android, and integrated with an already existing app.

It was also important to document the process accordingly and offer insight on cloud video handling protocols, giving recommendations and analyzing options to understand what standard is more appropriate for each different task.

Thus, the end goal being a dissertation that explains what are the most predominant techniques and protocols in the market and offers a recommendation of the best implementation practices and a final solution of a system capable of supporting several types of devices and with video recording and 4G streaming capabilities.

1.4 Dissertation Structure

This Dissertation follows a structure that aims to provide a basic understanding of the concepts discussed and the current state of the art and current work related to the theme. It also explains the issues that the developed platform aims to address and the requirements to be fulfilled. It then traces the steps of the implementation process and lastly, presents present conclusions of the work done.

In chapter 2 we look into the current state of the dashcam market, the way these cameras operate, their features and their position in the modern world. On the more technical side of things, it provides a basic understanding of relevant concepts such as video compression, storage (more specifically cloud storage), video streaming, CDN's and connectivity aspects, namely Wi-Fi Direct and WAN. The work done so far is also described as the several themes are explained.

In chapter 3 we take a look at the specific problem at hand and the solution to address it. As well as the functional and non-functional requirements of the platform, some use cases generated from those requirements, and the work plan to be conducted along with an explanation of what each step is more specifically and how it will be implemented.

The fourth chapter (4) looks at how the previously described solution was implemented step by step, and by giving special attention to the decisions made in the process.

To wrap everything up, in chapter 5 we conclude the dissertation by looking at in what measure the expected results were achieved and a feature roadmap for the future of the platform.

1.5 Summary

IoT devices are becoming a reality more and more every day, and by facilitating the access to its innumerable capabilities we are only going with the flow and keeping up with the outside world.

What this dissertation aims to achieve is a description of how a system that offers features in line with the market, as well as some innovative ones, can be implemented and interact with devices and tools the general public already knows and uses.

Chapter 2

State of the Art

2.1 Overview

This chapter describes the current state of the Dash camera market, going through how dashcams work in general and analyzing and comparing options and features available to paint a picture of the current landscape and to perceive the requirements to be fulfilled. Afterward, it looks into relevant technologies, defining and explaining them along with making a connection between each one and the theme at hand. More specifically, video compression, flash memory, adaptive streaming protocols, cloud upload protocols, cloud storage, CDNs, video metadata, Wi-Fi Direct, and WAN connectivity. One of the sections, regarding OpenStack, was only added during the development phase itself, when it was decided that it would be the cloud environment to use, as a suggestion from the host company. Lastly, it goes through some work, already made, related to the subject of dash cameras and handling of its video.

2.2 Dashcam Solutions

2.2.1 Analysis

The research process started by exploring dashcams themselves. What they are, how they work, and what features do they have. As previously mentioned, Dash cameras are specialized recording devices placed on a vehicle's dashboard, that record footage from the front of the vehicle, and sometimes the inside of the cabin or the rear of the vehicle. It doesn't necessarily have to be a dashcam in order to perform the basic task of recording the road or the cabin, any video recording device can do it, however only dashcams themselves have a set of extremely useful features, and some would consider vital, in the use and operation of these devices.

As soon as the vehicle is started, the camera starts recording and keeps recording until the power is cut. Power which can be fed to the device in multiple forms, it can either be through a connection to the cigarette lighter adapter (which is the default for most manufacturers), by connecting to the OBD-II port under the steering wheel (the OBD refers to a self-diagnostics system that has been mandatory for all vehicles in the USA and Europe since 1996) or hardwiring

directly to the cars or using an external power source. It's the ability to run on 12V DC that allows dashcams to be hardwired to the battery and start recording as soon as the car starts. Most cameras also have small internal batteries that prevent the loss of data when there are sudden power cuts.

Another important feature is the way dashcams record video. In almost all cases, dashcams use either a micro SD card or some form of local storage to house data. And at first glance that may not seem like the best idea, especially when considering how big some video files can be and the growing image quality in dash cameras. However video isn't saved in "trips" that could sometimes last hours, they are saved in clips of, usually 2 or 3 minutes, but that can go up to 10 (according to user preference). When the storage is full, instead of not recording new footage, the oldest clips are deleted (as long as they aren't specifically marked to be saved) and replaced with new footage, thus making the somewhat limited space of an SD card virtually "limitless".

Something present in almost all dash cameras is a G-Sensor. A G-Sensor is a sensor that can perceive the change of accelerating force which is the force that causes accelerated motion. This is useful because in the case of an impact or an event that causes some sort of hustle that particular 2 or 3 minute clip is marked and prevented from being overwritten when the situation described in the previous paragraph happens. It can also make that clip easier to find when the user is looking for it. Most cameras also have a physical button on them that has the same effect (marking a clip so it doesn't get overwritten) in case the event isn't something that would trigger the G-Sensor.

Finally, there is one feature that isn't on all dash cameras, but on a considerable number nonetheless, and that is parking mode. Parking mode, as the name suggests, is the ability to record when the vehicle is stationary and turned off. This feature only works if the device is connected to an external power source, hardwired directly to the battery, or connected to the OBD-II port (in the last two options some precautions must be exercised in order not to drain the battery). There are several types of parking modes, the simple mode consists of the camera not recording until the G-Sensor is activated, or a motion sensor in some cases. After an impact is detected the camera turns on and starts recording. A buffered parking mode records continuously and when an impact is detected the clip is marked so it doesn't get overwritten.

These four basic features are what distinguish specially designed dashcams from any other camera in the market and are making them more present each day.

2.2.2 Devices & Features

One of the first steps of the research process was understanding the landscape of the dashcam market. In order to paint this picture, many different devices were analyzed from many different brands and taking into account user satisfaction and overall quality and value. It was also important to select a single, representative, device that could be used for development and testing in the future.

The first restricting factor was Wi-Fi. Only Wi-Fi enabled cameras were selected into the final group of devices taking into consideration that there would no possible connection between the camera itself and the system to be developed without those capabilities. Remote access was also a desired feature so cloud capabilities were also a deciding factor.

After thorough research and analysis these were the dashcams selected:

- **Garmin Dash Cam 55**: The Garmin Dash Cam 55 is the mid-range model of Garmin dash cameras. It has a 2.0" screen that can be used to view the current feed and control the camera. It has GPS and voice control as well as parking mode. It also has driving assistance features, more specifically forward-collision warning when the vehicle gets to close to the one in front, and lane departure warning [gara].

- **Rexing V1 gen 3**: The Rexing V1 gen 3 also has a screen to aid in camera operation. Its video resolution can vary according to the desired frame rate, it provides a high video quality with a wide field of view, however, it doesn't have any type of parking mode, and GPS can only be obtained through a separately sold accessory [rex].

- **Roav Dashcam C1 Pro**: The C1 Pro from Roav is a more affordable option, however, it stills offers great value along with great features. It comes with a built-in GPS and has a parking mode feature. It also has a 2.4" screen [roab].

- **YI Smart Dash Camera**: The YI Smart Dash Camera is the most affordable option by some distance, and before cloud capabilities were such an important feature to have in the platform to develop, was the selected, representative, camera. It has a screen to view the live feed but no GPS or parking mode. It also has some driving assistance features such as lane departure warning and distance detection of a vehicle in front (similarly to the Garmin Dash Cam 55) [yic].

- **Owl Car Cam**: The Owl Car Cam is the first from this list with cloud capabilities. It's also the only one that doesn't use a Micro SD Card for storage, coming with 64 GB of internal memory. It has dual cameras, one pointing towards the road and another filming the inside of the vehicle's cabin. Remote connectivity is made through 4G and the AT&T network, meaning those features can only be taken advantage of in the USA, coming with monthly use fee aswell. It has a screen, GPS and parking mode that, besides the usual purpose, can also be used to record the inside of the vehicle in case of a break-in [owla].

- **Thinkware F800 Pro**: Thinkware's F800 Pro is the brand's higher-end cloud-capable camera. It's also a dual-camera setup, with one pointing towards the front and another towards the back of the vehicle. It has no screen but comes with built-in GPS. The parking mode works through a motion sensor. It has driver assistance features namely lane departure warning, forward collision warning and safety camera alert that alerts the driver of upcoming speed and red-light cameras. The cloud features are mostly GPS and location-oriented, but also have a collision warning that informs the user remotely.

- **BlackVue DR750S-2CH**: The DR750-2CH by BlackVue is the best choice for a representative camera because it checks all the boxes necessary for the platform. It has GPS, Wi-Fi and cloud capabilities. Those capabilities are better described in the next subsection. It's a

dual-camera setup with cameras pointed towards the front and the rear of the vehicle. It has buffered parking mode and impact and motion detection [blab].

As it would be expected the cameras with cloud capabilities come at a much more premium price. In 2.1 and 2.2 the devices mentioned above are compared among themselves regarding their main features and thus put into perspective.

Table 2.1: Cloud Capable Dashcams Comparison

	BlackVue DR750S-2CH	**Thinkware F800 Pro**	**Owl Car Cam**
Screen	-	-	2.4"
Resolution & Frame Rate	1080p 60fps (front) 1080p 30fps (rear)	1080p 30fps (both cameras) (the front camera is 60fps if used without the rear camera)	1440p 30fps (outside) 720p 30fps (inside)
FOV	139º (front and rear)	140º (front and rear)	120º
Parking Mode	buffered	buffered	yes
GPS	yes	yes	yes
Wi-Fi	yes	yes	yes
Price(Approx.)	440 USD	410 USD	350 USD
Max SDCard Size	128 GB	128 GB	no SD card (64 GB internal)
Cloud Capabilities	yes	yes	yes

Table 2.2: Other Dashcams Comparison

	YI Smart Dash Camera	**Roav DashCam C1 Pro**	**Rexing V1 Pro**	**Garmin Dash Cam 55**
Screen	2.7"	2.4"	2.4"	2.0"
Resolution & Frame Rate	1080p 60fps	1440p 30fps	2160p 24fps 1440p 30fps 1080p 60fps	1440p
FOV	165º	145º	170º	122º
Parking Mode	-	yes	-	yes
GPS	-	yes	external (not included)	yes
Wi-Fi	yes	yes	yes	yes
Price(Approx.)	50 USD	100 USD	140 USD	200 USD
Max SDCard Size	64 GB	128 GB	256 GB	64 GB
Cloud Capabilities	-	-	-	-

2.2.3 Companion Applications

Each of the representative cameras selected above has an Android/iOS companion mobile application generally used to manage recorded video from the camera, as well as some other features. We'll now take a more in-depth analysis of these apps, thus getting a more concrete view of what the market has to offer and molding what the to be developed platform turned into. Usually, brands have one, singular application for all their different camera models unless those models differ from each other in particularly important aspects and the use of the same app is unintuitive or unpractical. For example, Thinkware has several different dashcam models, some of them with cloud capabilities such as remote video access, and for those cameras there is a specific application different from the app geared towards cameras without cloud capabilities. Looking into each app, and taking into consideration that the platform in which the platform is to be developed is Android, only the Android strand was analyzed, these were the results:

- **Garmin**: Garmin's app is called Garmin VIRB and is used, not only in dash cameras but in others such as action cameras as well, therefore it isn't as specialized in some aspects as other options. It connects via Wi-Fi and after connecting the user is presented with a video listing where he/she can watch, delete, share and export (the latter two are preceded by a selection of the intended portion of the footage). Videos and photos are timestamped and have GPS coordinates and instant velocity at the time of recording [garb].

