Mestrado em Engenharia Informática

Estágio – 2014/2015

Relatório Final

Aplicação iOS para partilha, edição e controlo de UGC na TV

Tiago Miguel Alves Antunes

tantunes@student.dei.uc.pt

Orientador WIT-Software:

Sérgio Miguel Felício Dinis

Orientador Universidade:

Luis Miguel M. L. Macedo

Data: 2 de Setembro de 2015

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Mestrado em Engenharia Informática

Estágio – 2014/2015

Relatório Final

Aplicação iOS para partilha, edição e controlo de UGC na TV

Tiago Miguel Alves Antunes

tantunes@student.dei.uc.pt

Júri Arguente:

João Pedro Morais de Matos Moniz Ramos

Júri Vogal:

Paulo Alexandre Ferreira Simões

Data: 2 de Setembro de 2015

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Esta página foi deixada em branco intencionalmente.

Abstract

New generations of consumers have a growing disinterest in current TV model, without much change in the paradigm since its inception. Recent years have seen a growing number of mobile devices, smartphones and tablets that already are essential objects of day-to-day life of many users. Operators seek alternatives that might inhibit the breakdown of interest in traditional television. The user-generated content (UGC) has gained great popularity, especially among the younger age groups.

This internship has as its primary objective the analysis of this new paradigm of consumption, in order to understand trends, popularity patterns, and the differentiating points of the many existing UGC platforms in the market.

The result of this study aims to develop differentiated TV solutions, focusing this new paradigm of consumption. The aim is to give tools to the traditional operators to provide your service, content that has diverted the attention of customers to other screens.

Keywords

Apple Watch, GoPro, iOS, Live video share, Television, Second Screen, Streaming devices, User generated content, Video effects

Resumo

As novas gerações de consumidores apresentam um crescente desinteresse pelo atual modelo televisivo, sem grande alteração no paradigma desde a sua criação. Nos últimos anos assistimos a um número crescente de dispositivos móveis, *smartphones* e *tablets* que já são hoje objetos essenciais do dia-a-dia de muitos utilizadores. As operadoras procuram alternativas que possam travar a quebra de interesse na televisão tradicional. O conteúdo gerado pelo utilizador (UGC) tem vindo a ganhar grande popularidade, principalmente junto das faixas etárias mais jovens.

Este estágio tem como primeiro objetivo a análise deste novo paradigma de consumo, de forma a perceber as tendências, os padrões de popularidade, e os pontos diferenciadores das muitas plataformas de UGC existentes no mercado.

O resultado deste estudo visa o desenvolvimento de soluções televisivas diferenciadoras, tendo em foco este novo paradigma de consumo. Pretende-se assim dar aos operadores tradicionais ferramentas para fornecer, no seu serviço, o conteúdo que lhes tem desviado as atenções para outros ecrãs.

Palavras-Chave

Apple Watch, Conteúdo gerado pelo utilizador, Efeitos de vídeo, Partilha de vídeo em direto, GoPro, iOS, Televisão, Second Screen, Streaming devices

Esta página foi deixada em branco intencionalmente.

Agradecimentos

O presente relatório representa o culminar de um período de estudos em Engenharia Informática que não teria sido possível sem todo o suporte oferecido pelos meus pais. Apesar das adversidades que a vida nos reserva, nunca me deixaram desistir, levando-me sempre a acreditar que com trabalho e dedicação tudo é possível de alcançar. Um especial e eterno obrigado.

À minha namorada, por toda a paciência, por todo o suporte, por ter estado sempre ao meu lado ao longo deste percurso, um grande obrigado.

Aos meus orientadores, Sérgio Dinis e Luís Macedo, agradeço por toda a disponibilidade e suporte prestado durante todas as fases deste estágio. Agradeço também ao gestor de projeto Nuno Carvalho pela confiança depositada em mim ao longo dos diferentes desenvolvimentos efetuados.

Aos meus colegas de equipa, um forte obrigado pelo companheirismo, conselhos e suporte prestado ao longo do estágio. Um especial obrigado ao membro da equipa Filipe Brunido por toda a paciência e suporte prestado ao longo deste percurso, permitindo-me alargar conhecimentos em diferentes áreas.

Para finalizar, gostaria ainda de agradecer à empresa que me acolheu durante estes dez meses de projeto, oferecendo-me a oportunidade integração com uma fantástica equipa de trabalho, permitindo o desenvolvimento de novas capacidades.

Esta página foi deixada em branco intencionalmente.

Índice

Lista de Figuras

Lista de Tabelas

Esta página foi deixada em branco intencionalmente.

Glossário

Back-end	Parte da aplicação relativa ao servidor onde se encontra a camada lógica do negócio invocada pelo *front-end*.
Branch	Persistência de eventuais alterações efetuadas no repositório.
Commit	Persistência de alterações efetuadas num dado repositório.
Demuxing	Processo em que fluxos em um arquivo de vídeo ou áudio são separados em partes diferentes.
Front-end	Interface gráfica do sistema direcionada para o utilizador final com o qual este interage diretamente.
Memory Leak	Fenómeno que ocorre quando é alocada memória para uma determinada operação, mas não é libertada quando não é mais necessária.
Merge	Integração de várias alterações num mesmo ficheiro.
Middleware	Agente mediador entre o *software* e as demais aplicações visando fornecer formas de comunicações entre diferentes componentes.
Muxing	Processo onde partes separadas do vídeo ou áudio são unidos em um único arquivo.
Proxy	Servidor intermediário responsável por reencaminhar os *requests* de serviços efetuadas por um cliente a um determinado servidor.
Remuxing	Recombinação dos ficheiros de vídeo ou áudio via *muxing*.
Set-top Box	Dispositivo que se conecta a um televisor e que é responsável por descodificar um sinal digital mostrando o resultado sob a forma de imagem e som no televisor.
Thread	Fluxo de controlo sequencial isolado dentro de um processo.
Trunk	"Linha principal" de desenvolvimento num repositório SVN.
Waterfall	Modelo de desenvolvimento de *software* sequencial e estruturado.

Acrónimos

ADK	Applications Development Kit
API	Application programming interface
DIAL	Discovery and Launch
DLNA	The Digital Living Network Allience
GCD	Grand Central Dispatch
HDMI	High-Definition Multimedia Interface
HLS	Http Live Streaming
HTTP	Hypertext Transfer Protocol
IMDB	Internet Movie Database
JNI	Java Native Interface
LGPL	Lesser General Public License
LLVM	Low Level Virtual Machine
MVC	Modelo Vista Controlo
OTT	Over-the-top content
R&D	Research and development
RTSP	Real Time Streaming Protocol
RUP	Rational Unified Process
SDK	Software Development Kit
SDK	Software Development Kit
SSDP	Simple Service Discovery Protocol
STB	Set-top Box
TCP	Transmission Control Protocol
TDLS	Tunneled Direct Link Setup
UDP	User Datagram Protocol
UGC	User generated content
UI	User Interface
UPNP	Universal Plug and Play
UX	User experience
VOD	Video on demand

WOL Wake-on-Lan

XML eXtensible Markup Language

Esta página foi deixada em branco intencionalmente.

Capítulo 1
Introdução

O presente projeto enquadra-se no âmbito da disciplina de estágio do Mestrado em engenharia informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra. Foi inserido no contexto da empresa WIT Software, sob a orientação do professor Doutor Luís Miguel M. L. Macedo e com orientador na empresa o Engenheiro Sérgio Miguel Felício Dinis.

Este capítulo introdutório está dividido em quatro secções. A primeira faz apresentação da instituição que acolhe o estagiário durante o período de 10 meses. A segunda apresenta o enquadramento do estágio. Seguidamente, a terceira secção aborda as motivações, o problema atualmente existente no modelo televisivo e a sua perda de utilizadores para outras plataformas. Por último, e não menos importante, é apresentada a estrutura do documento, com uma breve descrição para cada capítulo.

1.1. Instituição

A WIT Software é uma empresa especializada no desenvolvimento de aplicações e serviços para operadores móveis e empresas de telecomunicações. Criada em 2001, conta com uma carteira de clientes em mais de 15 países, incluindo alguns dos maiores operadores mundiais, como o Grupo Vodafone, Unitel, Telefónica, Orange, Deutsche Telekom e TeliaSonera.

A empresa define três centros de competência: Telco, que desenvolve *software* de ponta para operadores móveis; Mobile, que tem desenvolvido soluções móveis avançadas para a banca, sector energético e utilidades, grupos de media e operadores móveis. Por sua vez, a unidade TV desenvolve soluções de *software* para operadores de TV por cabo e IPTV.

A WIT estabeleceu uma relação forte e duradoura com o mundo académico. Na última década, a empresa tem investido, todos os anos, 25% dos seus recursos em atividades de R&D. Trata-se de um investimento contínuo, que é determinante para a estratégia da empresa de diferenciação pela inovação.

1.2. Enquadramento do estágio

Durante o estágio estive integrado na unidade dedicada à televisão da WIT-Software. Esta teve por detrás da sua criação o intuito de inovar e dinamizar o setor de televisão, implementando soluções televisivas que tornam a sua experiência mais completa, com forte fator diferenciador.

É assim responsável por diversos produtos e serviços que se complementam à oferta realizada pelos operadores de IPTV. Recentemente foi desenvolvida nesta unidade a solução See What I See, introduzida no mercado pela Vodafone Portugal com o nome de LiveOnTv, tendo sido amplamente divulgada em diversas campanhas de *marketing* na televisão pública portuguesa.

O presente estágio inseriu-se no desenvolvimento de soluções televisivas diferenciadoras, tendo como especial foco um paradigma de consumo, o conteúdo gerado pelo utilizador (UGC). O UGC foi identificado pela WIT como um possível valioso ponto de convergência

para as distintas formas de consumo acuais. Procurando assim uma solução única no mercado que se torne numa mais-valia para os operadores na sua oferta de TV.

Os desenvolvimentos tiveram por base a plataforma móvel iOS da Apple, culminando na criação de diferentes provas de conceito, que visam alcançar o leque de utilizadores abrangido por este novo paradigma de consumo.

1.3. Motivação

Atualmente, o aumento do número de dispositivos móveis e as potencialidades que estes criam têm retirado algum interesse ao atual modelo televisivo. O paradigma de consumo tem-se alterado, retirando olhos do ecrã da TV, de um modelo de consumo ao vivo para outros ecrãs e para modelos "*on-demand*". O consumidor atual já não se conforma em ver o que lhe é imposto numa grelha de emissão, procura antes o conteúdo que lhe interessa para consumir quando e onde bem entender.

No seguinte gráfico apresentado pela figura 1, resultado da análise efetuada pela empresa *Bill estimates*, podemos visualizar o aumento exponencial que nos últimos anos o mercado de dispositivos móveis tem assistido.