- **Rexing**: The app Rexing uses is called TimaCam, it also connects via Wi-Fi and when started the user is presented with a screen containing two main sections, the Recorder section where the user can watch a live view, take a still photo, see the video/photos stored on the camera without downloading, download them and adjust the camera settings (there is an option to timestamp the videos), and the Local Albums where the downloaded items can be accessed [tim].

- **Roav**: Connects via Wi-Fi as well, allows firmware updates and adjusting camera settings, we can also manage videos and watch them without downloading. Videos show several aspects such as a map with the GPS location, time, instant velocity and G-Force, as well as a compass and an option to download or share the footage. There is also a section that allows the user to send feedback to the developers [roaa].

- **YI**: Connects via Wi-Fi with a password and is divided in three categories organized in tabs: Camera, which allows the user to view the camera's live feed and captures still photos, Album, where the video files stored in the camera are listed by date, and the user can view, download and delete them, Settings, where we can adjust the camera's settings. The camera's firmware can also be updated from the app [yia].

- **Owl**: Only supports a very limited amount of Android devices (Google Pixel 2/2XL, Samsung Galaxy S8/S9, and LG V30). The camera can connect either through Wi-Fi or 4G and allows live video viewing, watching footage in a timeline like format, or organized by

video files, the user can also trim and share a clip (it should be noted that there is a monthly limit of 60 total minutes of live video watched or downloaded via 4G, and after this limit is reached more "credits" need to be purchased to use this feature; it should also be noted that the use of the 4G capabilities has an extra cost of 99 USD per year). All these features can be accessed remotely from other places, not only when driving or inside the vehicle like the first four options [owlb].

- **Thinkware**: Connects via Wi-Fi, either by connecting to the camera's network or by using the phone's hotspot which allows retaining cellular connectivity and still have internet access. After connecting the user can watch a live view of the camera, adjust the camera's settings, and stream the video files stored in the camera, download them and watch them afterward, as well as deleting them. The cloud capabilities of the camera are as follows: Driving Impact Notification, where the user receives a notification if the vehicle is involved in an accident; Locate Vehicle, where the user is able to view the vehicle's location on a map via GPS; and Geo-Fencing, where the user can create a virtual geographic zone and be notified if the vehicle enters or leaves that zone. To use all these cloud capabilities the camera must have internet access via a wireless hotspot at all times [thi].

- **BlackVue**: The camera can connect to the app via Wi-Fi and the user can view the live feed or the stored videos, download them, adjust the camera's settings and update the firmware. The cloud features are attainable by creating an account and registering the camera via QR code or the camera's serial number, and by using a Wi-Fi hotspot in the vehicle. Through said features, the user can view the GPS location of the camera in the map, view the live feed and broadcast it to Youtube or Facebook, view all the stored clips, store them in the cloud or download them. It is also possible to send voice messages from the app to the camera and in that way establish a two-way communication, finally there are several alerts available through notifications such as manual recording, motion detection (only in parking mode), event recording (both in normal and parking mode), going over a certain speed (only in normal mode), parking mode toggle (it should be noted that the parking mode features are only available by acquiring either the OBD-II port adapter or an external battery pack; it should also be noted that there are a few plan options when it comes to cloud features, with the free plan having 1 camera per account, 10 minutes of live video per day, 100 times remove video playback/download per month and 5GB of cloud storage, among other specificities) [blaa].

In 2.3 all the mentioned options are compared with their main features highlighted. The only app with a relatively high rating in Google Play is the Roav app, the reason for such low scores on all the other ones are essentially trouble connecting or the given features not working fully. All the apps allow video management, stream, and download. The emergency alerts only work on cameras with cloud capabilities considering they're the only ones that allow remote use of the app. Live View is also a very common feature although in most cases only working when connected through Wi-Fi to the camera itself.

Table 2.3: Companion App Comparison

	BlackVue	Thinkware	Owl	YI	Roav	Rexing	Garmin
Google Play Rating	2.9	2.5	3.1	3.5	4.3	2.7	3.1
Live View	✓	✓	✓	✓		✓	
Still Photos of Live View				✓		✓	
Manage Videos*	✓	✓	✓	✓	✓	✓	✓
Watch Videos**	✓	✓	✓	✓	✓	✓	✓
Download Videos	✓	✓	✓	✓	✓	✓	✓
Edit Videos			✓				✓
Share Videos			✓		✓		✓
GPS	✓	✓			✓		✓
Emergency Alerts	✓	✓	✓				
Adjust Camera Settings	✓	✓		✓	✓	✓	
Update Firmware				✓	✓		
Two-Way Voice Communication	✓						

*includes listing, selecting and deleting videos, **refers to watching without having to download (streaming)

2.3 Video Compression

One of the most important aspects of video storage is video compression, as it allows for much more information to be stored in less storage. It is also vital in streaming services and it is only through video compression that website like YouTube and Netflix work. One of the ways compression works is through spatial compression, or intra frame compression, where each frame is compressed separately. Today, the most prevalent syntax for such use is JPEG [Wal92]. To compress a video frame, the same process that is used to compress a still image can be used. When a .JPEG is created, color information of the image is reduced in a process called chroma subsampling[WvK01] and afterward the image is segmented into equal-sized blocks of 8 x 8 pixels called macro blocks. The blocks are then transformed by a DCT [ANR74], and the DCT coefficients are quantized and transmitted using variable length codes. Another, more improved, compression method is temporal compression, or inter frame compression, which takes advantage of the large amount of temporal redundancy in video content. This was recognized as long ago as 1929 [Kel29]. Since much of the depicted scene is essentially just repeated in frame after frame, without significant change, video can be represented more efficiently by sending only the changes in the video scene rather than coding all regions repeatedly. The MPEG-4 standard [Sik97] is the most widely used method of audio and video compression. Like JPEG, the MPEG standard breaks a video frame into 8 x 8 macro blocks, and each macro block receives instructions on what to do with the pixels they already have. Those instructions can be to stay the same, moving, rotating, changing color, etc. The video frames with instructions like these are called P-frames. There are also I-frames, which are complete images, similar to a JPEG file, and B-frames, which are predictions or interpolations between P and I-frames. P-frames use much less data than I-frames and B-frames use even less, thus saving vast amounts of space in the process.

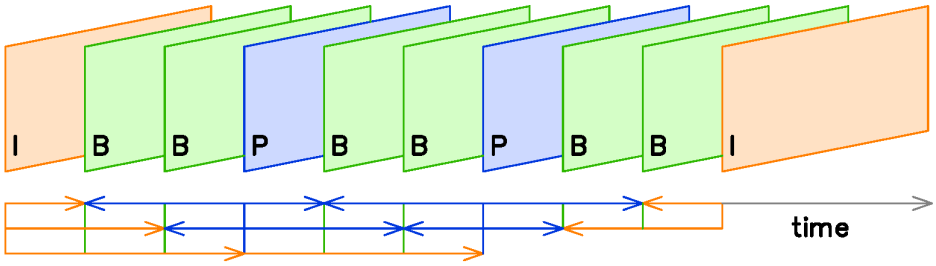


Figure 2.1: Interframe Compressions Diagram

Video compression cannot be mentioned without talking about codec and containers as well. “Codec” is a concatenation of “coder” and “decoder”, or “compressor” and “decompressor”, and is a "device or software that is capable of encoding or decoding a digital data stream". A container is a "metafile format whose specification describes how different elements of data and metadata coexist in a computer file" [HL16], and a video file normally consists of a container containing video data alongside audio data and metadata which consists of information about the video file such as bitrate, resolution, subtitles, etc. one of the most important pieces of metadata is the codec and, as mentioned above, it is responsible for compressing/coding the audio and video streams. There are many different codecs for audio and video files, and some of the most widely used, especially in dash cameras and other similar devices, are MPEG codecs. We’ll take a closer look into the two used codecs in all the representative cameras selected (H.264/AVC and H.265/HEVC) as well an open-source format developed by Google as a competitor to HEVC (VP9) and, another, open-source, emerging, format by the Alliance for Open Media (AV1).

2.3.1 H.264/AVC

H.264/AVC, which stands for Advanced Video Coding, is the most widely supported video coding standard, mostly because it provides a much better bitrate for the same file size than its predecessor, H.263. It’s a block-oriented motion-compensation-based video compression standard and is also referred to as MPEG-4 Part 10. Its applications cover broadcast, storage on optic and magnetic devices, conversational services, VOD or multimedia streaming, multimedia messaging services (MMS), etc. all these one several, different, mediums like ISDN, Ethernet, LAN, DSL, wireless and mobile networks, among other more recently developed features.[WSBL03];

2.3.2 H.265/HEVC

H.265/HEVC, which stands for High Efficiency Video Coding and also known as MPEG-H Part 2, is a successor to AVC and when in comparison it can offer double the data compression ratio at the same level of quality, or substantially improved video quality at the same bit rate. It is also a "block-based motion-compensated hybrid video coding" technology and, like AVC, HEVC is a

proprietary codec, and has a royalty associated with its usage, and since its inception in 2012 it isn't widely supported.[SOHW12];

2.3.3 VP9

VP9 is an open-source video coding format developed by Google. VP9 was originally developed for YouTube and its development was initiated to address the "explosive growth in consumption of video over the internet, including professional and amateur video-on-demand, as well as conversational video content"[MBG+13]. VP9 was finalized in 2013 and despite competing with H.264/AVC and H.265/HEVC, according to experimental results, the coding efficiency of VP9 was shown to be inferior to both with an average bitrate increase at the same image quality of 8.4% to AVC and 79.4% to HEVC.[GMM+13] Also, it was shown that the VP9 encoding times are larger by more than 100 times when compared to those of the x264 encoder (the most used encoder for H.264/AVC);

2.3.4 AV1

AV1, or AOMedia Video 1, is also an open-source video coding format designed for video transmissions over the internet. It was developed by the Alliance of Open Media, which was founded in 2015 by leading companies from various industries such as Google, Mozilla, Microsoft, AMD, Intel, Nvidia, Amazon and Netflix, with the purpose of developing royalty-free technology for multimedia delivery [av1]. AV1 was designed to replace Google's VP9 and to compete with H.265/HEVC and tests on a particular Multi-codec DASH Dataset concluded that, performance wise, AV1 is able to outperform VP9 and even HEVC by up to 40% [Fel18].

2.4 Storage

2.4.1 Video Metadata

Metadata can be simply put as "data about data", and is information that's used to catalog and organize other information, such as a video file. It can be used to optimize the searching of content by associating the metadata with relevant terms and tags. In the more concrete case of video files, metadata is just a label on it, it doesn't affect the content itself. It can be the video's title, description, tags and other, relevant, information[Tel08] [Tex15]. It's through extracting and preparing metadata that will be possible to have GPS information, timestamps, user identification, camera characteristics, video bitrate, etc. when handling video files in the system to be developed.

2.4.2 SD Card/Flash Memory

Taking into account that most dash cameras operate using an SD Card for storage, it's important to understand how these devices store video and data in general. The way they work is by recording

data into a solid-state chip inside the card using flash memory (flash memory is a type of non-volatile computer storage medium that can be rewritten in units called blocks). The flash memory records information when electrical charges change in its circuits and a similar process erases portions of memory for the rewriting process. The non-volatile, solid-state, chip inside the memory card contains many electrical circuits and when the card isn't in use the circuits retain their charges without any additional power. When a card is placed onto an activated device, such as a camera, a small electrical current from the device moves electrons in the flash memory chip and the digital patterns stored in the chip correlate to the data stored there [Liu06].

SD cards also have safety features and copyright protection, as copyright-protected works are written with a media identifier code that is placed on a protected portion of the solid-state memory. The copyrighted worked cannot be copied without access to the identifier code [EPS99]. Cards can also protect their content from erasing or modification and prevent access by non-authorized users.

2.4.3 Cloud Storage

Another option when it comes to dashcam video storage, and storage in general, is cloud storage. It's not used in many devices (dashcams) but it's one of the focal points of this dissertation. The developed platform was built around the idea of using the cloud to store, and access, video content, and the dissertation itself also contemplates an analysis and recommendation of standards and best practices in this interaction. The development process was documented with special attention towards the interaction with the cloud, and more specifically video handling.

Cloud storage works, essentially, by having data servers connected to the internet. A client then sends copies of files, over the internet, to the data servers, which store them so they can be accessed through a web-based interface. The server can either send the files back to the user or allow access and handling of those files on the server itself. The servers are usually in large quantities and are housed in storage facilities called data centers. One of the reasons for this large quantity of servers is the fact that servers occasionally require maintenance or repair in case of a malfunction or power outage, thus it's important to store the same information on multiple computers. This redundancy ensures that clients can access their data at any given time and, more importantly, preserve that same data [WPG+10].

However, getting large amounts of content moved over long distances is still a challenge. And to make it easier for companies to move data to the cloud, cloud service providers have storage units that allow customers to store their content and then ship them to the cloud storage company itself, to be stored in their own servers[Cou].

Looking at cloud services providers and the protocols supported for data streams upload:

2.4.3.1 Amazon Web Services

AWS allows services related to live video streaming, adaptive bitrate streaming, video encoding, video packaging, video compression, and video delivery. The ingest protocols supported are RTP

push, RTMP push or pull and HLS streams pull [Ama].

2.4.3.2 Azure Media Services

Azure enables the ingestion of live content, stream encoding into adaptive bitrate, recording and storing the ingested content and delivering the content through common streaming protocols to the users directly, or to a CDN. The protocols of video ingestion supported are RTMP and Smooth Streaming (fragmented MP4) [Kor].

Both RTP[SCFJ03] and RTMP[Ado14] are traditional non-HAS (chapter 2.5) IP-based streaming protocols, which means that the client receives media content, usually pushed by a media server. The difference between both rests on the fact that RTMP is connection-oriented and RTP is connectionless. Smooth Streaming[Mic15] and HLS[App15] are Microsoft and Apple's commercial HAS (chapter 2.5) solutions, respectively.

2.4.3.3 OpenStack

OpenStack is a set of several open-source software tools for building and managing private or public cloud platforms. It's backed by companies such as AT&T, Huawei, Intel, among many others, and managed by the OpenStack Foundation. It allows users to deploy virtual machines and other instances that handle different tasks for managing a cloud environment, making horizontal scaling simple, which means that concurrent tasks can easily serve more users by just executing more instances. Being open-source means that if anyone chooses to make any modifications to the source code, it can be done freely and shared with the community. OpenStack falls in the Infrastructure as a Service (IaaS) category of cloud computing, meaning that it makes it easy for users to quickly add new instances, upon which other cloud components can run. Usually, the infrastructure then runs a "platform" where developers can create software applications that are delivered to the end-users. OpenStack is modular in the sense that its architecture has several different components for different tasks. There are 44 in total, but the main ones are:

- **Nova** is the primary computing engine behind OpenStack. It's used as a full management and access tool to compute resources, handle scheduling, creation, and deletion of virtual machines and other instances.
- **Swift** is a highly fault-tolerant, available, distributed object/blob storage service. Rather than the traditional idea of referring to files by their location on a disk drive, developers can instead refer to a unique identifier referring to the file or piece of information and let OpenStack decide where to store it. This makes scaling easy, as developers don't have to worry about the capacity on a single system behind the software. It also allows the system rather than the developer, to worry about how to make sure that data is backed up in case of failure. It's ideal for storing unstructured data that can grow without bound;

- **Cinder** is a block storage component, which is more analogous to the traditional notion of a computer being able to access specific locations on a disk drive. This more traditional way of accessing files might be important in scenarios in which data speed is the most important consideration. It virtualizes the management of block storage devices and provides end-users with a self-service API to request and consume those resources without requiring any knowledge of where their storage is actually deployed or on what type of device.

- **Neutron** provides the networking capability for OpenStack. It helps to ensure that each of the components of an OpenStack deployment can communicate with one another quickly and efficiently.

- **Horizon** is the dashboard behind OpenStack and its only graphical interface. Developers can access all of the components of OpenStack individually through an API, but the dashboard provides system administrators a look at what is going on in the cloud and to manage it as necessary.

- **Keystone** provides identity services for OpenStack. It authenticates and authorizes all OpenStack services and is also their endpoint catalog.

- **Glance** provides image services to OpenStack. Referring to images of hard disks. It allows these images to be used as templates when deploying new VM instances. It stores and retrieves them from a variety of locations from simple filesystems to object-storage systems like Swift.

2.5 Streaming

Video streaming is another important aspect of the way dashcams work, and video is managed overall. The ability to play video without having to download it is highly valued in today's content-driven landscape. HTTP Adaptive Streaming (HAS) has quickly become the dominant approach for video streaming due to its adoption by leading service and content providers. It uses HTTP as the application and TCP as the transport-layer protocol, the data is hosted in a standard HTTP server, from which the client pulls. On the server side, it uses conventional web servers available within the networks of ISPs and CDNs. On the client side, each segment is requested and fetched independently from others and maintain the playback session state. It also doesn't require a persistent connection between the client and the server.

2.5.1 DASH

Dynamic Streaming Over HTTP (DASH) is a standard developed by the Motion Pictures Experts Group (MPEG) that provides an open specification for adaptive streaming over HTTP, leaving to the developers, the adaptation logic. The DASH Server is an HTTP server that hosts the media segments, each encoded at multiple bitrate levels listed in the Media Presentation Description (MPD).

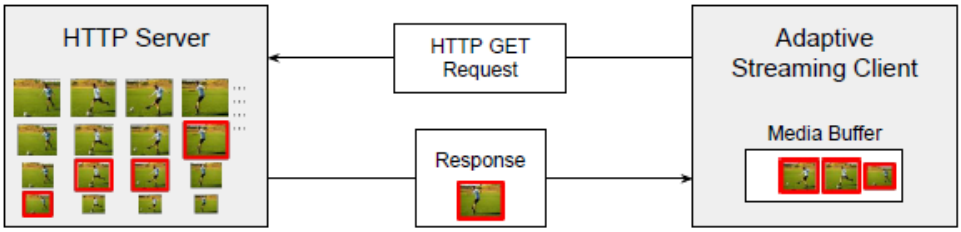


Figure 2.2: HTTP Adaptive Streaming

The DASH Client then issues timed requests and downloads the required segments, described in the MPD, from the server, using HTTP GET messages[BTB+18].

2.6 Content Delivery Networks

It's through Content Delivery Networks (CDN) that content is delivered to users all over the world. A CDN is a system of distributed servers that deliver web content based on geographic locations. Instead of accessing the origin server every time a user requests a certain content, a cached version, in a server at a closer physical location, is used, thus reducing the response time and providing an overall better user experience. What, usually, happens is: content owners, such as media companies, pay CDN operators to deliver the content, and a CDN pays ISPs and other entities for hosting its servers in their data centers [Inc].

2.7 Connectivity

2.7.1 Wi-Fi Direct

All the Wi-Fi enabled dash cameras connect directly to a user's smartphone (thus enabling the use of their respective companion app) through Wi-Fi Direct. Wi-Fi direct is the standard used that allows, two or more, Wi-Fi enabled devices to connect to each other without requiring a wireless access point. Wi-Fi becomes a way of communicating wirelessly, similar to Bluetooth. Wi-Fi direct embeds a software access point into any device that supports it (to establish a connection between devices only one of them needs to support Wi-Fi Direct). Then, the Soft AP provides a version of Wi-Fi protected setup with its push button or PIN-based setup. When a device enters the range of the Wi-Fi Direct host, it can connect to it and gather setup information. Soft AP's can be as simple or as complex as required. The standard also provides WPA2 security, outperforming Bluetooth in yet another aspect [Wi-].

2.7.2 WAN Connection

A WAN(Wide Area Network) is a communications network that extends to a wide geographical area. Several different types of entities, those being corporations, universities or governments,

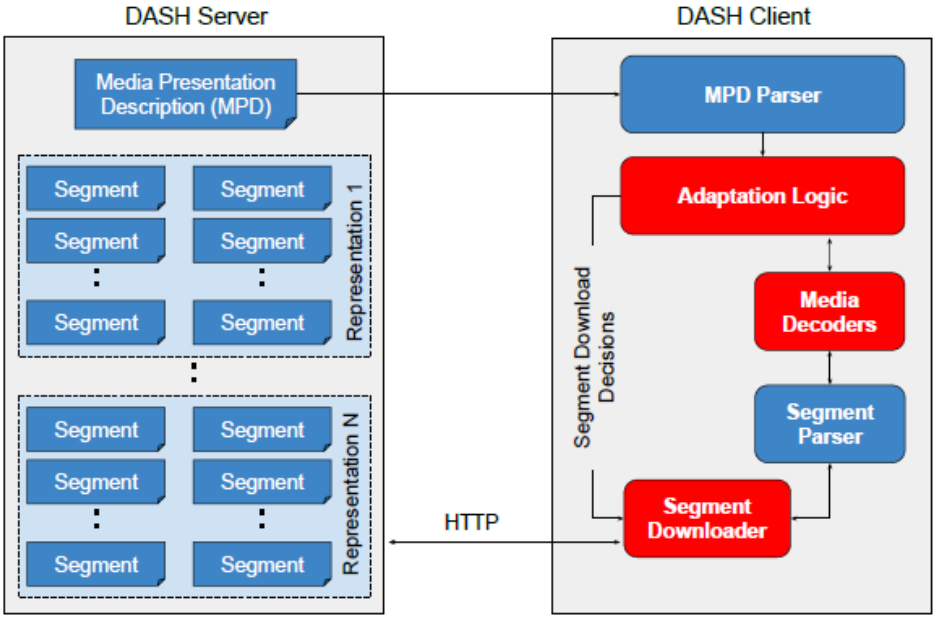


Figure 2.3: Dynamic Streaming Over HTTP

among many others, use WANs to relay information anywhere across the world. Most professionals define a WAN as "any network that uses routers and public network links". WANs can be centralized or distributed. A centralized WAN consists of a central computer to which the other computers connect. And a distributed WAN consists of many interconnected computers in many locations. The Internet is considered to be a distributed WAN and that's why it is a relevant topic in this platform because the access to the platform will work mostly via an internet connection to the server where all the relevant data will be stored [GS05].

2.8 Related Work

Despite the fact that there isn't very much work done in the particular topic of dashcam video handling, a few examples can still be numbered.

A different type of video handling, but still done using dashcams, can be seen in [CCC+14] and [CCL+15]. In both cases what was attempted was the combination of video clips to give the illusion of see-through vehicles, in order to improve safety and comfort in driving by somewhat extending the field of view. The main challenge, for both teams, was the high level of discrepancy between the two video feeds. The developed methods work by contouring an unobstructed view, estimating local homography transformations, adjusting perspective and bound it to the obstructed area in [CCC+14], while in [CCL+15] the approach revolves more around frame-by-frame motion estimation and thus the perspective difference between the two clips can be approximated.