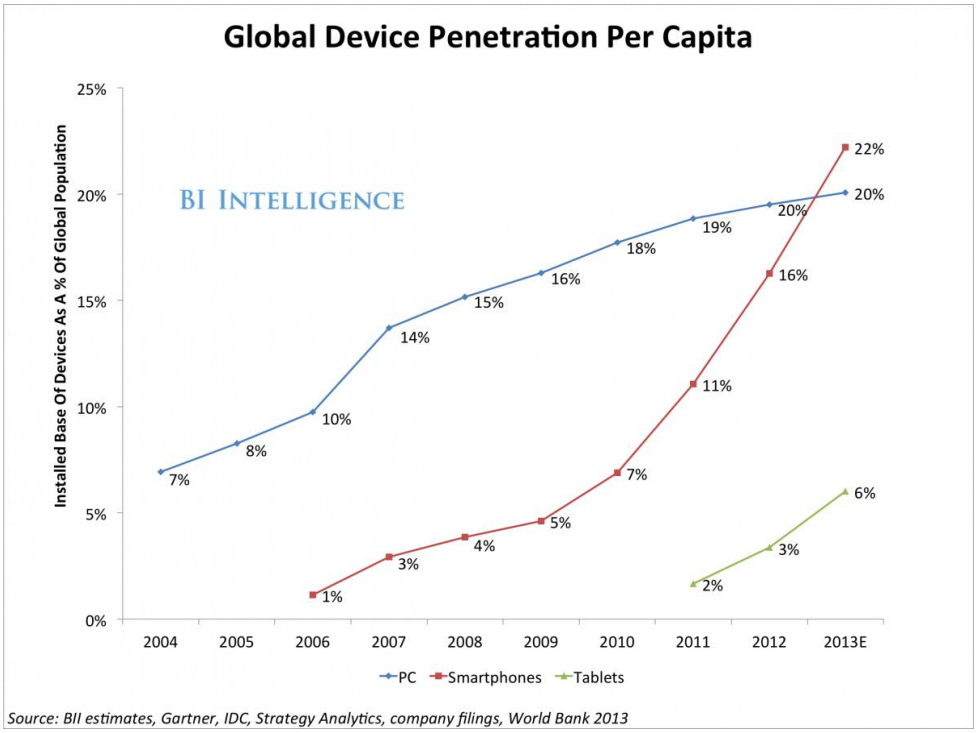

Figura 1 – Penetração de dispositivos a nível mundial no período compreendido entre os anos 2004 e 2013 [1]

A interatividade que certas plataformas de conteúdo gerado pelo utilizador fornecem vão de encontro aos interesses das novas gerações de utilizadores, onde estas sentem que têm liberdade de escolha, interatividade, e até possibilidade de fazer parte desse mesmo conteúdo. A falta de alternativas no tradicional modelo televisivo faz com que cada vez mais novas gerações percam o seu interesse na televisão.

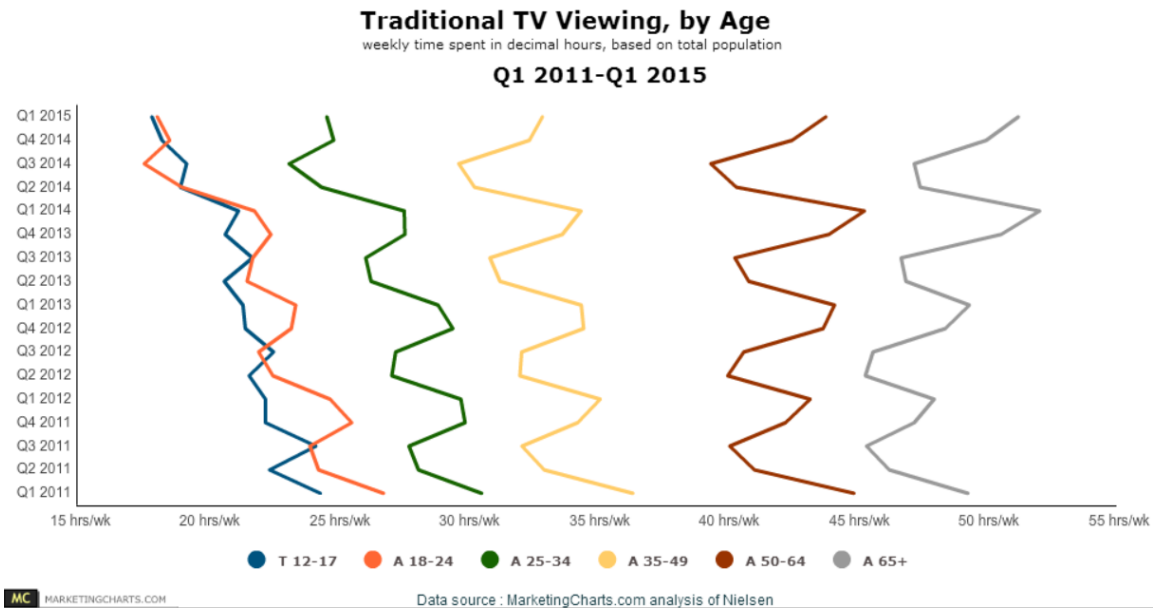

Figura 2 – Tempo despendido a ver televisão dividido em faixas etárias durante o período de 2011 a 2014 [2]

O gráfico exibido na figura 2, resultado do estudo efetuado pela empresa *Nielsen,* apresenta os níveis de consumo televisivo junto da população americana, dividido por faixas etárias, desde o primeiro trimestre de 2011 ao primeiro trimestre de 2015. Podemos verificar que as faixas etárias mais idosas são fiéis ao tradicional modelo televisivo, não apresentado sinais de decréscimo de interesse durante o período analisado. Já as faixas etárias mais jovens no período analisado revelam estar a perder o interesse neste modelo, tendo as faixas etárias dos 12 aos 17 anos e dos 18 aos 24 anos apresentado o maior decréscimo. A proposta de estágio vai de encontro com estes factos, visando o desenvolvimento de soluções que possam inverter esta tendência.

1.4. Estrutura do documento

Este documento segue a seguinte estrutura:

- **Capítulo 2 (Apresentação do estágio)** – apresentação dos objetivos do estágio, riscos inerentes, equipa de trabalho, planeamento inicial, desvios verificados e resultados finais alcançados.
- **Capítulo 3 (Gestão do projeto)** – especificação da metodologia, processos, gestão de requisitos e gestão riscos associados.
- **Capítulo 4 (Estado da arte)** – análise das tecnologias e soluções existentes no mercado dentro do âmbito dos projetos desenvolvidos.
- **Capítulo 5 (Levantamento de requisitos)** - especificação de requisitos das aplicações desenvolvidas
- **Capítulo 6 (Desenho das aplicações)** – especificação das arquiteturas nas aplicações desenvolvidas.
- **Capítulo 7 (Desenvolvimento)** – descrição do trabalho desenvolvido, justificando decisões de desenvolvimento tomadas ao longo dos projetos.
- **Capítulo 8 (Controlo de qualidade)** – descrição dos processos utilizados para controlo de qualidade final das aplicações desenvolvidas.
- **Capítulo 9 (Conclusões)** – reflecção final ao trabalho desenvolvido, tendo como focos o trabalho realizado e resultados alcançados.

Os diferentes capítulos descritos neste relatório são complementados por vários anexos, contendo maior detalhe para as matérias abordadas. Estes seguem a seguinte estrutura:

- **Anexo A (Estado da arte)** – contém o estudo completo do estado da arte;
- **Anexo B (Requisitos)** – apresenta a lista completa de requisitos bem como o seu nível de prioridade;
- **Anexo C (Arquitetura)** – contém um estudo das diferentes *Application Programming Interfaces* (APIs) utilizadas e a definição da arquitetura para as aplicações desenvolvidas;
- **Anexo D (UI/UX)** – apresenta a definição final a nível de *user interface* (UI) e *user experiecie* (UX);
- **Anexo E (Testes de software)** – apresenta o resumo dos diferentes testes de software realizados às funcionalidades implementadas;
- **Anexo F (Gestão de riscos)** – apresenta o resumo dos principais riscos verificados e ações de mitigação durante o período de estágio;
- **Anexo G (Informação complementar)** – reúne informação abordada durante o estágio curricular, complementar à proposta principal de estágio.

Capítulo 2
Apresentação do estágio

Neste capítulo são apresentados o conjunto de objetivos pretendidos com o desenvolvimento do estágio, análise de riscos, constituição da equipa de trabalho, planeamento de trabalhos, desvios verificados, terminando com resultados alcançados.

2.1. Proposta e objetivos

Tirando partido do enorme crescimento de dispositivos móveis a que os utilizadores hoje em dia têm acesso pode-se criar novas formas de interatividade. Pode-se igualmente tirar partido do seu forte potencial para a criação de UGC e simplificar a sua partilha com outros dispositivos. Pode-se também enriquecer o do valor final deste conteúdo através alterações ao conteúdo original.

A proposta de estágio vai de encontro a estes pressupostos, tendo como objetivos a compreensão dos modelos utilizados pelas principais plataformas de conteúdo gerado pelo utilizador. Esta análise permitirá identificar fatores diferenciadores que podem ser incorporados na aplicação a desenvolver. Esta aplicação móvel em iOS, para iPhone e iPad, deve permitir a partilha de conteúdo gerado pelo utilizador com a televisão.

Sintetizando, na fase inicial deste estágio, foi pretendido realizar o seguinte plano de trabalhos:

i. Análise de plataformas baseadas em UGC, estudando as principais plataformas no mercado e identificando as principais tecnologias;
ii. Identificação de tendências e fatores distintivos, baseadas na análise de tendências às diferentes plataformas;
iii. Criação de protótipos ou proposta de novas funcionalidades a incorporar em produtos desenvolvidos pela empresa, fundamentando devidamente as suas escolhas.

As novas funcionalidades propostas podem passar pelo desenho e implementação de uma aplicação de raiz. Ainda assim, atendendo à complexidade da temática, aliada à relativa curta duração do estágio e à curva de aprendizagem que foi necessária ultrapassar, foi dada preferência a propostas possíveis de integrar com produtos existentes na empresa.

Este conjunto de propostas teve por base a plataforma de desenvolvimento móvel iOS, pensadas para permitir a:

i. Partilha de conteúdo com a televisão;
ii. Edição de conteúdo partilhado;
iii. Controlo do conteúdo exibido na televisão.

O objetivo geral do projeto passou por complementar a oferta de soluções diferenciadoras da empresa na área da televisão, integrando, quando possível, com produtos já existentes.

2.2. Riscos

Um projeto de desenvolvimento de *software* está habitualmente sujeito a riscos, que, quando não identificados e minimizados atempadamente, podem conduzir um projeto ao fracasso.

Neste subcapítulo são identificados os principais riscos inerentes à proposta de estágio apresentada.

2.2.1. Curva de aprendizagem

A falta de conhecimentos na tecnologia utilizada num dado projeto representa sempre um risco no seu desenvolvimento, podendo conduzir a alterações no planeamento inicial, e, em casos mais graves, conduzir ao seu insucesso.

O facto de eu, aquando o inicio deste projeto, não ter qualquer contacto com a tecnologia utilizada representou um acréscimo no risco de possível incumprimento de prazos por falta de experiência. Este risco foi progressivamente minimizado com o decorrer do estágio devido ao estudo da linguagem e arquitetura dos dispositivos.

2.2.2. Indefinição do âmbito

Quando um projeto não está devidamente definido poderá levar ao incumprimento de prazos e objetivos. Para a realização deste projeto foi pretendido que apresentasse novas ideias, integradas no âmbito da proposta.

Esta indefinição inicial relativa ao plano de trabalho aumentou o risco de incumprimento de objetivos previamente estabelecidos.

2.2.3. Divergências e alterações de prioridade

Por vezes, as divergências entre as escolhas realizadas pelo estagiário e a visão proposta pela empresa, podem originar conflitos de interesse e afetar o percurso do projeto.

Ao longo do decorrer do projeto este risco verificou-se, tendo em conta ligeiras alterações aos requisitos e alterações pontuais para preparação de demonstrações a clientes.

Para minimizar o impacto deste risco na execução do projeto foi realizado um acompanhamento constante de modo a garantir uma sincronização entre as necessidades da empresa e objetivos do estágio curricular.