In [KLY+17] an automated public service that enables sharing of dashcam video anonymously. Each video takes a compact form and that form is treated as an entity, instead of their owners, and

Figure 2.4: Results of [CCC+14] (left column) and of [CCL+15](right column)

thus used for search, verification, etc. The tool exploits the line of sight of each clip with nearby ones that share the same sight to build a map of visibility around a given event. Videos are never transmitted unless they are verified to be of importance towards the incident and are anonymously solicited.

Lastly, in [PKML16], the main motive is also the sharing of dashcam video but more oriented towards the reasons behind that sharing process and any concerns that come from it. The team conducted tests and the results showed that "reciprocal altruism/social justice" and "monetary reward" were the key reasons, with the former supplanting the latter. And that groups with greater privacy sensitivity had a lower reciprocal altruism and social justice motives but a higher monetary motive.

2.9 Summary

Most of the research conducted so far focused more on the engineering aspect of the dissertation, namely how dash cameras work, what are their features and what technologies are relevant to the topic, meaning that it is not yet completely solidified. The scientific aspect will focus even more on video handling in the cloud, most notably, upload, streaming, etc.

The research of hardware was an efficient method of understanding, not only, how dashcams operate, but also, what use cases and features should be in the platform to be developed thus, helping delineate the steps for its specification and implementation.

Understanding the technologies and protocols behind many of the relevant topics in the theme also helps that delineation process, as it gives a more clear landscape of cloud storage and video handling thus aiding in choosing how to implement the desired system.

Chapter 3

Solution

3.1 Overview

This chapter describes what the problem to be solved is, by combining the needs of the host company and the technological aspects of the dissertation. Afterward, it explains what the solution to resolve the aforementioned problem is, using the knowledge obtained from the conducted research. It, then, goes over the initial requirements of the platform and proceeds to explain why some of them didn't apply to the final product. It ends with the work plan that helped guide the development of the dissertation and a description of each step, along with changes that were made in the development phase.

3.2 Problem & Solution

After the research and analysis described in the previous chapter, the problem to be solved became more clear and more easily definable.

The problem revolves around the lack of a dashcam video management system that is independent of the source and is also cloud-centered, with most of the current solutions being proprietary systems and focused around local storage. The solution then became a proof of concept of a video management system, that handles video through and from the cloud, and interacts with an already existing app, to provide a more compact and pleasant user experience. Alongside the implementation of the proof of concept it's also important to document the entire process with special attention to the use of cloud features, such as upload, streaming, handling, etc. and in doing so, provide insight on the best practices to do it along with an overview of standard protocols and techniques.

3.3 Requirements

3.3.1 Functional

Functional requirements are defined as something the system should do. In other words, a functional requirement will describe a particular behavior or function of the system when certain conditions are met.

3.3.1.1 R01 - Login

Requirement As a user, I want to log into my account, so that I can access my information and manage my cameras and clips.

Description By entering his/her credentials a user can log into his/her account and access all the application's contents.

Pre-requisites

- The user is logged out
- The user has an account

Acceptance criteria

- Display empty text boxes labeled accordingly for the user to input his/her credentials;
- Display a check box that can be marked in case the user wants the credentials to be memorized and not have to insert them again;
- Display an error message in case the credentials aren't correct;
- After the credentials are approved redirect the user towards the applications main page.

3.3.1.2 R02 - Cameras

Requirement As a user, I want to be able to associate one or more cameras with my account, so that I can save and manage their respective video clips.

Description The user should be able to have several different video sources (cameras) associated with his/her account whose video will later be managed. The user should be able to add new ones, or manage the already existing ones as well as deleting them.

Pre-requisites

- The user is logged in

Acceptance criteria

- Display a list of cameras already associated or an empty list, both with a button to add a camera;
- When the button is clicked, display a form to be filled with information about the camera;
- After filling the information, attempt to connect to the camera;
- Add the camera to the list of cameras, displaying its name, and an icon if it exists;
- If there is a connection made with the camera display an indicative symbol;
- Clicking any camera should trigger a pop up displaying the information about the camera and an option to connect or disconnect depending on the current situation.

3.3.1.3 R03 - List Clips

Requirement As a user, I want to be able to list all saved video clips, so that I can manage them.

Description The user should be able to list all videos and filter the results by: source (i.e. one or more of the associated cameras), time and date, GPS location, event tag (if one of the five types of events (abrupt cornering, lateral movement, hard acceleration, hard braking or speeding) is detected and tagged in a particular) and storage (locally or cloud).

Pre-requisites

- The user is logged in

Acceptance criteria

- Display the list of all the video clips;
- Display an icon that opens a menu that allows the user to filter the results. The results can be filtered by:
  - Camera, one or more of the cameras associated with the user;
  - Time and Date;
  - GPS Location;
  - Event, all events or events of a particular type;
  - Storage, local storage or cloud.
- When a video from the list is clicked, the user is presented with a clip view popup, the should contain a preview (thumbnail) of the clip and some options (play, download, info and delete);

3.3.1.4 R04 - Stream Clips

Requirement As a user, I want to be able to stream a video clip, so that I can view without having to download it.

Description After listing the videos, the user should be able to select one and play it, by streaming it from the cloud storage, thus not having to download it into the device's local storage.

Pre-requisites

- The user is logged in
- There are videos stored in the cloud storage

Acceptance criteria

- Play the selected clip by streaming it from the cloud storage when the "play" button is clicked in the clip view referenced in R03.

Exceptions

- When the selected video is stored locally there is no need to stream it and it should instead be played from the local storage.

3.3.1.5 R05 - Download Clips

Requirement As a user, I want to be able to download a clip into the local storage, so that I can use it later.

Description After listing or viewing a video clip, the user should be able to download it into the device's local storage.

Pre-requisites

- The user is logged in
- There are videos stored in the cloud storage

Acceptance criteria

- Download the selected clip, from the cloud storage into the local storage, when the "download" button is clicked in the clip view referenced in R03.

Exceptions

- When a video is already stored locally, it shouldn't be possible to download it again without deleting it from the local storage first.

3.3.1.6 R06 - Play Local Clips

Requirement As a user, I want to be able to play a clip stored locally, so that I can view the video I've previously downloaded.

Description The video listing should also display clips that have been downloaded and play them as well, but by accessing the version stored in the device instead of streaming the cloud-stored one.

Pre-requisites

- The user is logged in
- There are videos stored in the device's local storage

Acceptance criteria

- Play the selected clip from the local storage when the "play" button is clicked in the clip view referenced in R03.

3.3.1.7 R07 - Delete Clips

Requirement As a user, I want to be able to delete video clips stored in the cloud or in the local storage when I don't need them anymore, so that I can save space.

Description When listing the video clips the user should be able to select and delete any particular one.

Pre-requisites

- The user is logged in
- There are videos stored in the cloud or in the local storage

Acceptance criteria

- When clicking the "delete" button in the clip view referenced in R03 one of two things should happen:
  - If the clip is stored locally it should be deleted from the local storage only, while the cloud version remains stored.
  - If the clip is stored in the cloud only, it should be deleted from the cloud storage.

3.3.1.8 R08 - Navigable Video Player Bar

Requirement As a user, I want to be able to use a navigable progress bar when playing a video clip, so that I can select different parts of the clip accordingly.

Description There should be a, navigable, progress bar in the video play that can be used to view what part of the clip is being played and to select what point should be played.

Pre-requisites

- The user is logged in
- There are videos stored in the cloud or in the local storage

Acceptance criteria

- Display a progress bar, when a clip is being played, that shows what point of the clip is currently being played;
- When clicking in any point of the progress bar that very point should be played.

3.3.1.9 R09 - Clip Information

Requirement As a user, I want to be able to view a video clips information, so that I know more about it.

Description When selecting the information section of a clip the user should be presented with the video's title, description, duration, time and date, GPS location, and event tags, if there are any.

Pre-requisites

- The user is logged in
- There are videos stored in the cloud or in the local storage

Acceptance criteria

- Display video information when the user click the "info" button in the clip view mentioned in R03. The information should be:
  - Title;
  - Description;
  - Duration;
  - Time;

- Date;
- GPS location;
- Event tags (if they exist).

3.3.1.10 R10 - Edit Clip Information

Requirement As a user, I want to be able to edit a video's title and/or description, so that I can more easily identify and recognize them.

Description When in the information section of a clip, the user should be able to change the clip's title and/or description.

Pre-requisites

- The user is logged in

Acceptance criteria

- When viewing a clips information there should be an icon indicative of the ability to edit the title and description of the video;
- When clicking that icon the selected field should be made available to edit, and confirmed afterwards;
- When clicking that icon the selected field should be made available to edit, and confirmed afterwards;

3.3.1.11 R11 - Trip View Video Selection

Requirement As a user, I want to be able to view a clip of a particular portion of a trip by selecting that same portion in the trip details view, so that I can get faster and easier access to the clip I desire.

Description When in the trip details view, the user should be able to select any portion of the trip and be presented with the video clip corresponding to that segment.

Pre-requisites

- The user is logged in

Acceptance criteria

- Display the clip view mentioned in R03 corresponding to the GPS location and/or timestamp of the trip section selected

3.3.1.12 R12 - Trip View Event Selection

Requirement As a user, I want to be able to select a tagged event from the trip details view, and view the video clip from that particular incident, so that I can get faster and easier access to the clip I desire.

Description When selecting any of the event tags in a trip, in the trip details view, the user should be presented with the clip tagged with the aforementioned tag.

Pre-requisites

- The user is logged in

Acceptance criteria

- Display the clip view mentioned in R03 corresponding to the clip tagged with the selected tag specifically.

3.3.1.13 R13 - Trip Recap

Requirement As a user, I want to be able to replay the video content equivalent to an entire trip, in real time or by fast forwarding, do that I can get a better idea of what the overall trip looked like.

Description By looking at the details of a full trip the user should be able to view, and fast forward, the video of the entire trip.

Pre-requisites

- The user is logged in
- The user has made at least one full trip

Acceptance criteria

- Display an icon indicative of the option to replay the entire trip in the trip details view;
- Play the video clips of the entire trip when click said icon;
- Display icons, in the video player, indicative of the video playback speed;
- Change the playback speed of the video according to the icons clicked.

3.3.2 Non-Functional

On the other hand, non-functional requirements describe how the system should behave and imposes constraints upon that behavior.

- Store Videos - The system should store recorded videos so that the user can access them later.
- Assign Video Information - The system should assign certain information to each video clip. This information includes a timestamp, GPS coordinates, the trip it belongs to, an event if appropriate, etc.

3.3.3 Requirements Revision

During all stages of development, the requirements listed above were used as guidelines on what to do and how to do it. However, in the end, not all of them were applicable, more specifically:

- R01 was implemented and used in the web app, however, the final app used an already full-fledged user database and login system, therefore, the implemented login wasn't used in the app.
- In the early stages of development, both the data model and the backend were prepared to implement R02, however, since the camera connection wasn't a priority and more of a 'nice to have' it ended up not being in the final product.
- Despite being implemented in the backend API and used in the web app, R07 isn't used in the app so that all features function properly, for example, if a user deleted a particular clip from the database the trip replay wouldn't work and neither would the viewing of an event clip, however, any downloaded clip is deletable via a file management app, as are most files.
- Since the video listing, in the app, doesn't work exactly as initially thought, R10 became less relevant and therefore, wasn't implemented.

The other requirements were implemented and are used in the final app, somewhat similar to what was initially thought out.

3.4 Use Cases

In order to better understand how the described solution works in practical terms, some simple use cases could be helpful and illustrate how a user might interact with it. These use cases are generated by reviewing the revised functional and non-functional requirements and associating them with the mobile app to obtain a realistic and fluid user experience

- **UC1:** After the user has selected a trip and is on the map screen of said trip, the user will select one of the marked events by clicking on it. The system will respond by highlighting

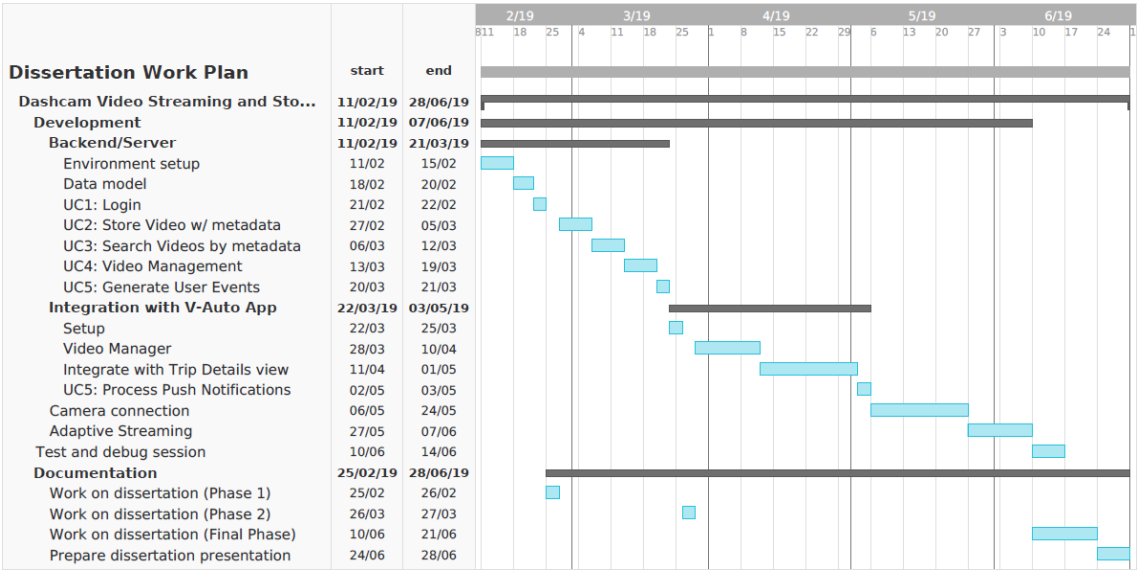


Figure 3.1: Work Plan

that specific event a showing its severity and by streaming the video clip associated with it. The user will then navigate through all the events of the trip by clicking on each marker. The system will respond by refreshing the information presented, namely the severity and name of each event and the video clip on which it occurs.

- **UC2:** After the user has selected a specific event by clicking on its marker on the map screen of a specific trip, the user will click on the icon displayed in the top-right corner of the video player. If it's the first time, the system will respond by requesting the user permission to store files in the local storage. After the user accepts the system will download the video file that corresponds to the video clip being streamed at that moment.

- **UC3:** After the user has select a trip and is in the map screen of said trip, the user will click on the button placed in the bottom of the screen labeled as "Replay Trip". The system will respond by tracing the polyline that corresponds to the trip and, simultaneously, streaming the video clips associated with the trip, in chronological order, and synchronized with the map animation. The user will then select a different portion of the trip by clicking on a point of the progress bar representative of the progress of the trip. The system will respond by updating both the animation of the trip and the video clip playing to match the desired time of the trip.

3.5 Work Plan

3.5.1 Backend

The first step of development focused on the backend/server-side. It began with the environment setup to prepare for development itself. Afterward came the data model definition.

This stage consisted of the definition of the video data model, namely what metadata to assign to the video clips. Besides the usual title and description, other important pieces of information are GPS coordinates, time and date stamps, a boolean variable in case the clip contains an incident or not, among others. The database specification was also included in this step. It was important to define the relational DB structure, particularly what tables exist and what is stored in them. Some examples are the Clips or the Events table, where relevant information would be stored about both entities and several events could be assigned to one clip as well as one clip to several events. The DB was implemented with PostgreSQL [Thea].

The next stage was the implementation of the backend use cases themselves, for example, login, upload of video to the cloud with the previously defined metadata, the search of clips using different metadata, video management, handle use events, etc. The technology used in this section was Spring Boot [Piva].

3.5.2 Frontend Integration

The next stage of development contemplates the frontend side, more specifically, the integration of the work done so far with the, already existing, V-Auto app [Vod], developed by WIT-Software. This app records car trips with some details and also contains a map view of a certain trip along with user events detected by an external sensor.

After the environment setup required for Android development, the video player aspect was created. Followed by the integration with the, already mentioned, maps screen as well as adding new features to this screen.

3.5.3 Other Features

After the main implementation process, there are a few other features, that despite being accessory to the proof of concept (considering that the main aspect of the PoC is the video management and the handling of video in the cloud), are still useful and relevant. There is some time allocated for their development and these features can be, for example, the connection to the camera itself and the use of adaptive streaming to play the video.

3.5.4 Work Plan Revision

Despite the somewhat restrictive nature of the initial work plan, there was some wiggle room when it came to the latter part, namely the 'Other Features' mentioned above. That time ended up being used elsewhere.

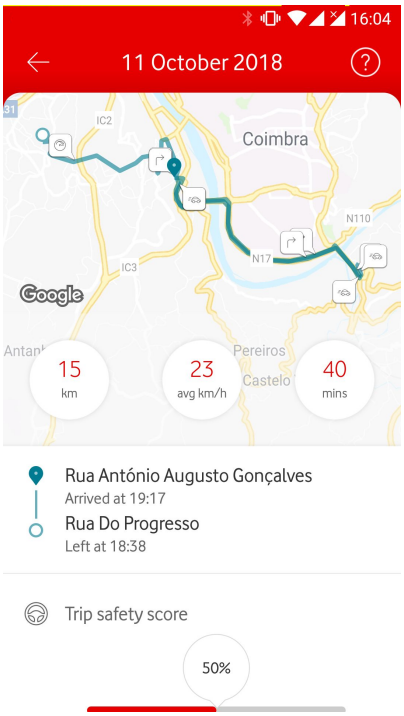


Figure 3.2: Example of a Trip Details Map View of the V-Auto App, with a highlighted trajectory and user events

The time allocated for the Backend/Server implementation was, mostly, used accordingly, with the only significant difference being the implementation of a web app to serve as a development auxiliar and a way to demonstrate the work done thus far. That delayed the app integration a bit which, in itself, took longer than anticipated and happened concurrently with the cloud implementation.

3.6 Summary

This chapter not only describes what the solution to be implemented is, alongside its specification and requirements listing but also provides a guide to that same implementation. It goes without saying that despite the work plan being somewhat detailed, and maybe even a bit restrictive, all aspects were flexible and modified according to the development's needs, as described.

Parallel to this implementation there also was the writing process, with a special focus to the aggregation of the most predominant protocols and techniques when it comes to cloud storage and video handling in the cloud, with a recommendation/reference to the best practices in the implementation of similar systems.

Chapter 4

Implementation

4.1 Overview

This chapter delves into the implementation process itself. It describes each step in detail giving insight into each decision and what the result does and how it does it. It goes over the changes that happened to the database, to the server and its requests, the cloud implementation, and the android app.

4.2 Database Structure Definition

Before the implementation itself began, it was important to define what the database would look like. The first decision was to make it a relational database, not only for the ease of use and familiarity, which would therefore, result in a faster and cleaner implementation, but also for performance and support reasons, alongside the fact that the data to be stored wouldn't require a different, more complex, approach.

After this first step, the next one was to define the tables and the connections between them. The first draft consisted of seven tables: Users, Cameras, Clips, Trips, Location, Timestamp, and Events.

There would be several one-to-many connections, between Users and Cameras, Cameras and Trips, Cameras and Clips, Trips and Clips, Events and Clips, Timestamp and Clips, and Location and Clips. Overall this is how it would work: each user can have multiple cameras, however, cameras can only be associated with one user, several trips and clips can be associated with one, and only one, camera and several clips could also be associated with one trip. Each clip could be associated with one location but each location could have multiple clips associated with it. Just like with timestamp, each clip would only be associated with one timestamp, however many clips could be associated with the same timestamp. Lastly, each clip could be associated with zero or one event, however, many different clips could be associated with one event, as there are only a few limited types.

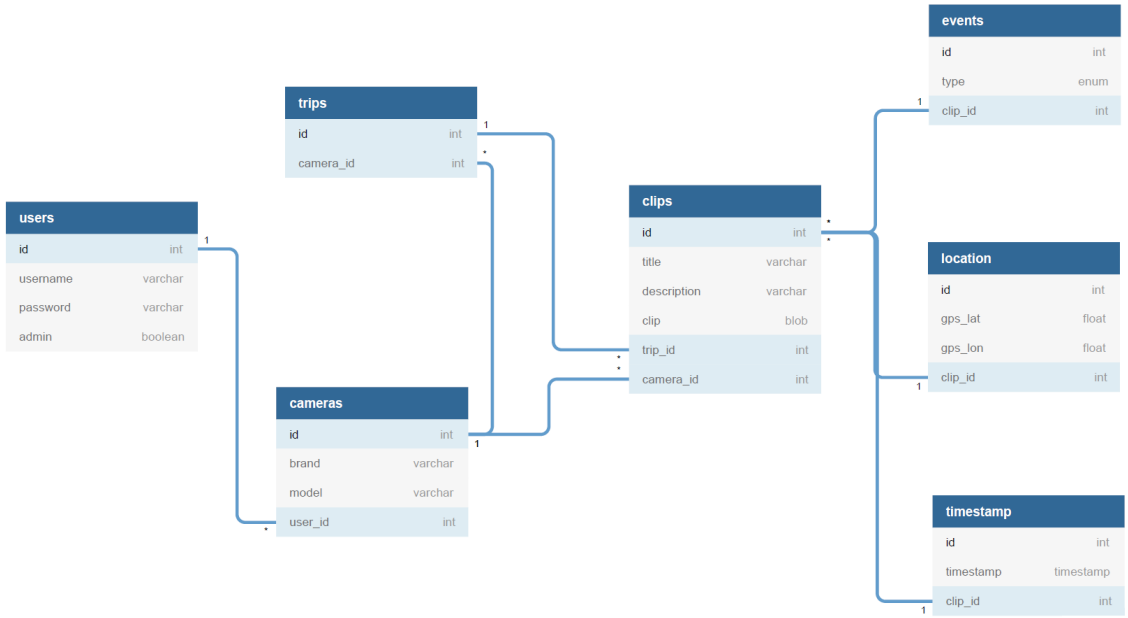


Figure 4.1: Database Model 1

After the first draft, some aspects were reviewed and subsequently altered. The Location and Timestamp tables were removed, because the video file itself, present in the Clips table already has that information in its meta-data and can, therefore, be accessed without the need to store it in an additional table. The Events table was also removed because there would be no need for an additional table when this piece of information consists simply in a tag, thus a simple field in the Clips table would suffice. Lastly, the one-to-many connection between Cameras and Clips was also removed because, after second thought, it wouldn't make much sense to have clips not associated to any trip, so by taking a more minimalistic approach we can obtain a solution that makes sense and is simple and manageable without being oversimplified.

Later, in the midpoint of the semester, and in the transition from backend development to the integration with the V-Auto app, the data model was again revised and modified in some aspects. Trip would no longer be a table and would instead become a column in the Clip table, since the only information about a trip, that isn't already in the Clip table, is its ID. Therefore the connections Cameras-Trips and Trips-Clips would become a single connection Cameras-Clips, where a clip can only belong to one single camera, and one camera can have many clips associated with itself. Another adjustment was that Events would become a table, connected to Clips, so that one Clip could have multiple Events occurring within itself, something that wasn't allowed by the previous model.