2.3. Equipa

O estágio inseriu-se na equipa de trabalho da empresa na área da televisão. Aqui a equipa está dividida em pessoas responsáveis pelo desenvolvimento e manutenção de produtos ou serviços. Dada a natureza dos objetivos pretendidos para o estágio curricular, eu estive inserido na equipa do produto. A constituição da equipa, aquando o início do projeto, pode ser visualizada na seguinte tabela 1.

Gestor de projeto	**Nuno Carvalho**
Líder de equipa	Sérgio Dinis
Produto	David Queirós
	Pedro Coelho
	Mário Amaral
	Tiago Mano
	Tiago Antunes
Serviços	Diogo Mestre
	Filipe Brunido
	André Cravo

Tabela 1 – Constituição da equipa de televisão na WIT-Software

2.4. Planeamento inicial

A proposta de estágio inicialmente apresentada visa a implementação de uma aplicação para iPhone e iPad que tenha a capacidade de enviar UGC do dispositivo móvel para a Televisão. A aplicação deverá ser suportada pelos seguintes dispositivos que se ligam à televisão: Apple TV, Google Chromecast, Amazon Fire, Smart TVs da LG e Samsung. Deve ainda suportar o envio de fotos, vídeos e músicas. O UGC pode estar guardado localmente no iPhone/iPad assim como alojado em serviços de cloud já existentes (ex: YouTube, Facebook, Instagram).

Após a análise do estado da arte e apresentação de novas ideias que visaram contextualizar a proposta inicial foram realizadas algumas provas de conceito com *streaming devices*, entre elas a criação de uma aplicação recorrendo à biblioteca ConnectSDK e criação de uma aplicação nativa para os dispositivos Chromecast da Google. Ambas as aplicações desenvolvidas permitem a partilha de conteúdo fotos, vídeo e música com a televisão. Estes desenvolvimentos efetuados permitiram identificar algumas limitações das plataformas e protocolos de *streaming devices* atualmente existentes, não estando alinhadas com os objetivos pretendidos para novas soluções pelas operadoras.

Tendo sido identificados diversos aspetos diferenciadores nas câmaras de ação GoPro, durante a análise de soluções e plataformas de UGC, a empresa WIT-Software decidiu iniciar uma prova de conceito de uma nova funcionalidade no produto "See What I See". Esta nova funcionalidade desenvolvida consiste na possibilidade de obtenção de conteúdo em direto em

modo vídeo através câmaras GoPro. Dada a sua relação com temática do projeto, a empresa decidiu inseri-la no âmbito do estágio curricular em curso.

Foi então definido o seguinte planeamento para o estágio, apresentado na tabela 2.

Setembro	**Análise do estado da arte**
Outubro	Proposta de novas ideias e análise de requisitos
	Introdução à plataforma móvel iOS e ambiente de desenvolvimento Xcode
Novembro	Prototipagem rápida plataforma ConnectSDK
	Prototipagem rápida de aplicações para Chromecast
	Prototipagem rápida com bibliotecas de efeitos de vídeo
Dezembro	Estudo de comunicação com câmaras GoPro
	Introdução do projeto "See What I See"
	Integração das câmaras GoPro com a aplicação "See What I See"
Fevereiro	Suporte a efeitos de vídeo na aplicação "See What I See"
Março	Suporte a streaming devices na aplicação "See What I See"
Maio	Comunicação bi-direcional com set-top boxes na aplicação "See What I See"

Tabela 2 – Planeamento inicial de trabalhos

2.5. Desvios

Após o desenvolvimento da prova de conceito da aplicação See What I See com suporte às câmaras GoPro foram realizadas demonstrações a potenciais clientes. Daqui resultou o interesse da Vodafone Portugal na incorporação desta nova funcionalidade no produto "Live on TV", nome comercial utilizado pela Vodafone Portugal da solução da WIT-Software.

Os desenvolvimentos efetuados até ao momento estavam restritos aos modelos das câmaras GoPro 3. O interesse da Vodafone Portugal culminou num pedido de alargamento de compatibilidade ao novo leque de modelos das GoPro 4.

Como fui o elemento na empresa responsável pelo desenvolvimento desta solução, fiquei igualmente responsável pelos seus novos desenvolvimentos. Esta adição de trabalhos não programada ao planeamento inicial obrigou a uma reavaliação de prioridades das tarefas. Com a falta de interesse por parte de potenciais clientes em soluções que fizessem uso de *streaming devices*, e com o impacto temporal causado pela adição do suporte aos modelos GoPro 4, a empresa decidiu remover do plano de trabalhos o suporte a *streaming devices* na aplicação "See What I See".

Após o desenvolvimento do módulo responsável pela adição de efeitos de vídeo em direto na aplicação "See What I See" existiu novamente a oportunidade e inserir o meu estágio no contexto de uma solução comercial. Dado o recente lançamento do produto Apple Watch, a empresa decidiu iniciar o desenvolvimento de uma solução que fizesse uso deste novo relógio. Foi pretendido que esta solução pudesse controlar set-top boxes e exibir conteúdo acerca de programas em reprodução e videoclube. Dada a igualdade da plataforma de desenvolvimento e similaridade com os desenvolvimentos inicialmente programados, comunicação com set-top boxes, a empresa decidiu inserir esta proposta no âmbito do meu estágio.

Seguidamente podemos visualizar na tabela 3, o plano de trabalhos final, atualizado com as diferentes reavaliações de tarefas.

Setembro	**Análise do estado da arte**
Outubro	Proposta de novas ideias e análise de requisitos
	Introdução à plataforma móvel iOS e ambiente de desenvolvimento Xcode
Novembro	Prototipagem rápida plataforma ConnectSDK
	Prototipagem rápida de aplicações para Chromecast
	Prototipagem rápida com bibliotecas de efeitos de vídeo
Dezembro	Estudo de comunicação com câmaras GoPro
	Introdução do projeto "See What I See"
	Integração das câmaras GoPro com a aplicação "See What I See"
Fevereiro	Adição de suporte aos recentes modelos GoPro 4 na aplicação "See What I See"
Abril	Suporte a efeitos de vídeo na aplicação "See What I See"
Junho	Desenvolvimento de aplicação para Apple Watch para controlo remoto de conteúdo em set-top boxes

Tabela 3 – Planeamento final de trabalhos

2.6. Resultados alcançados

Após o término do período de estágio foi possível alcançar vários objetivos, alguns deles previamente estabelecidos e outros alterados, adaptando-se assim às necessidades da empresa e clientes em dado momento.

Sintetizando o conjunto de objetivos alcançados com o estágio, foi possível o desenvolvimento das seguintes aplicações móveis na plataforma iOS:

- Partilha de vídeo em direto a televisão oriundo das câmaras de ação GoPro;
- Filtros de vídeo e *overlay* de imagens em direto;
- Controlo remoto de conteúdo na televisão via Apple Watch.

As três aplicações desenvolvidas representam uma prova de conceito realizada para avaliar o resultado final e interesse por parte de potenciais clientes. O seu desenho seguiu uma estrutura modelar, permitindo a incorporação ou remoção de funcionalidades, fornecendo desta forma flexibilidade no desenvolvimento de diferentes soluções.

A seguinte figura 3 apresenta o resultado final da aplicação de partilha de vídeo em direto com a televisão oriundo das câmaras de ação GoPro. A Vodafone Portugal foi um dos clientes interessados na solução, tendo sido responsável pela adição de algumas novas funcionalidades.

Figura 3 - Aplicação SeeWhatISee com integração das câmaras GoPro

Na seguinte figura 4 podemos visualizar o aspeto final da aplicação de efeitos de vídeo em direto e *overlay* de conteúdo com deteção de faces. Esta solução foi possível recorrendo às bibliotecas OpenCV para deteção de faces e GStreamer para a adição de efeitos de vídeo e imagens ao vídeo final.

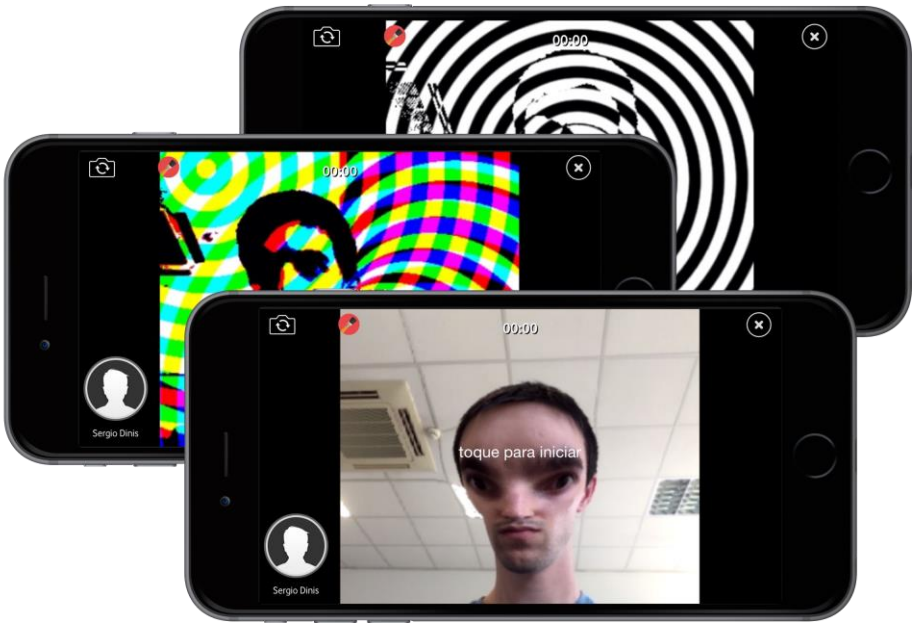

Figura 4 - Aplicação SeeWhatISee com integração de filtros de vídeo

Por último, a figura 5 apresenta o resultado alcançado com o desenvolvimento de uma aplicação móvel para o recentemente introduzido no mercado Apple Watch. Esta aplicação no relógio funciona em modo extensão à aplicação no iPhone. Com ela é possível, a partir do relógio, obter informação acerca da programação do canal em reprodução na televisão, controlo remoto da televisão ou navegação no videoclube. A Vodafone Portugal foi mais uma vez um dos clientes interessados na solução desenvolvida, estando esta integrada com o seu sistema OTT.

Figura 5 - Aplicação para Apple Watch para controlo de set-top boxes

Capítulo 3
Gestão do projeto

A gestão de um projeto de *software* representa um desafio para conseguir um controlo adequado das diferentes variáveis envolvidas, como âmbito, custos, prazos, qualidade e riscos envolvidos. É necessário por isso adotar processos que permitam uma adequada gestão.

Este capítulo destina-se a apresentar a metodologia seguida ao longo deste estágio curricular, processos utilizados no desenvolvimento em equipa e gestão de riscos associados aos desenvolvimentos efetuados.

3.1. Metodologia

Na empresa não existe uma metodologia de trabalho imposta obrigatoriamente, nem tal ação faria completo sentido, pois as metodologias de desenvolvimento devem ser escolhidas tento em conta a equipa, projeto e cliente/consumidor final. A escolha definida foi o desenvolvimento interativo e incremental [3], que tenta com as alterações introduzidas, dar resposta as fraquezas existentes no tradicional modelo de desenvolvimento Waterfall [4].

O desenvolvimento iterativo e incremental é um tipo de desenvolvimento que combina *design* iterativo com o modelo de criação incremental no desenvolvimento de *software*. Este tipo de desenvolvimento contém partes essenciais dos modelos Waterfall, Rational Unified Process [5], Extreme Programming [6] e das várias *frameworks* de desenvolvimento de *software* ágeis.