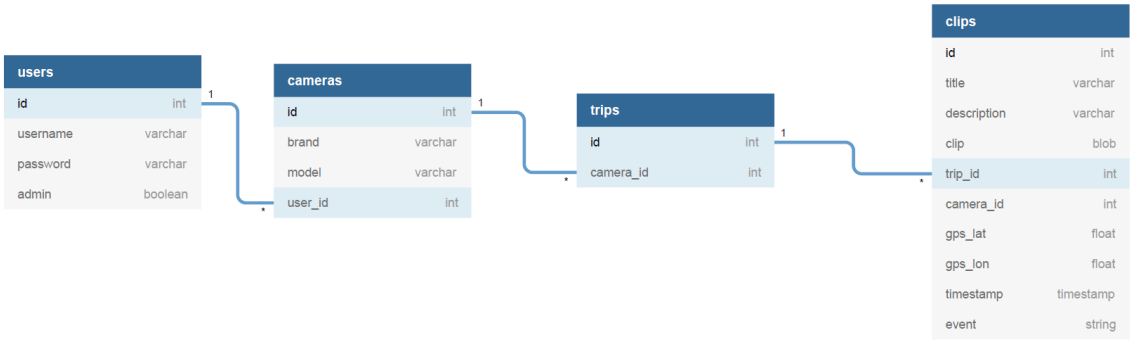


Figure 4.2: Database Model 2

4.3 Database Setup

After the definition of the Database structure, the next step was the implementation itself. PostgreSQL was the selected Database Management System(DBMS). Despite having worked with other relational DBMSs such as MySQL and SQLite before, and even a graph DBM in Neo4j, PostgreSQL was chosen out of personal preference and familiarity with the tool. A graph database wasn't considered because the data to be stored isn't meant to be used and accessed that way. Each user has his own clips, trips, cameras, etc. and that information can be, more simply, mapped to tables and columns.

The first procedure was creating a database that would run locally during the implementation process, and then the creation of the defined tables and columns followed by populating them. Even with the structure definition in the previous section not all the columns and their data type were defined until the development required it. For example, the timestamp column in the Clips table was initially a String but was later altered for the native Java and SQL type Timestamp for more easily implemented and intuitive comparison operations.

The database was populated with mock information and video footage from a real trip to be later used in the app. The last step was exporting the DB and importing it into the OpenStack instance so that it would be accessible anywhere.

4.4 Server Environment Setup

Server implementation was the first major step in the development process as a whole. It consisted mostly of implementing a RESTful API to use and manipulate the data in the DB. Since there was no previous, wide, experience with any particular framework, with a suggestion from the company coordinator, the selected tool ended up being Spring Boot. Spring Boot is a Spring project for creating stand-alone, production-grade Spring-based applications. It's preconfigured with the best configurations and third-party libraries so that applications need very little configuration [Piva].

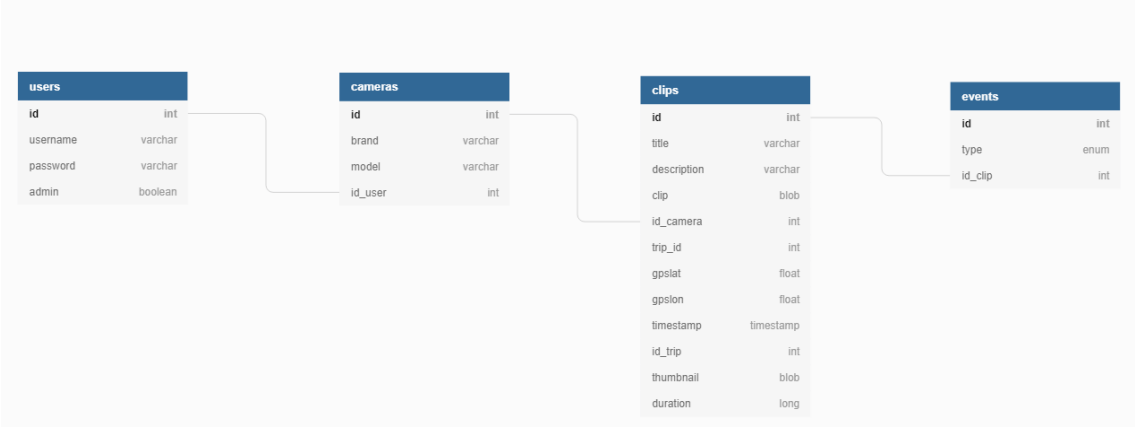


Figure 4.3: Database Model 3

4.5 Server Implementation

4.5.1 Authentication

Considering that all the functionalities of the platform are to be used by authenticated users only, authentication is a vital part of the implementation process, however, this method of authentication isn't used on the final product because there already is a different one in its place. This method is used, essentially, in the web interface. For testing purposes, a user with administrative permissions was created. Regular users can only view, edit or delete their own information and data, and when adding new data that is correlated with an entity that is associated with any user, it must be with the one attempting to add the new data, i.e. when a user attempts to add a new trip it has to be associated with a camera, and the camera must belong to the same user.

The authentication is done using Spring Security which is also a Spring project and allows customizable authentication as the standard for securing Spring-based applications [Pivb].

The password encoding is done through the java bcrypt implementation. Bcrypt is a password hashing function designed in 1999 [PM99] that, besides other features, is an adaptive function, which means that over time, the iteration count can be increased to make slower, so it remains resistant to brute-force search attacks. It's the default password hash algorithm for several systems, including some Linux distributions, and has implementations in several programming languages (C, C++, Java, JavaScript, Python, etc.).

4.5.2 Entities

The following entities evolved with each iteration of the Database model, and in each of them, there was an entity representative of each table. Each entity has a unique generated ID, except for the Event whose ID is given based on a pre-existing ID and given by the app when accessing a trip's map.

4.5.2.1 User

This entity represents a User of the platform. Each one has a unique Username (String), a Password (String) and a List of Cameras. There is also an admin boolean that is false by default and true for administrators to grant them the proper permissions, however, the admin role only exists for development purposes and isn't present in the final platform.

4.5.2.2 Camera

This entity represents a Camera. Each one has a Brand (String), a Model (String), one associated User (each camera can only be associated with one particular camera), and a List of Clips (in this previous version the Camera would have a list of Trips instead).

4.5.2.3 Clip

This entity represents a Video Clip. Each one has a title (String), a description (String), GPS coordinates (latitude and longitude), a timestamp, a trip id, the clips duration, the clip itself, a thumbnail, and one or more associated events.

The GPS coordinates are stored as doubles, the timestamp as a Timestamp. The trip ID and the duration are stored as longs. The video file itself and the thumbnail are stored by using the Large Object feature in which the binary data that comprises the file is stored in a separate table in a special format and refers to that table by storing a value of type OID in the clip column of the Clip table [Thec].

4.5.2.4 Event

This entity represents an Event. Each one has a type (Abrupt Cornering, Lateral Movement, Hard Acceleration, Hard Braking, and Speeding), and one associated Clip (each event can only be associated with one particular clip).

4.5.3 REST Requests

After the database was set up and the entities instantiated the next move was creating the requests that would be responsible for inserting, deleting, viewing and handling those entities. All those requests are implemented in the Controller (see 4.8). All the requests responsible for the main features are in *UserController*, however, to help in the development process AdminController implements some features that users don't have like adding and deleting users and accessing all the data in the database being their own or not. To aid in the demonstration made in the midpoint of the semester LandingController was also implemented and is responsible for handling the URL in some cases and redirects to the user dashboard after the login. The requests responsible for the main features of the platform, and more specifically, the mobile app are:

- **Upload Clip:** Implemented in *newClipInfo()*, it receives a video file and analyses it to obtain all the data that will fill the corresponding row in the DB's table. It fetches the file's name,

type, timestamp and duration, and then uploads it as a Large Object [Theb], and generates a thumbnail from a singular frame of the video clip and uploads it in the same manner. It also creates a new Event for each new Clip, whose information will later be updated.

- **Stream Clip:** Implemented in *viewClip()*, it receives the ID of a singular clip and reads its buffer outputting it through and OutputStream without saving it to any file.
- **Get Encoded File:** Implemented in *getFileBuf()*, it works similarly to *viewClip()* but saves the output to a new file created for that purpose. It was later altered to return an encoded string that corresponded to the file so that it could be saved to a file in the android app implementation(or any other outside the API for that matter).
- **Update Trip ID:** Implemented in *updateIdTrip()*, it receives two timestamps (corresponding to beginning and ending) and a Trip ID, it then fetches all the Clip IDs and timestamps from all the Clips and if Clip's timestamp fits with the two given, it updates its id_trip to match the given one.
- **Create New Event:** Implemented in *newEventUpdate()*, it receives an Event ID, a Trip ID, an Event type, and two timestamps, it then fetches all the IDs, timestamps, associated Event IDs and duration of all the Clips with that Trip ID, and updates the associated Event's information with the received data.
- **Get Clip Name:** Implemented in *getClipNameById()*, it receives a Clip ID and fetches the file name associated with it. This is useful when checking if a particular file is saved in the local storage.
- **Get Event's Clip ID:** Implemented in *getClipIdByEvent()*, it receives an Event ID and fetches the ID of the Clip associated with this event. This is useful when playing the Clip associated with an Event.
- **Get Clips from a Trip:** Implemented in *getUserTrip*, it receives a Trip ID and then fetches all the IDs and timestamps of Clips associated with that Trip, orders their ID by timestamp and returns that a list with those IDs. This is used when playing an entire trip.

There are more requests for tasks like adding new Entities or deleting them, as well as login and logout, and GETs for clips according to all the, stored, video information such as GPS coordinates, time, date, etc.

4.5.4 Web App

During the first stage of development (the backend/server section) it was viewed as useful the implementation of a rudimentary web interface, with the purpose of more easily interacting and to serve as a demonstrative tool for work done thus far.

The web app was implemented using JSP on top of what had already been developed with Spring Boot.

The interface consists of simple pages that illustrate the main features of the platform. It's possible to log in and log out securely, be directed towards a dashboard where the user's data is presented, organized by Cameras, Clips and Trips. The user can add or delete each of the categories. The Clips are where most of the features are. The user can upload clips, get information about them such as GPS coordinates, Timestamp, etc. search them by Data, Date and Time, Location and Event, Download them, watch them directly from the server, without the need to download the file and Upload a new file.

To access all functionalities a login was required, working for regular users and administrators alike. Afterward, both types having different pages they can access, with the regular users only being able to access and handle entities associated with their own ID. An admin is able to perform any task a regular user is, for any user, as well as adding and deleting regular users. If not logged in any URL redirects towards the login screen. The *LandingController* is responsible for handling some URLs that didn't automatically redirect to the login.

The entire web app was made with two purposes, firstly to serve as a development tool, so that vital tasks, such as viewing data, uploading and streaming, were more easily completed and thus time saved to be used developing other aspects of the platform. Lastly, to help in better demonstrating what the backend could do while the Android app wasn't being developed or was still in its early stages.

4.6 Android Media Player

The core feature to be implemented revolved around video playback, therefore, a video player for android was the first thing that came to mind when the frontend integration began. To expedite the process and better understand how to implement video players in android, an isolated, small, project was implemented. A simple video player that would work with the, already developed, backend and serve mostly as a familiarization and testing tool.