A ideia básica por trás deste modelo de desenvolvimento, exemplificado na seguinte figura 6, é a criação de um sistema através de ciclos repetitivos (iterativo), em pequenos espaços temporais (incremental). Esta forma permite aos programadores tirar vantagem do conhecimento adquirido no desenvolvimento de partes anteriores.

O desenvolvimento incremental é uma estratégia de planeamento em que várias partes de um determinado sistema são desenvolvidas em paralelo, sendo integradas mais tarde quando completas. Já o desenvolvimento iterativo é uma estratégia de planeamento de trabalho em que o tempo de revisão e aplicação de melhorias é pré-definido.

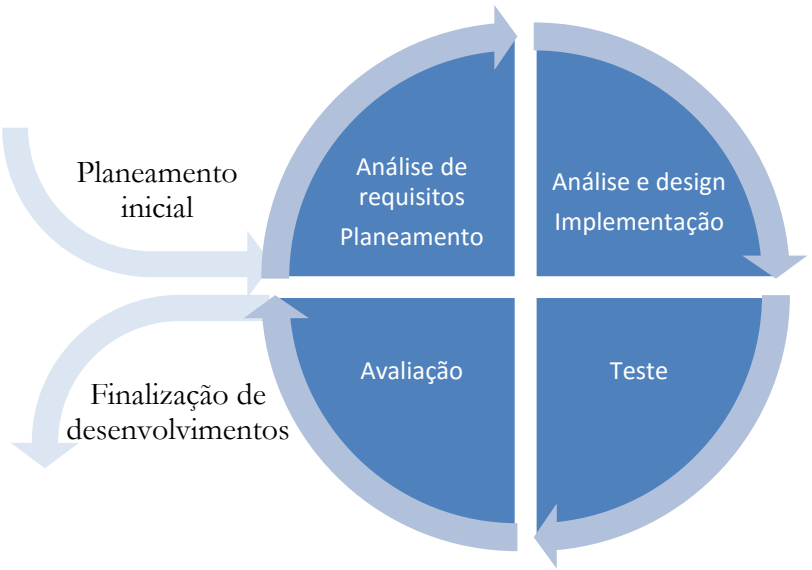

Figura 6 - Modelo de desenvolvimento iterativo e incremental

3.1.1. Vantagens

Entre algumas importantes vantagens conseguidas com esta metodologia encontramos:

- Possibilidade de avaliar mais cedo os riscos e pontos críticos do projeto, permitindo identificar medidas para os eliminar ou controlar;
- Os requisitos mudam com o tempo e um processo iterativo e incremental mantém frequentes os contatos com o cliente, o que ajuda a manter os requisitos sincronizados;
- Redução dos riscos envolvendo custos a um único incremento. Se for necessário repetir uma iteração, a organização perde somente o esforço mal direcionado de uma iteração, não o valor de um produto inteiro;
- Aceleração do tempo de desenvolvimento com um todo pois é mais fácil para o programador trabalhar em projetos de menor dimensão, que juntos formam um projeto de maior dimensão;
- Altamente motivador para a equipa de desenvolvimento (e para o cliente) ver o *software* funcionando mais cedo.

3.2. Processos

Para sistema de controlo de versões foi utilizado o SVN. A sua escolha apenas baseou na utilização prévia por parte dos elementos da equipa. O sistema de controlo de versões facilitou trabalho em equipa, permitindo os desenvolvimentos colaborativos num único repositório.

Os desenvolvimentos dos módulos de partilha de vídeo com a televisão e efeitos de vídeo foram iniciados a partir de *branches* do produto SeeWhatISee existentes, de modo a dotar o produto de novas funcionalidades. No início dos desenvolvimentos foi realizada uma cópia do (branch), onde foram efetuados os desenvolvimentos. No final foi realizado um *merge* com o *trunk*, que representa a reintegração dos novos desenvolvimentos no repositório principal do projeto.

No final de cada fase dos diferentes módulos foi realizado um *code review* ao trabalho realizado por um elemento sénior da equipa de televisão, tendo por objetivo avaliar o cumprimento de *standards* e discutir as decisões de implementação realizadas.

Para comunicação entre os elementos da equipa foi utilizada a recente e popular plataforma Slack [7]. Esta mostrou-se extremamente prática para a criação de diferentes grupos de chat, partilha de ficheiros e registo de histórico de conversações.

3.3. Gestão de requisitos

Para os diferentes módulos existiu a necessidade de realizar o levantamento de requisitos detalhados aquando da sua implementação. Esta forma permitiu adaptar os requisitos às necessidades/interesses do cliente final.

Os requisitos são classificados segundo o seu tipo e prioridade. Para melhor entendimento desta diferenciação o próximo subcapítulo identifica com detalhe estas classificações.

A plataforma Redmine foi utilizada ao longo os desenvolvimentos realizados. Graças a ela foi possível centralizar a gestão dos diferentes projetos, monitorizando o progresso dos diferentes requisitos identificados.

3.3.1. Prioridade de requisitos

Os requisitos identificados apresentam diferentes prioridades consoante o seu contributo para o objetivo final no projeto. Esta divisão é realizada em três níveis, sendo estes Alta, Média ou Baixa. Seguidamente é apresentada uma lista que identifica com maior detalha esta classificação:

- Alta: Requisitos fulcrais para o sucesso do projeto, representando funcionalidades base do mesmo.
- Média: Requisitos que apesar da sua importância para o projeto final não implicam a falha na obtenção dos objetivos estabelecidos.
- Baixa: Requisitos que apesar de adicionarem valor para o projeto têm baixa prioridade de implementação.

3.3.2. Tipo de requisitos

Os requisitos são também classificados quanto ao seu tipo, isto é, a divisão entre requisitos funcionais e requisitos não funcionais. Os requisitos funcionais definem uma função de um *software* ou parte dela, podendo ser cálculos, manipulação de dados, processado de *inputs*, entre outras funcionalidades específicas que definem um determinado sistema. Os não funcionais são requisitos que relacionam o uso da aplicação em termos de desempenho, usabilidade, confiabilidade, disponibilidade, segurança e tecnologias envolvidas.

3.4. Gestão de riscos

A gestão de riscos identifica os riscos que podem vir a afetar o projeto em análise. Estes riscos, quando não identificados e devidamente mitigados, podem condicionar o sucesso do projeto ou conduzir ao mesmo ao fracasso. Para assegurar que tal não se verifica, a gestão de riscos foi analisada e atualizada semanalmente, identificando novos ou atualizando riscos existentes.

Os riscos são classificados segundo o seu nível de impacto:

- Tolerável (1);
- Sério (3);
- Catastrófico (5).

Cada risco é ainda classificado segundo a sua probabilidade de ocorrência com o nível de:

- Muito baixo (1);
- Baixo (2);
- Moderado (3);
- Alto (4);
- Muito Alto (5).

A relevância dos riscos identificados é calculada segundo a formula, **probabilidade x impacto**. Para valores iguais ou superiores a 15 foram apresentadas ações de mitigação, podendo ser estas do tipo de plano de contingência, estratégia de evitação ou estratégia de minimização. Para valores inferiores os riscos apenas são acompanhados por observações.

Capítulo 4
Estado da Arte

A análise do estado da arte é uma tarefa de elevada importância no desenvolvimento de um projeto. É assim possível realizar um estudo de viabilidade do projeto através da análise de soluções existentes, efetuando uma análise de pontos fortes e fracos. Com base nesta informação é possível identificar o risco associado ao projeto e realizar a identificação de possíveis oportunidades.

Correspondendo a diferentes fases do estágio, o presente capítulo está dividido nas seguintes secções:

i. **Análise preliminar:** Apresentação de tendências no segmento televisivo criadas pelo novo paradigma de consumo, o UGC;
ii. ***Streaming devices*:** Análise de dispositivos e tecnologias de comunicação de *streaming devices*;
iii. **Análise de aplicações com suporte a GoPro:** Identificação de principais aplicações que permitem o controlo das câmaras GoPro;
iv. **Análise de aplicações com suporte a efeitos de vídeo:** Identificação de aplicações móveis de efeitos de vídeo mais populares no mercado;
v. **Análise de bibliotecas com suporte a efeitos de vídeo:** Identificação de bibliotecas de suporte a efeitos de vídeo;
vi. **Análise de aplicações para controlo de set-top boxes:** Identificação de principais aplicações com suporte a controlo de set-top boxes.

4.1. Análise preliminar

A televisão sempre foi mais que um aparelho. Durante décadas foi a caixa mágica que reuniu a família à sua volta, foi a janela que permitiu ver além-fronteiras. Para muitos um companheiro, como se de mais um elemento da família se trata-se. Com a introdução da internet para todos e o crescimento do número de dispositivos no mercado, a televisão foi ficando para segundo plano. Novas gerações de utilizadores estão a perder a o interesse na tradicional televisão.

A televisão foi evoluindo, passou-se do preto e branco para a televisão a cores; passou-se de ecrãs pequenos e com baixa qualidade para ecrãs grandes com o último grito no que toca à sua qualidade; adicionamos mais canais, muitos canais; introduzimos o 3D; mas nada está a ser suficiente para inverter os números de abandono do seu interesse [8].

Na internet as alternativas aos tradicionais canais proliferam. Nomes como NetFlix [9], Hulu [10] ou AmazonPrime [11] fazem as delícias de uma nova geração que procura diversidade sem local ou hora marcada.

Nos últimos anos o UGC inunda a internet com horas, dias, anos de informação pronta a ser consumida, partilhada, num sem fim de possibilidades. Plataformas como Youtube [12], Twitch [13], entre outras criam uma nova montra de conteúdo para diferentes tipos de utilizadores.

O conteúdo gerado pelo utilizador ganhou mais força que nunca, é possível encontra-lo na maior parte dos locais na internet. Este tipo de conteúdo tem ajudado as marcas a vender, devido ao seu maior poder de influência na compra de um produto quando comparado com a publicidade tradicional. [14]

4.2. *Streaming Devices*

Os *streaming devices* representam uma das mais populares formas de partilha de conteúdo com a televisão, sendo muito deste conteúdo partilhado UGC.

Neste capítulo são abordados os principais dispositivos disponíveis no mercado, tendo como foco as suas tecnologias para comunicação e plataformas de desenvolvimento.

4.2.1. Análise de dispositivos

Este subcapítulo analisa os principais dispositivos no mercado de *streaming devices*. Para cada um dos dispositivos analisados é apresentada uma breve descrição das suas principais caraterísticas.

4.2.1.1. Chromecast

O Chromecast [15] é um popular adaptador desenvolvido pela Google, para partilha de fotos, música e áudio com a televisão. Permite também a utilização de aplicativos como Netflix [9], Hulu Plus [10], Youtube [12], entre outros.

Em fevereiro de 2014 a Google disponibilizou o seu SDK para desenvolvimento de novas aplicações. Esta *framework* permite o desenvolvimento de dois tipos diferentes de aplicativos:

i. **Aplicativos para envio:** aplicação que disponibiliza controlos de descoberta e envio de conteúdo para um Chromecast;
ii. **Aplicativos para receção:** aplicação web que é executa no Chromecast, num ambiente simular a um *browser*.

Este utiliza o protocolo *multicast* Domain Name System para realizar procura de dispositivos na rede *wifi*. Em versões anteriores o chromecast utilizou o protocolo Discovery and Launch (DIAL).

4.2.1.2. Google Nexus Player

O Nexus Player [16] pretende ser a alternativa do Google aos dispositivos que dão às televisões acesso a conteúdos presentes na internet, disponibilizando uma forma de poder aceder aos mais variados aplicativos presentes na Play Store [17], e também de outros serviços de conteúdos que estão disponíveis na Internet.