This "proto-app" consisted of a basic activity that would play the clip, using the requests already implemented, when clicking a button. The video player itself uses ExoPlayer, "an application-level media player for Android" and an open-source project that provides an alternative to Android's MediaPlayer API [Goob]. The reason for this choice was due to ExoPlayer's support of features like DASH and Smooth Streaming, as well as its easy customization and extendability.

Programatically speaking, an ExoPlayer object receives a MediaSource, that can be either video or audio, stored locally or from the web. This feature is important in the sense that, in the end-result, the user should be able to play video from the cloud storage and the device's local storage.

A button was also created to download a given clip to the local storage, thus verifying another important functionality.

4.7 Cloud Environment

After developing the database and the API for the server, the next step was using Cloud features. The decided tool was OpenStack. An instance was created in WIT-Software's servers and accessed through Putty [Tat19]. The instance was running CentOS 7 [cen19]. In the setup, there were, essentially, two steps: Importing the database and running the API.

For the database, it was necessary to, first, install PostgreSQL 11 in the OpenStack instance for the DB to work. Since PostgreSQL allows a full database to be exported and imported as a single file, the subsequent step was simplified. Afterward, changing the data source in the API was enough, and the new, and final, database was set.

To have the server running in the instance this was the procedure: first, build the project as a .jar file, then move the project into the OpenStack instance and finally run the generated .jar as a service. These three steps with double-checks in between then completed the setup of the cloud environment.

4.8 V-Auto App

V-Auto is a mobile application developed by WIT-Software that works in conjunction with a device that is connected to the user's vehicle. The app allows the user to track his/her trips and gives detailed information on them. It also gives a driving score for each trip and the user himself by using the data from all the trips. This score is obtained from several driving events that the device placed in the vehicle's OBD-II port detects. Other features include a vehicle tracker for when the car is parked and can give directions to it and Auto SOS which, in case of a serious accident in Portugal, contacts the user in case assistance is required [Vod].

After the setup process, the app consists of a tabbed screen with five tabs: State, Trips, Score, Auto SOS, and Settings. The state tab is a map with a marker placed where the vehicle is at that moment along with a Google Streetview image of that location (if possible) and a button to get directions from the current location to the vehicles. The trips tab has a feed of all the completed trips, with a map, time and location of the start and endpoints of the trip, the trip score, and the number of driving events that happened in it. The score tab has an overall driving score calculated from all the user's trips and more information on how it is obtained and some detailed information.

After setting up the cloud environment it was time to move into the integration with the existing application, V-Auto. And in order to achieve what was intended it was necessary to understand how it operates and its architecture.

V-Auto follows the MVP(Model-View-Presenter) Architectural Pattern which is a derivation of the MVC(Model-View-Controller) Pattern that was previously used in the backend and web interface implementation.

In MVP, the model contains the data, state, and business logic and its responsibilities include using API's, caching data and managing databases, among others. The View is, usually, implemented by an Activity and contains a reference to the Presenter, which acts as a middle man

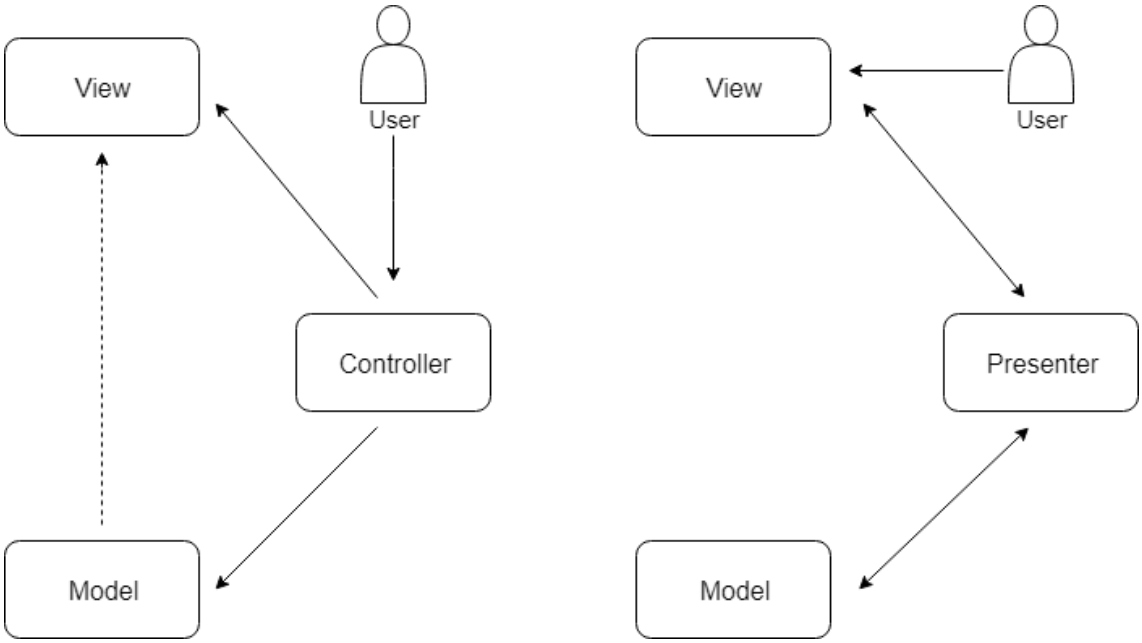


Figure 4.4: Diagram of the MVC(left) and MVP(right) design patterns

between the other two, retrieving data from the model and returning it to the View ready to be displayed. In MVC, the model has the same function, while the Controller is responsible for determining which View to display in response to an action. In MVC every action in the View correlates to a call to a Controller along with an action and in MVP actions route through the View to the Presenter. MVP also differs from MVC in the sense that it divides the application into modular, single-purpose components

V-Auto uses several libraries and frameworks, thus it's necessary to understand them to understand how to work with what already exists in the app and to implement new features with a consistent method.

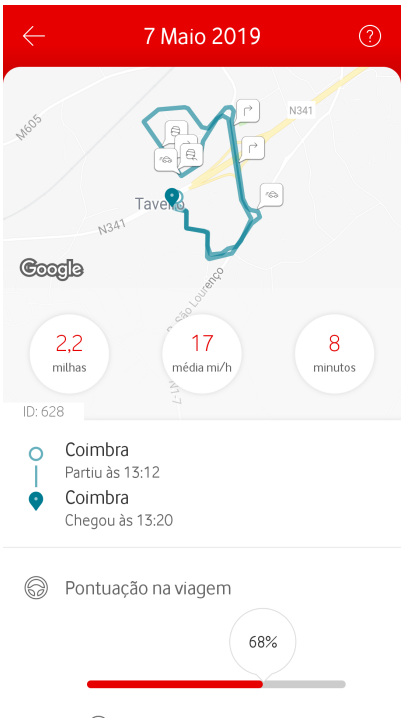
Most of the code in the files that were handled was in Java except for a few in Kotlin. Some of the frameworks that are used in the app and showed themselves useful in the development were: Dagger which is a dependency injector for Android and Java [Gooa], RxJava which is a library for composing asynchronous and event-based programs using observable sequences for the JVM [rxj], and Butterknife which is used to bing fields and methods for Android views which uses annotation processing to generate code that can be reused in many parts of the project [Wha].

After understanding and exploring the existing app and its state, the integration of the API into new features begun.

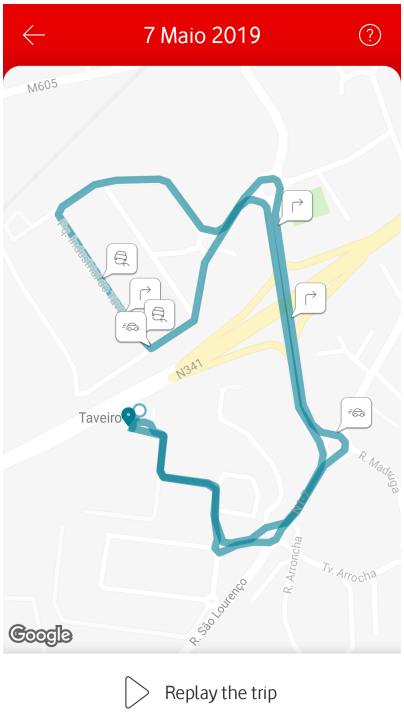
The features that were to be added to the app were in the map part of each trip, more specifically in two parts: when an event was clicked, and the second in the trip replay.

There were, essentially, two features added to the app. One for each event, and another for the replay of an entire trip.

The first alteration was in the Full Trip Detail screen, which occurs when the User clicks on the

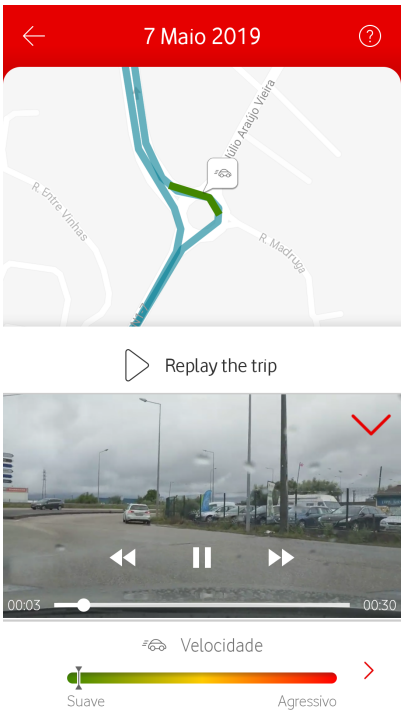


(a) TripDeatil Screen

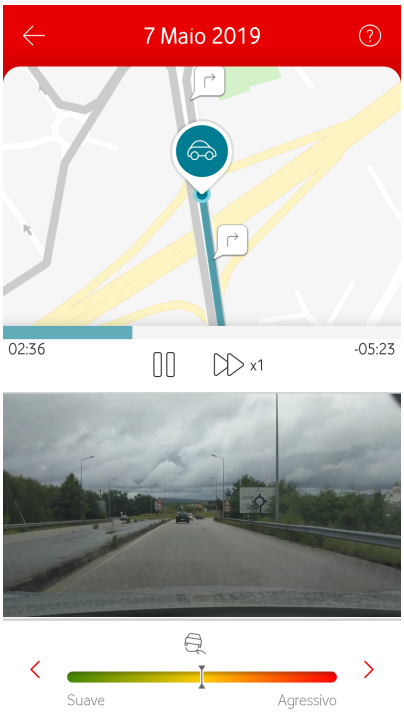


(b) TripFullDeatil Screen

Figure 4.5: Screen Caps of Unaltered Screens



(a) Details of an Event



(b) TripReplay Screen

Figure 4.6: Screen Caps of Modified Screens

map after selecting a specific trip from the Trip Feed. This screen consists of a map with a polyline, that represents the selected trip, and a button on the bottom to replay the trip by going through the polyline. If one or more events occurred in this trip they are represented in the polyline through markers, one for each event, the markers have an image representative of the type of event, and by clicking one of them a popup shows up at the bottom of the screen with the type and severity of the clicked event.

In order to view the video equivalent to each event, a video player was added to this popup, so that when an event marker is clicked the user is also presented with the clip that contains that event. After the addition of the video player the map became quite small after clicking in an event marker, but considering that the video is the central aspect of this screen and the map serves only to guide the user in which of the trip the event occurred it isn't a hassle. In the top right corner of the video player, there is an icon that downloads the video file into the local memory, and in the future, if the same clip is played it's done so by playing the local file instead of streaming from the cloud.