Este dispositivo corre a versão Android 5.1 e utiliza e o a funcionalidade de Google Cast, que permite controlar conteúdo na televisão, introduzida inicialmente no Chromecast.

Tal como o Chromecast, o Nexus Player apenas necessita uma porta HDMI e o estabelecimento de uma ligação à internet. Trata-se de um equipamento preparado com Google Cast Ready, permitindo receber conteúdo diretamente de qualquer dispositivo móvel ou de qualquer computador ou Chromebook.

4.2.1.3. Apple TV

A Apple TV [18] é a solução desenvolvida pela Apple com objetivo de reprodução de conteúdo em modo *streaming*. É possível assistir a conteúdo de aplicativos como Netflix, Youtube ou Vimeo. A tecnologia proprietária AirPlay possibilita o envio de conteúdo, fotos, música ou vídeos para a Apple TV.

4.2.1.4. Roku Streaming Player

O Roku Streaming Player [19], mais conhecido simplesmente pelo nome Roku, representa uma linha de dispositivos para reprodução de conteúdo em modo *streaming*. Desde o seu lançamento inicial, no ano de 2008, já foram lançadas duas novas gerações do dispositivo.

A plataforma utilizada por este dispositivo suporta o protocolo Simple Service Discovery Protocol (SSDP), para a descoberta de serviços web, e suporta ainda o protocolo DIAL.

Recentemente foi introduzida uma nova oferta no mercado, denominada por Roku streaming stick, de menores dimensões que os dispositivos Roku até agora apresentados. Este vem competir diretamente com a oferta realizada pelo Google com o Chromecast.

4.2.1.5. Amazon Fire TV

O Amazon Fire TV [20] é a proposta apresenta pela Amazon no mercado de set-top box *streaming*. Permite a visualização de conteúdo presente em populares aplicativos como Netflix ou Youtube. A sua plataforma permite o desenvolvimento de aplicativos por terceiros.

São suportados os protocolos de comunicação Miracast, DIAL e tecnologias proprietárias desenvolvidas pela Amazon.

Tal como o Roku, a Amazon recentemente lançou uma nova oferta, denominada por Fire Tv strick [21]. Esta representa uma proposta minimalista do Fire TV, vindo competir diretamente com o Chromecast e Roku stick.

4.2.2. Análise de tecnologias

Este subcapítulo é realizada a análise das tecnologias de comunicação utilizadas pelos principais dispositivos de *streaming devices* no mercado.

4.2.2.1. DIAL

O Discovery and Launch (DIAL) é um protocolo desenvolvido pelas empresas NetFlix e Youtube, disponibilizando um mecanismo de descoberta e lançamento de aplicações numa rede *wifi* que suportem o protocolo.

4.2.2.2. DLNA

The Digital Living Network Allience foi fundada por um conjunto de empresas eletrónicas para desenvolver e promover um conjunto de diretrizes de interoperabilidade para a partilha de conteúdo entre dispositivos. Este foi baseado em normas anteriormente existentes na tecnologia Universal Plug and Play (UPNP).

4.2.2.3 Miracast

A tecnologia Miracast é uma solução, baseada na especificação *wifi allience - wifi display*, que permite o espelhamento multimédia entre dispositivos via *wifi*.

Utiliza a tecnologia wifi-direct na formação de conexões, evitando a ligação a um ponto de acesso. Opcionalmente suporta também a tecnologia Tunneled Direct Link Setup (TDLS).

4.2.2.4. WiDi

A tecnologia Intel WiDi, desenvolvida pela Intel, representa um aprimoramento da tecnologia Miracast, fornecendo ligações mais rápidas e fiáveis, baixa latência na transmissão e suporte a níveis de qualidade superiores.

Está direcionado para dispositivos com *hardware* intel, computadores ou tablets. É suportado por defeito no Windows 8.1.

4.2.2.5. Airplay

O Airplay é uma tecnologia de exibição de conteúdo sem fios desenvolvida pela Apple para os seus dispositivos. É flexível o suficiente para trabalhar de duas maneiras diferentes, podendo ser usado apenas para espelhar o conteúdo do ecrã do dispositivo, ou usado um modo de *streaming*.

4.3. Análise de aplicações com suporte a GoPro

4.3.1. GoPro App

A GoPro App [22] é aplicação oficial visualização de conteúdo registado com Câmaras GoPro. Em versões das câmaras com suporte *wifi* é também possível o controlo da câmara. Está disponível para as plataformas iOS, Android e Windows Phone.

Nas câmaras com *wifi* é possível:

- Ligar/desligar;
- Alterar modos (foto, vídeo, etc);
- Alterar configurações de imagem, como resolução ou exposição;
- Visualizar pré-visualização;
- Iniciar gravação.

É também possível a visualização de conteúdo presente no cartão de memória da aplicação, possibilitando a sua partilha direta com as redes sociais.

4.3.2. Conclusões

As populares câmaras de ação GoPro são foram desenhadas para permitir aos seus utilizadores registar momentos em modo de fotos ou vídeo. Possui características e qualidades de câmaras profissionais, sendo ao mesmo tempo versátil para ser utilizada em desportos como paraquedismo, automobilismo, surf, motocross, etc.

Tal como a Red Bull, a GoPro construiu uma marca que vai para além de um simples produto, sendo atualmente também considera uma empresa de media [23]. Todos os dias mais de 6 mil vídeos são carregados para o Youtube tendo origem nas câmaras GoPro. A popularidade deste tipo de conteúdos é indiscutível, os seus vídeos no Youtube somam visualizações e avaliações positivas.

A não existência de uma API oficial para este tipo de câmaras limita o desenvolvimento de aplicações para controlo e obtenção de conteúdo.

4.4. Análise de aplicações com suporte a efeitos de vídeo

O desenvolvimento do módulo de efeitos de vídeo para a aplicação SeeWhatISee exigiu uma análise prévia das principais aplicações no mercado. Esta análise permitiu realizar a identificação das suas principais características, bem como dos seus fatores diferenciadores.

4.4.1. Magisto

O Magisto [24] é uma aplicação móvel, também disponível em aplicação web, que disponibiliza uma ferramenta de edição automática de vídeo. Indicando apenas alguns elementos, como o conteúdo a editar e alguns gostos pessoais, a ferramenta cria vídeos com os momentos mais relevantes do conteúdo selecionado aplicando efeitos ao gosto do utilizador.

3.4.2. Instagram (modo vídeo)

O Instagram [25] é uma popular uma popular aplicação de partilha de fotografias e vídeos, permitindo a partilha de conteúdos com diversas redes sociais, entre elas Facebook, Twitter, Tumblr ou Flickr.

Tal como a plataforma Vine [26], o Instagram apenas permite a partilha de vídeos limitados a uma determinada duração máxima. Da mesma forma que os efeitos aplicados em fotos fazem alusão às populares máquinas fotográficas da nossa história, alguns efeitos aplicados aos vídeos recordam as propriedades destas câmaras.

4.5. Análise de bibliotecas com suporte a efeitos de vídeo

Antes de iniciar os desenvolvimentos de funcionalizes de edição de vídeo em aplicações móveis, foi realizada uma análise das diferentes bibliotecas atualmente existentes no mercado. Estas análises são apresentadas seguidamente, sendo finalizadas com uma tabela comparativa.

4.5.1. Av Foundation

A Av Foundation é uma *framework*, para os sistemas operativos iOS e OS X, que disponibiliza o conjunto de ferramentas que permitem a manipulação de conteúdo audiovisual. Para o caso especifico de vídeo, disponibiliza serviços essenciais para trabalhar com elementos baseados em tempo.

Disponibiliza uma interface em Objective-C, estando disponível para as versões 4 ou superiores do iOS. Com esta *framework* é possível reproduzir, capturar e editar vídeo.

4.5.2. GStreamer

O GStreamer é uma *framework*, desenvolvida em linguagem C, para o desenvolvimento de aplicações que lidam com *streaming* de áudio ou vídeo. Com ela é possível reproduzir, capturar, editar e realizar conversão de formatos ao vídeo. No seu repositório podemos encontrar diversos módulos para a adição de efeitos ao vídeo [27].

É utilizada em muitos aplicativos de reprodução e edição de conteúdo, como é o exemplo do Banshee [28] ou Rhythmbox [29].

Disponibiliza compatibilidade multiplataforma, sendo possível utilizá-lo sobre Linux, Solaris, Mac OS X ou Microsoft Windows.

4.5.3. FFmpeg/Avlib

O FFmpeg [30] é uma biblioteca, desenvolvida em linguagem C, que permite a gravação, edição e criação de *streaming* de áudio ou vídeo em diferentes formatos. É composto de uma coleção de *software* livre e bibliotecas de código aberto como libavcodec e libavformat. Com ela é possível reproduzir, capturar, editar e realizar conversão de formatos ao vídeo.

A Avlib [31] representa um *fork* realizado à biblioteca FFmpeg no ano de 2011, iniciando um processo de desenvolvimento de novas versões separadamente. Apesar da separação, continuam a partilhar muito do código, sendo comum encontrar utilizadores da Avlib em fóruns de discussão da biblioteca FFmpeg.

4.5.4. Libopenshot

A libopenshot [32] trata-se de uma biblioteca desenvolvida em linguagem C++, tendo a si associado o desenvolvimento da aplicação de edição de vídeo OpenShot Video Editor.

Tal como outras bibliotecas, apresenta diversas dependências de outras bibliotecas *open source*, que podem não estar disponível para todas as arquiteturas. Por este facto não foi possível confirmar a combatibilidade desta biblioteca com as plataformas iOS ou Android.

4.5.5. GPUImage

O GPUImage [33] trata-se de uma das mais populares biblioteca para o sistema operativo iOS, que permite a aplicação de filtros a imagens e vídeos. Em comparação com o Av Foundation, destaca-se pelo seu suporte de criação de filtros personalizados.

Apesar de ser desenvolvida apenas para a plataforma iOS, existem projetos onde foi realizado o porte da biblioteca para a plataforma Android [34].

4.5.6. Tabela comparativa

	Av foundation	GStreamer SDK	FFmpeg	Libopenshot	GPUImage
Juntar ficheiros de vídeo	✓	✓	✓	✓	✗
Manipular áudio	✓	✓	✓	✓	✗
Cortar vídeo	✓	✓	✓	✓	✗
Sobrepor texturas	✓	✓	✓	✗	✓
Sobrepor texto	✓	✓	✓	✗	✗
Efeitos de vídeo	✓	✓	✗	✓	✓
Criar novos efeitos de vídeo	✗	✓	✗	✓	✓

Tabela 4 – Comparação de funcionalidades suportadas nas bibliotecas de edição analisas

4.6. Análise de aplicações para controlo de set-top boxes

A análise de aplicações com a capacidade para controlar set-top boxes realizada visou identificar a atual oferta existente no mercado. Para cada aplicação analisada são identificadas as suas principais funcionalidades, terminado com uma tabela comparativa, onde são reunidas as aplicações e funcionalidades disponíveis.

4.6.1. Meo Remote

O Meo Remote [35] é uma aplicação móvel, disponível para as plataformas iOS e Android, que permite o controlo remoto da set-top box. Antes e iniciar o controlo é necessário realizar o emparelhamento, bastando para isso estar conectado à mesma rede *wifi* que a set-top box, e seguidamente selecionar na aplicação.

Esta aplicação permite executar toda a lógica que habitualmente encontramos no controlo remoto, com funcionalidades para mudar de canal ou volume. Disponibiliza ainda a informação acerca da programação televisiva, agendamento de gravações, pesquisa de conteúdo e integração com redes sociais.

4.6.2. Iris Remote

A aplicação Iris Remote [36] é fornecida pela empresa de telecomunicações NOS, permitindo o controlo remoto das suas set-top boxes a partir do *smartphone* ou *tablet*. Tal como na aplicação

Meo Remote, é também necessário realizar o emparelhamento com a set-top box pretendida, antes de poder avançar para o seu controlo.

Após realizar o emparelhamento a aplicação passa a disponibilizar toda a lógica disponível num tradicional comando e ainda a possibilidade e visualização da programação televisiva e agendamento de gravações.

4.6.3. Tv Vodafone

A aplicação Tv Vodafone [37] disponibiliza uma experiência completa de televisão no *smartphone* ou *tablet*. Ao contrário das anteriores aplicações analisadas, a Vodafone optou por não disponibilizar uma aplicação móvel apenas dedicada a controlo remoto, juntando todo o conteúdo televisivo numa única aplicação.

Para além dos controlos tradicionais a aplicação da Vodafone disponibiliza ainda a funcionalidade de controlo por comandos de voz.

4.6.4. Peel Smart Remote

A aplicação Peel smart remote [38] apresenta-se como uma solução que permite a criação de um controlo remoto universal, para controlo de televisores, set-top boxes, roku, apple tv, entre outros.

Tendo por base os programas visualizados, a aplicação fornece recomendações inteligentes de programas e guia de TV. Esta aplicação diferencia-se das anteriores analisadas pois utiliza infravermelhos para realizar a comunicação com os diferentes dispositivos.

4.6.5. Tabela comparativa

	Meo Remote	Iris Remote	Tv Vodafone	Peel Smart Remote
Controlos tradicionais	✓	✓	✓	✓
Controlo por voz	✗	✗	✓	✗
Guia Tv	✓	✓	✓	✓
Agendamento de gravações	✓	✗	✓	✓
Pesquisa de conteúdo	✓	✓	✓	✓
Recomendação inteligente	✗	✗	✗	✓
Videoclube	✗	✗	✓	✗
Partilha com redes sociais	✓	✗	✓	✗

Tabela 5 – Comparativo de aplicações para controlo remoto de set-top boxes

Capítulo 5
Levantamento de requisitos

Este capítulo apresenta a definição de requisitos, funcionais e não funcionais, para os três módulos desenvolvidos durante o estágio, sendo eles:

i. **Partilha de vídeo proveniente da GoPro com a televisão:** desenvolvimento de módulo de comunicação com câmaras GoPro para integração com aplicação SeeWhatISee;
ii. **Adição de efeitos de vídeo em direto:** desenvolvimento do módulo de efeitos de vídeo em direto para integração com aplicação SeeWhatISee;
iii. **Controlo de conteúdo em STBs via Apple Watch:** desenvolvimento de aplicação para Apple Watch para controlo e visualização de conteúdo em set-top boxes.

5.1. Partilha de vídeo proveniente da GoPro com a televisão

5.1.1. Objetivo

Desenvolvimento um módulo para aplicação SeeWhatISee que permite a integração das câmaras de ação GoPro. Este módulo fica responsável pelas tarefas de gestão de comunicação com a camara, bem como de integração dos diferentes formatos de vídeo com o fluxo de vídeo da aplicação.

5.1.2. Requisitos funcionais

RF-01 – API de comunicação com a GoPro
O módulo da aplicação a desenvolver deve ser capaz de comunicar com as câmaras GoPro para realizar diversas funções, sendo elas ligar/desligar a câmara, iniciar/parar a gravação de previsualização, obtenção de *status* da câmara contendo informações acerca da bateria, outras definições da câmara e implementação de funcionalidade *keep-alive*.

RF-02 – Deteção de ligação com a GoPro
Esta funcionalidade permitirá testar a existência de uma conexão a uma câmara GoPro.

RF-03 – Integração do UI GoPro com a aplicação
Inclui a adição de uma nova entrada no menu de escolha de *sources* (câmara do dispositivo, galeria ou câmara GoPro) de partilha de conteúdo, *placeholders*, entre outras alterações que permitirão a escolha na aplicação destas funcionalidades.

RF-04 – Captura de vídeo da GoPro
Deve ser implementada toda a lógica que permita a obtenção da pré-visualização de vídeo da câmara GoPro.

RF-05 – Exibição da pré-visualização proveniente da GoPro na aplicação
A aplicação deve exibir uma pré-visualização do *stream* proveniente da câmara GoPro, de forma a poder visualizar a informação que irá estar a transmitir.

RF-06 – Envio do *stream* proveniente da GoPro aos servidores Wowza
Para que a partilha com os televisores seja conseguida deve existir uma comunicação aos servidores Wowza com o envio contínuo contendo a informação do *stream* de vídeo e áudio proveniente da câmara GoPro.

RF-07 – Bloquear/desbloquear o toque no ecrã do dispositivo durante uma partilha
A aplicação deve incorporar um botão que irá permitir bloquear o toque no seu ecrã. Depois de bloqueado deve implementar uma funcionalidade robusta para a inativação deste bloqueio, impedindo desbloqueios indesejados.

RF-08 – Desativar a pré-visualização durante o bloqueio
A aplicação deverá desativar a pré-visualização proveniente da GoPro após o início da funcionalidade de bloqueio, de forma a poupar bateria no dispositivo.

RF-09 – Indicação de alerta de bateria fraca da GoPro durante uma partilha
A aplicação consumindo a informação obtida pelo **RF-01** deverá informar o utilizador quando, durante uma partilha, o nível de bateria da câmara GoPro se aproximar de valores baixos.

RF-10 – Verificação do estado da GoPro e ligar automáticamente
A aplicação consumindo a informação obtida pelo **RF-01** deverá, em caso de deteção de GoPro desligada, proceder ao seu inicio e continuar o fluxo normal de partilha.

RF-11 – Deteção de *Bandwidth* disponível na rede a que está conectado
A aplicação deve ser capaz de calcular a *bandwidth* disponível na rede de forma a garantir condições mínimas em que é possível efetuar uma partilha com a televisão.

RF-12 – Adição de suporte para *tablets* iPad
A aplicação deve ser capaz de adaptar a informação e sua interface a diferentes tamanhos de dispositivos, entre eles os tablets iPad de diversos tamanhos de ecrã.

RF-13 – Suporte para Google Analytics
A aplicação deve fornecer dados estatísticos acerca da sua utilização ao serviço Google Analytics, permitindo desta forma o *tracking* utilização da aplicação para fornecer dados estatísticos ao cliente por detrás da aplicação.

5.1.3. Requisitos não-funcionais

RnF-1 – A aplicação ter consumos de bateria otimizados
A aplicação deverá de forma mais otimizada, realizar as ações necessárias a uma partilha, conseguindo níveis de consumo de bateria baixos.

RnF-2 – O código desenvolvido deve cumprir os *standards*
O código desenvolvido deve adaptar-se à estrutura do projeto existente cumprindo os *standards* seguidos pela empresa. Deve igualmente cumprir os *standards* de aprovação da App Store da Apple.

RnF-3 – A arquitetura desenvolvida deve cumprir os *standards*
A arquitetura do projeto desenvolvido deve cumprir os *standards* utilizados na empresa e os de aprovação na App Store da Apple.

RnF-4 – A interface implementada deve estar de acordo com a interface existente
A interface dos novos menus da aplicação a desenvolver deve obedecer aos estilos já utilizados pela aplicação existente.

5.2. Adição de efeitos de vídeo em direto

5.2.1. Objetivo

Desenvolvimento um módulo para aplicação SeeWhatISee que permite a adição de efeitos de vídeo em direto. O novo módulo desenvolvido deve substituir o atual fluxo de vídeo existente na aplicação.

O módulo deve ser capaz realizar a pré-visualização dos efeitos de vídeo, bem como de aplica-los ao vídeo partilhado com a televisão.

5.2.2. Requisitos funcionais

RF-01 – Integração da biblioteca Gstreamer no projeto iOS
A biblioteca de edição de vídeo Gstreamer deve ser incluída num projeto para a plataforma iOS.

RF-02 – Criação de core multiplataforma em c++
Deverá ser criada uma estrutura em código c++ para partilha de código entre plataformas iOS e Android

RF-03 – Adição de lógica da biblioteca Gstreamer ao core multiplataforma
A lógica da biblioteca Gstreamer deverá ser estruturada na linguagem de programação c++ para adição ao core/interface multiplataforma.

RF-04 – Criação de gestor de pipeline para adição/remoção de efeitos em vídeo em direto
Deverá ser criado um gestor de pipeline que permita a adição, alteração e remoção de efeitos de vídeo em direto. Estas alterações não devem comprometer a estabilidade do vídeo final produzido.

RF-05 – Adição de gestor de pipeline ao core multiplataforma
A lógica do gestor de pipeline deverá ser estruturada na linguagem de programação c++ para adição ao core/interface multiplataforma.

RF-06 – Integração da biblioteca OpenCV no projeto iOS
A biblioteca de deteção facial OpenCV deve ser incluída num projeto para a plataforma iOS.

RF-07 – Adição de lógica da biblioteca OpenCV ao core multiplataforma
A lógica da biblioteca OpenCV deverá ser estruturada na linguagem de programação c++ para adição ao core/interface multiplataforma.

RF-08 – Integração módulo de efeitos de vídeo com a aplicação SeeWhatISee
O módulo de efeitos de vídeo criado deverá ser integrado com a aplicação SeeWhatISee, procedendo às alterações no fluxo de vídeo necessárias.

RF-09 – Pré-visualização com o resultado final das transformações ao vídeo
O resultado final das transformações efetuadas ao vídeo, efeitos de vídeo e *overlay* de imagens, deve ser exibido no espaço destinada para a previsualização na aplicação SeeWhatISee.

RF-10 – Adição de opção no UI para ativar/desativar efeitos de vídeo
A aplicação SeeWhatISee deverá estar preparada para, através de um toque num botão, ativar ou desativar a aplicação de efeitos de vídeo ao vídeo final produzido.

RF-11 – Adição de opção no UI para ativar/desativar deteção de faces com *overlay* de imagens
A aplicação SeeWhatISee deverá estar preparada para, através de um toque num botão, ativar ou desativar a deteção de faces com *overlay* de imagens.

RF-12 – Adição de seleção no UI para escolha entre diferentes efeitos de vídeo
A aplicação SeeWhatISee deverá suportar a transição entre diferentes efeitos de vídeo através de *swipe* horizontal.

RF-13 – Adição de seleção no UI para escolha entre diferentes imagens a realizar *overlay*
A aplicação SeeWhatISee deverá suportar a transição entre diferentes conjuntos de imagens a realizar *overlay* através de *swipe* horizontal.

RF-14 – Adição de suporte para *tablets* iPad
O UI definido para iPhone deverá ser portado para iPads, adaptando a informação contida na interface a diferentes tamanhos de ecrã.

5.2.3. Requisitos não-funcionais

RnF-1 – A adição de lógica ao fluxo de vídeo não deve comprometer a qualidade do vídeo final produzido.
O nível de *frames* por segundo (FPS) após o processamento do vídeo deve ser superior ou igual a 15.

RnF-2 – A aplicação deve manter consumos de bateria otimizados
A aplicação deverá, de forma mais otimizada possível, realizar as ações necessárias a uma partilha, conseguindo níveis de consumo de bateria otimizados.