The second alteration was in the trip replay. When clicking the 'Replay Trip' button at the bottom of the map screen of a trip, a marker would follow the polyline accordingly. In the same way a video player pops up when clicking an event, it also pops up in this situation and plays the entire trip. Since video is stored in small clips, a trip isn't just one big video file and is instead a playlist of all the clips that belong to that trip ordered by timestamp and then played consecutively.

4.9 Summary

This chapter serves as a guide of the whole development process, by describing how the previously defined solution was reached. It specifies each step and goes into detail on what decisions were made and why, as well as mentioning the technologies used and what part they played in the development itself or the final product.

Chapter 5

Conclusions

5.1 Overview

This final chapter describes the conclusion of the entire dissertation. It goes over what the attained results were, as well as real-world applications of the developed platform and some future work that could improve the platform and perhaps extend its functionality to a new, previously unthought, setting.

5.2 Results

After analyzing the state of the art in the market and obtaining information regarding software features and requirements along with gathering information regarding relevant technologies and conjugating all that information with the initial objectives it was possible to trace what the expected results of the dissertation were. Defining the solution and a work plan helped tremendously in all stages of development. Weekly checkpoints and meetings were also vital in staying coordinated with the host company in developing something that would interest both parties.

Since the solution defined before the development process itself wasn't strict in every aspect, not everything about how the platform looked and worked was defined. What was, however, ended up becoming what was initially intended. For example, before the use of the cloud, several options were considered, and the protocols to implement would depend largely on what cloud service was chosen. At first a public provider was thought of, like AWS or Microsoft Azure, but in the end, a private option, OpenStack instance, was used.

As the first step of implementation, the backend laid the ground-work for all the features that ended up on the Android app, as well as some that didn't but were defined in the initial functional requirements, like the login or the camera and user handling.

In regards to the mobile application: at first, it was thought that a listing of all the videos would be the best option, however with the evolution of the platform and familiarization with the app, a pop-up in the trip screen became the new objective, as it would offer a more straight-forward and intuitive experience as well as preserving the design language of the app.

By going over the functional and non-functional requirements, and the use cases set up by them, defined in the solution specification, it's possible to somewhat measure and evaluate the results of the developed work.

All the non-functional requirements were implemented. In regards to the functional requirements, the exceptions are described in the revision. The combination of these two types of requirements resulted in a system that, by being developed with the use cases in mind, achieves what was desired and described.

5.3 Future Work

Despite all the work done so far, many places could be improved and even expanded, upon.

The improvements could be focused on performance aspects. The app is responsive, and the streaming is fluid, however, it could still be improved and maybe give the user more information and control over the entire process, both when streaming and downloading the video clips.

When considering expansion, the first thing that comes to mind is the camera connection. When defining the requirements this feature was seen as a 'Nice-to-have' and ended up not being implemented. The API is prepared to handle video upload and does it through the web app, but having the camera do it automatically is the goal. Having a live-view feature is also a possible next step.

Going a bit further away from the app but still in its range is the improvement of the web app. It could cease to be a development tool, and become a full-fledged web app that could be used in conjunction with the app. There could be two sides to this new platform: it could be a companion to the app, giving users access to their information, trips and clips in an organized manner; or it could serve as a management tool by higher-level users. For instance, in a fleet environment, where an owner can check in on his drivers and review footage of trips, the owner would be a higher-level user than the drivers and could access their information and manage their accounts.

5.4 Summary

By analyzing the end product, and correlate it to real-world use it's possible to have a better understanding of the platform itself and the environment in which it conforms itself. By going over possible improvements and the future of the platform it's possible to understand that, despite all the work done and features implemented, a lot more can be achieved.

References

[Ado14] Adobe. REAL-TIME MESSAGING PROTOCOL (RTMP) SPECIFICATION. https://www.adobe.com/devnet/rtmp.html, 2014.

[AIM10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 2010.

[Ama] Amazon. Live Streaming on AWS. https://aws.amazon.com/answers/media-entertainment/live-streaming/.

[ANR74] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete Cosine Transform. *IEEE Transactions on Computers*, 1974.

[App15] Apple. HTTP Live Streaming. https://developer.apple.com/streaming/, 2015.

[av1] Bitmovin & AV1. https://bitmovin.com/av1/.

[blaa] BlackVue App. https://play.google.com/store/apps/details?id=comb.blackvuec&hl=en.

[blab] BlackVue DR750S. https://www.blackvue.com/dr750s-2ch/.

[Bla19] BlackBoxMyCar. BlackBoxMyCar - What is a Dash Cam?, 2019.

[BTB+18] Abdelhak Bentaleb, Bayan Taani, Ali C. Begen, Christian Timmerer, and Roger Zimmermann. A Survey on Bitrate Adaptation Schemes for Streaming Media over HTTP, 2018.

[CCC+14] Shao Chi Chen, Hsin Yi Chen, Yi Ling Chen, Hsin Mu Tsai, and Bing Yu Chen. Making in-Front-of Cars Transparent: Sharing First-Person-Views via Dashcam. *Computer Graphics Forum*, 2014.

[CCL+15] Hsin I. Chen, Yi Ling Chen, Wei Tse Lee, Fan Wang, and Bing Yu Chen. Integrating dashcam views through inter-video mapping. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[cen19] The CentOS Project, 2019.

[Cou] Tom Coughlin. Moving Data To The Cloud. https://www.forbes.com/sites/tomcoughlin/2017/07/19/moving-data-to-the-cloud/#17faa292567c.

[EPS99] Paul England, Marcus Peinado, and Mukund Sankaranarayan. Secure video card in computing device having digital rights management (DRM) system, 1999.

[Fel18] Christian Feldman. Multi-Codec DASH Dataset: An Evaluation of AV1, AVC, HEVC and VP9, 2018.

[gara] Garmin Dash Cam 55. https://buy.garmin.com/pt-PT/ES/p/567282.html.

[garb] Garmin VIRB App.

[GMM+13] Dan Grois, Detlev Marpe, Amit Mulayoff, Benaya Itzhaky, and Ofer Hadar. Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders. In *2013 Picture Coding Symposium, PCS 2013 - Proceedings*, 2013.

[Gooa] Google. Dagger.

[Goob] Google. ExoPlayer.

[GS05] David Groth and Toby Skandier. Network Fundamentals. In *Network+ Study Guide*. 2005.

[HL16] Anthony Ho and Shunjun Li. *Handbook of Digital Forensics of Multimedia Data and Devices*. 2016.

[Inc] Incapsula. What is a CDN. https://www.incapsula.com/cdn-guide/what-is-cdn-how-it-works.html.

[Kel29] R.D. Kell. Improvements relating to electric picture transmission systems, 1929.

[KLY+17] Minho Kim, Jaemin Lim, Hyunwoo Yu, Kiyeon Kim, Younghoon Kim, and Suk-Bok Lee. ViewMap: Sharing Private In-Vehicle Dashcam Videos. In *NSDI2017*, 2017.

[Kor] Julia Kornich. Overview of Live Streaming using Azure Media Services. https://docs.microsoft.com/en-us/azure/media-services/previous/media-services-manage-channels-overview.

[Lau18] Jeremy Laukkonen. Lifewire - What is a Dashcam?, 2018.

[Liu06] Chin-Chun Liu. Micro secure digital memory card, 2006.

[MBG+13] Debargha Mukherjee, Jim Bankoski, Adrian Grange, Jingning Han, John Koleszar, Paul Wilkins, Yaowu Xu, and Ronald Bultje. The latest open-source video codec VP9 - An overview and preliminary results. In *2013 Picture Coding Symposium, PCS 2013 - Proceedings*, 2013.

[Mic15] Microsoft. Smooth Streaming. https://www.iis.net/downloads/microsoft/smooth-streaming, 2015.

[owla] Owl Car Cam. https://owlcam.com.

[owlb] Owl Car Cam App. https://play.google.com/store/apps/details?id=us.owl.owlapp&hl=en_US.

[Piva] Pivotal Software. Spring Boot. http://spring.io/projects/spring-boot.

[Pivb] Pivotal Software. Spring Security.

[PKML16] Sangkeun Park, Joohyun Kim, Rabeb Mizouni, and Uichin Lee. Motives and Concerns of Dashcam Video Sharing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, 2016.

REFERENCES

[PM99] Niels Provos and David Mazières. A future-adaptive password scheme. In *ATEC '99 Proceedings of the annual conference on USENIX Annual Technical Conference*, 1999.

[rex] Rexing V1 Gen 3. https://www.rexingusa.com/products/rexing-v1-gen-3/?from=products.

[roaa] Roav App. https://play.google.com/store/apps/details?id=com.zhixin.roav.cam&hl=en_US.

[roab] Roav DashCam C1. https://goroav.com/products/roav-dashcam.

[rxj] RxJava.

[SCFJ03] M Schulzrinne, S Casner, R Frederick, and V Jacobson. RTP: A Transport Protocol for Real-Time Applications. https://www.ietf.org/rfc/rfc3550.txt, 2003.

[Sik97] Thomas Sikora. The MPEG-4 video standard verification model. *IEEE Transactions on Circuits and Systems for Video Technology*, 1997.

[Smi18] Luke John Smith. Driver WARNING - Your dash cam could land you up to £9,000 fine and see you JAILED abroad, 2018.

[SOHW12] Gary J. Sullivan, Jens Rainer Ohm, Woo Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 2012.

[Tat19] Simon Tatham. PuTTY: a free SSH and Telnet client, 2019.

[Tel08] Telestream. Extracting and Preparing Metadata to Make Video Files Searchable. Technical report, 2008.

[Tex15] Laci Texter. What is Video Metadata and How Do I Use It? https://sproutvideo.com/blog/what-is-video-metadata-and-how-do-i-use-it.html, 2015.

[Thea] The PostgreSQL Global Development Group. PostgreSQL. https://www.postgresql.org/docs/.

[Theb] The PostgreSQL Global Development Group. PostgreSQL: Documentation. Chapter 35. Large Objects.

[Thec] The PostgreSQL Global Development Group. The PostgreSQL JDBC Interface, Chapter 7. Storing Binary Data.

[thi] Thinkware Cloud App. https://play.google.com/store/apps/details?id=com.thinkwaresys.thinkwarecl

[tim] TimaCam App. https://play.google.com/store/apps/details?id=com.tima.dr.novatek.bs.en&hl=pt_PT.

[Vod] Vodafone. V-Auto by Vodafone. https://play.google.com/store/apps/details?id=com.vodafone.auto&h

[Wal92] Gregory K. Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 1992.

[Wha] Jack Wharton. ButterKnife.

[Wi-] Wi-Fi Alliance. Wi-Fi Direct. https://www.wi-fi.org/discover-wi-fi/wi-fi-direct.

REFERENCES

[WPG+10] Jiyi Wu, Lingdi Ping, Xiaoping Ge, Wang Ya, and Jianqing Fu. Cloud storage as the infrastructure of Cloud Computing. In *Proceedings - 2010 International Conference on Intelligent Computing and Cognitive Informatics, ICICCI 2010*, 2010.

[WS19] WIT-Software. WIT-Software. https://www.wit-software.com, 2019.

[WSBL03] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 2003.

[WvK01] S ; Winkler, C J; van den Branden Lambrecht, and M. ; Kunt. *Vision and Video: Models and Applications*. 2001.

[yia] YI Dash Cam App. https://play.google.com/store/apps/details?id=com.xiaoyi.car.camera.international&hl=

[yic] YI Smart Dash Camera. https://www.yitechnology.com/yi-smart-dash-camera.