RnF-3 – O código desenvolvido deve cumprir os *standards*
O código desenvolvido deve adaptar-se à estrutura do projeto existente, cumprindo os *standards* utilizados na empresa. Deve igualmente cumprir os *standards* de aprovação da App Store da Apple.

RnF-4 – A arquitetura desenvolvida deve cumprir os *standards*
A arquitetura do projeto desenvolvido deve cumprir os *standards* utilizados na empresa e os de aprovação na App Store da Apple.

RnF-5 – A interface implementada deve estar de acordo com a interface existente
A interface dos novos menus da aplicação a desenvolver deve obedecer aos estilos já utilizados pela aplicação existente.

5.3. Controlo remoto de conteúdo na set-top box via Apple Watch

5.3.1. Objetivo

Desenvolvimento uma de aplicação para o Apple Watch e uma aplicação para iPhone. A aplicação para Apple Watch deve ser capaz de controlar remotamente uma set-top box, bem como visualizar a sua programação e exibir o videoclube. A aplicação para iPhone deve disponibilizar o formulário de login de utilizador e emparelhamento com set-top boxes disponíveis na rede *wifi*.

5.3.2. Requisitos funcionais

RF-01 – iPhone: ecrã de login
A aplicação para iPhone deverá exibir um ecrã de login da primeira vez que o utilizador entra na aplicação. Este ecrã deverá autenticar um dado utilizador verificando as suas credenciais no OTT da Vodafone.

RF-02 – iPhone: ecrã de listagem de STBs
Após um dado utilizador realizar login com sucesso na aplicação, deverá ser exibido um ecrã que lista as STBs disponíveis na rede wifi a que o iPhone está conectado.

RF-03 – iPhone: Gestão de listagem e comunicação com STB
No iPhone, o utilizador poderá selecionar uma data STB para poder utilizar mais funcionalidades da aplicação no Apple Watch. Ao selecionada uma dada STB deverá ser verificado o seu emparelhamento, e em caso negativo, deve ser realizada a ação contida no RF-05.

RF-04 – iPhone: Verificação de estado de ligação entre STB e Apple Watch
De forma a poder dar indicações ao utilizador no relógio, quando algum elemento não está devidamente conectado, deverá ser realizada uma monitorização permanente de estados dos respetivos *devices*.

RF-05 – iPhone: Emparelhamento automático com STB
O iPhone deverá realizar o emparelhamento automático com uma data STB. Para isso deverá recorrer a um serviço contido no OTT para efetuar o emparelhamento com a respetiva STB. Após enviar o pedido deverá ficar a pingar a STB até confirmar o emparelhamento com sucesso.

RF-06 – iPhone: Proxy API – enviar comandos a STB
A extensão à aplicação principal do iPhone deverá incluir um módulo responsável por enviar um dado comando, recebido a partir do relógio, à STB selecionada.

RF-07 – iPhone: Proxy API – obter informação de dado canal a reproduzir na STB
A extensão à aplicação principal do iPhone deverá incluir um módulo responsável por obter informação acerca de um canal em reprodução na STB. Esta informação deverá ser cruzada com informação obtida no TvGuide via OTT.

RF-08 – iPhone: Proxy API – obter informação relativa aos VODs
A extensão à aplicação principal do iPhone deverá incluir um módulo responsável por obter informação relativa aos VODs (categorias, capas, detalhes) a partir do OTT.

RF-09 – iPhone: Proxy API – abrir página com detalhes de um VOD
A extensão à aplicação principal do iPhone deverá incluir um módulo responsável por enviar um pedido à STB para abrir terminado endereço (neste caso em especifico, os endereços das páginas de detalhes dos VODs).

RF-10 – Apple Watch: lógica aplicacional de integração com proxy API
A aplicação a executar no relógio deverá contactar a extensão à app principal no iPhone para realizar ações que necessitam de ligação *wifi*.

RF-11 – Apple Watch: ecrã principal
A aplicação no relógio deverá mostrar no seu ecrã principal três menus, que devem permitir a seleção das opções "No ar", "Videoclube" e "Comando"

RF-12 – Apple Watch: ecrã "No ar"
O ecrã "No Ar" no relógio deverá mostrar a informação do programa anterior, atual e seguinte para um dado canal ativo na STB. O ecrã deverá exibir o logo do canal, número, título do programa, barra de duração e descrição. Para os programas atual e futuro deverá ser implementada a funcionalidade de gravação; via OTT para programa futuro e via STB para programa atual.

RF-13 – Apple Watch: ecrã de VODs – listagem de categorias
O ecrã de "videoclube" deve mostrar no seu início dois níveis de categorias, correspondendo a categorias e subcategorias de uma data categoria.

RF-14 – Apple Watch: ecrã de VODs – listagem de VODs associados a uma categoria
O ecrã de "videoclube", para uma data subcategoria, deve exibir as capas dos VODs associados. Deve permitir a navegação entre as diferentes capas.

RF-15 – Apple Watch: ecrã de VODs – descrição de um dado VOD
O ecrã de "videocluble", aquando selecionado um dado VOD, deve exibir um ecrã com os seus detalhes. Deve fazer parte desta informação, quando disponível, o seu título, descrição, duração, classificação etária, elenco, realização, país e classificação segundo a plataforma IMDB.

RF-16 – Apple Watch: ecrã de Comando – teclas direcionais
O primeiro ecrã "comando" deverá exibir os controlos compostos pelas teclas direcionas, ok e voltar.

RF-17 – Apple Watch: ecrã de Comando – teclas de controlo de volume e canal
O segundo ecrã "comando" deverá exibir os controlos de volume, mais e menos, e controlo de programas, mais e menos.

RF-18 – Apple Watch: ecrã de Comando – teclas numéricas e botões coloridos
O terceiro ecrã "comando" deverá exibir os controlos numéricos, compostos pelos dígitos de zero a nove e tecla colorias, verde, vermelho, azul e amarelo.

RF-19 – Apple Watch: ecrã de Comando – teclas multimédia
O último ecrã "comando" deverá exibir os controlos de reproduzir, pausar, parar, gravar, ir para o início, ir para o final, avançar velocidade e diminuir velocidade de reprodução.

RF-20 – Apple Watch: ecrã de mensagens de erro
Sempre que o utilizador tente realizar uma ação e não reúna todos os requisitos, ou surja um problema inesperado, deverá ser apresentado um ecrã de erro a informar acerca da situação.

5.3.2. Requisitos não-funcionais

RnF-1 – O código desenvolvido deve cumprir os *standards*
O código desenvolvido deve cumprir os *standards* seguidos pela empresa. Deve igualmente cumprir os *standards* de aprovação da App Store da Apple.

RnF-2 – A arquitetura desenvolvida deve cumprir os *standards*
A arquitetura do projeto desenvolvido deve cumprir os *standards* utilizados na empresa e os de aprovação na App Store da Apple.

Capítulo 6
Desenho das aplicações

A arquitetura é um componente essencial em qualquer desenvolvimento de *software*, pois esta define a estrutura e comportamento dos diferentes elementos envolvidos no sistema.

O presente capítulo está dividido nas seguintes secções:

i. **Fundamentos iOS:** descreve os principais aspetos arquitetónicos do desenvolvimento de aplicações para iPhone e iPad;
ii. **Fundamentos WatchKit:** descreve a estrutura das aplicações não nativas para o Apple Watch e de que forma se integram com as aplicações móveis iOS;
iii. **Solução SeeWhatISee:** apresenta a arquitetura da solução SeeWhatISee desenvolvida pela empresa, solução esta que serviu de base a alguns dos desenvolvimentos efetuados durante o estágio;
iv. **Partilha de vídeo em direto oriundo das câmaras GoPro:** define a arquitetura utilizada para controlo e captura de vídeo em direto;
v. **Efeitos de vídeo em direto:** descreve a arquitetura para a funcionalidade de edição de vídeo e sua integração com a biblioteca Gstreamer;
vi. **Controlo de conteúdo em STBs via Apple Watch:** disponibiliza a arquitetura seguida na criação de uma aplicação não nativa para Apple Watch.

6.1. Fundamentos iOS

Os desenvolvimentos, realizados durante o estágio curricular, foram efetuados para a plataforma móvel da Apple, o iOS. É por isso importante compreender os seus aspetos arquitetónicos mais relevantes.

6.1.1. Modelo vista controlo

A linguagem de desenvolvimento utilizada ao longo do projeto foi o Objective-C, tendo alguma da lógica multiplataforma sido desenvolvida recorrendo a linguagem C++, abordado em detalhe no próximo capitulo 7.1.1 - código partilhado.

Todos os componentes de interface com o utilizador do projeto seguem do padrão MVC, que atribui um dos três papéis a cada objeto da aplicação: modelo, vista e controlo. A seguinte figura 7 mostra o fluxo existente entre os três diferentes elementos.

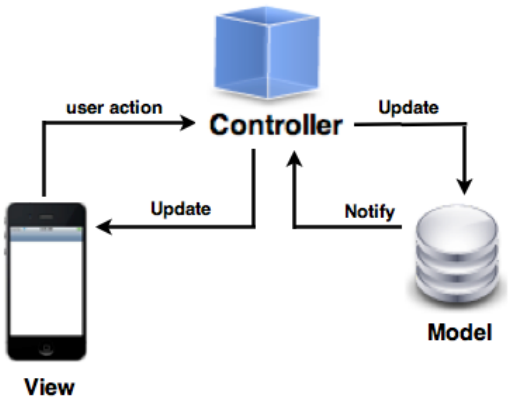

Figura 7 - Modelo vista controlo utilizado pela plataforma iOS [39]

i. **Modelo** – Camada que contém a lógica e dados aplicação. É responsável pela definição da lógica que manipula e processa a informação
ii. **Vista** – Camada da aplicação visualizada pelo utilizador, tendo como responsabilidade exibir ao utilizador os dados armazenados no modelo.
iii. **Controlo** – Intermediário entre o Modelos e a Vistas, sendo responsável pela notificação de alterações no modelo à vista e vice-versa.

O uso do padrão MVC na aplicação contribui para uma separação entre tarefas e melhor reutilização de objetos.

6.1.2. Ciclo de vida

As aplicações em iOS têm os seguintes estados possíveis:

i. **Desligada** – A aplicação não foi iniciada ou está em funcionamento mas foi encerrada pelo sistema
ii. **Inativa** – O aplicativo é executado em primeiro plano, mas atualmente não está a receber eventos.
iii. **Ativa** - O aplicativo é executado em primeiro plano e está a receber eventos. Este é o modo padrão para aplicativos em primeiro plano.
iv. **Background** – A aplicação está em background, em execução, antes de passar ao estado de suspensa
v. **Suspensa** – A aplicação está em segundo plano, sem estar a executar qualquer código. Enquanto suspenso, um aplicativo permanece em memória.

A seguinte figura 8 apresenta as possíveis transições entre os diferentes estados da aplicação.

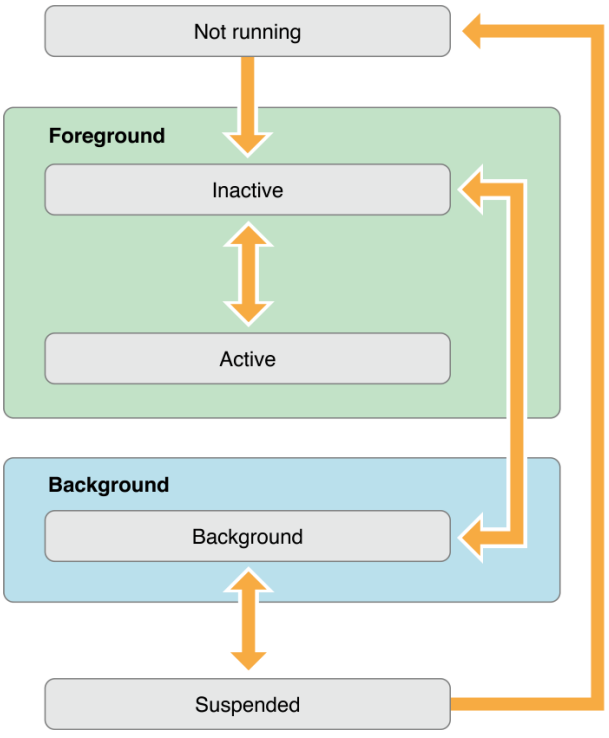

Figura 8 – Ciclo de vida de uma aplicação em iOS [40]

A maioria das transições entre estados de execução da aplicação são acompanhadas por uma chamada correspondente aos métodos do aplicativo delegado. Estes permitem detetar e responder a alterações de estado de uma forma adequada. A seguinte lista apresenta os métodos desencadeados nas transições entre diferentes estados.

i. **application:willFinishLaunchingWithOptions** – Este método representa a primeira oportunidade para executar código após o início da aplicação;
ii. **application:didFinishLaunchingWithOptions** – Este método permite realizar inicializações finais antes de a aplicação ser exibida;
iii. **applicationDidBecomeActive** – Este método permite identificar que a aplicação veio a *foreground*;
iv. **applicationWillResignActive** – Este método informa a aplicação que surgiu uma interrupção que obrigou a uma mudança de estado. Esta interrupção poderá passar pelo recebimento de uma chamada telefónica ou uma mensagem de SMS;
v. **applicationDidEnterBackground** - Este método informa a aplicação que irá passar para *background* e será suspensa de seguida;
vi. **applicationWillEnterForeground** – Este método é chamado quando a aplicação está a transitar do estado de *background* para o estado inativo. Aqui devem ser anuladas as possíveis alterações realizadas quando a aplicação passou para o estado de *background*;
vii. **applicationWillTerminate** – Este método informa a aplicação que esta irá ser terminada.

6.1.3. Grand Central Dispatch

Os processadores *multi-core* possuem dois ou mais núcleos. A partir das gerações do iPhone 4S, iPad 2 e iPod Touch de quinta geração foi introduzido o processador da Apple A5, com suporte multi-core.

Cada núcleo é capaz de realizar mais de uma tarefa em simultâneo, possuindo a sua própria cache. Esta autossuficiência que os diferentes núcleos possuem permite realizar uma divisão de tarefas. Para conseguir uma boa experiencia de utilização na aplicação desenvolvida é necessário fazer uma correta gestão de recursos. Os pedidos à API devem realizados numa tarefa secundária, permitindo o não bloqueio da tarefa principal responsável pela gestão da interface, e consequentemente mantendo a fluidez da aplicação.

A otimização da gestão de múltiplas tarefas, em sistemas *multi-core*, no ambiente de desenvolvimento iOS é possível usado o Grand Central Dispatch (GCD), tecnologia desenvolvida pela Apple, introduzida na quarta geração do seu sistema operativo móvel.

A ideia fundamental passa pela retirada da gestão de tarefas das mãos dos desenvolvedores, tornando-a mais próxima do sistema operativo. Este modelo melhora a simplicidade no escalonamento, portabilidade e performance final das tarefas.

O desenvolvedor tem ao seu dispor diferentes filas de prioridade para realizar a execução da respetiva tarefa. Existem dois tipos de execução de tarefas possíveis:

i. **Serial Queues** – Execução de uma tarefa de cada vez, cada tarefa apenas é iniciada quando a precedente tarefa tiver terminado.

ii. **Concurrent Queues** – Execução de múltiplas tarefas em simultâneo, sendo a sua gestão da responsabilidade do GCD.

O sistema disponibiliza um tipo especial de fila em série conhecida como Main Queue. A sua fila garante a execução de tarefas na *thread* principal, sendo a única com capacidade de realizar alterações a nível de UI.

Estão ainda ao dispor quatro filas de execução concorrentes com diferentes níveis de prioridade, sendo elas Alta, Padrão, Baixa e Background. Por último, é ainda possível criar filas de execução em séries concorrentes. É possível visualizar a estrutura de filas identificada na seguinte figura 9.

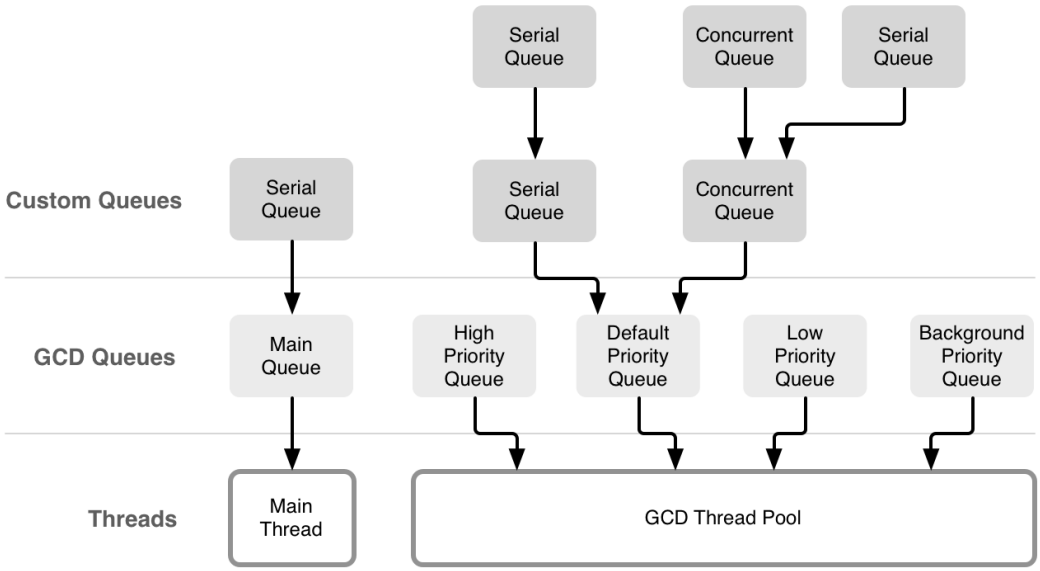

Figura 9 – Estrutura das diferentes filas de execução em iOS [41]

De forma a garantir uma correta gestão de tarefas por parte do GCD é necessário que sejam feitas as escolhas certas das diferentes filas de execução para cada tarefa.

6.2. Fundamentos WatchKit

O Apple Watch é o mais recente produto apresentado pela Apple, um relógio interativo que funciona como complemento ao iPhone. O WatchKit é o seu SDK, permitindo o desenvolvimento de aplicações de terceiros. Esta solução concorre diretamente com a solução previamente apresentada pela Google, o Android Wear [42].

6.2.1. Estrutura

A Execução de aplicações de terceiros no Apple Watch necessita da realização prévia de emparelhamento com um iPhone.

A criação de uma aplicação de terceiros obriga ao desenvolvimento de dois grupos diferentes, a WatchKit App, que é executada no relógio, e a extensão WatchKit, que é executada no iPhone previamente emparelhado.

i. **WatchKit App** – Executada no relógio, contém apenas a interface da aplicação (storyboards e recursos associados à interface)
ii. **WatchKit Extension** – Executada no iPhone em background, contém toda a lógica necessária à utilização da aplicação no relógio (segundo o modelo MVC, esta extensão contém o modelo e o controlo)

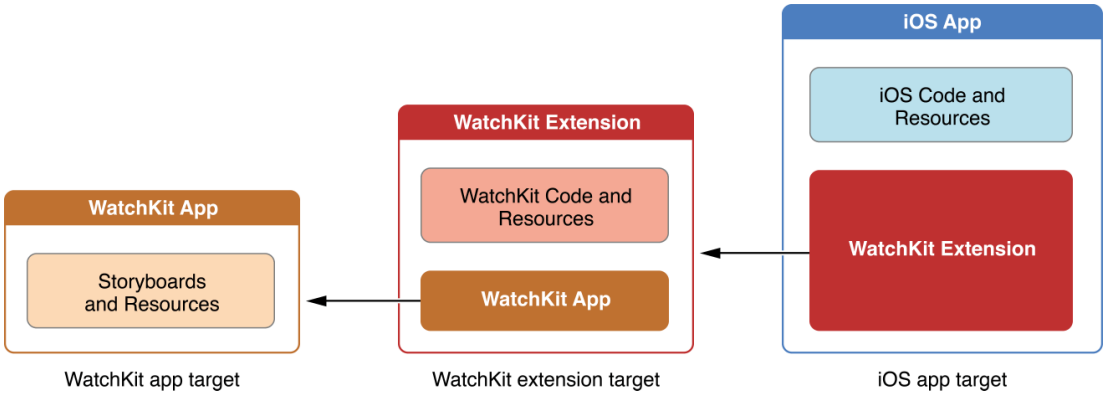

Figura 10 Estrutura de uma aplicação WatchKit [43]

A figura 10 ilustra a estrutura da aplicação iOS e executáveis WatchKit. A aplicação WatchKit é armazenada dentro da extensão WatchKit, estando esta igualmente armazenada dentro de uma aplicação iOS.

O utilizador, ao instalar uma aplicação no iPhone com extensão para Apple Watch, é informado se pretende instalar a aplicação para o Apple Watch emparelhado.

6.2.2. Comunicação

Como já identificado no capítulo anterior, o Apple Watch e respetivo iPhone emparelhado necessitam de funcionar em conjunto para poder disponibilizar uma aplicação não nativa no relógio. A transferência de informação entre as duas realiza-se via *bluetooth*, sendo que quando a aplicação é iniciada ou um novo ecrã exibido é realizado o pedido dos objetos necessários para gerir uma dada vista.

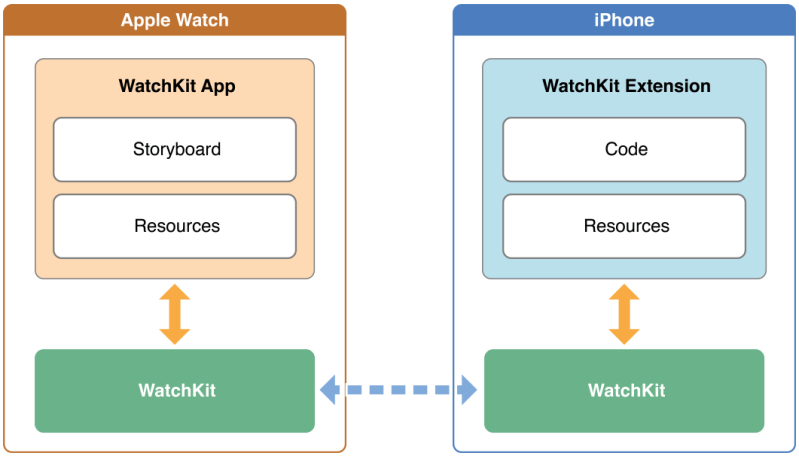

Figura 11 – Comunicação entre a aplicação WatchKit e respetiva extensão [43]

A figura 11 ilustra o fluxo de comunicação entre a extensão e a aplicação não nativa no Apple Watch. A transferência de informação entre as duas partes é totalmente invisível para o utilizador final.

6.2.3. Ciclo de vida

As aplicações WatchKit, tal como as aplicações para iPhone, têm um ciclo de vida associado. Na seguinte figura 12 podemos visualizar a transição entre diferentes estados possíveis desde a inicialização de uma aplicação não nativa no Apple Watch.

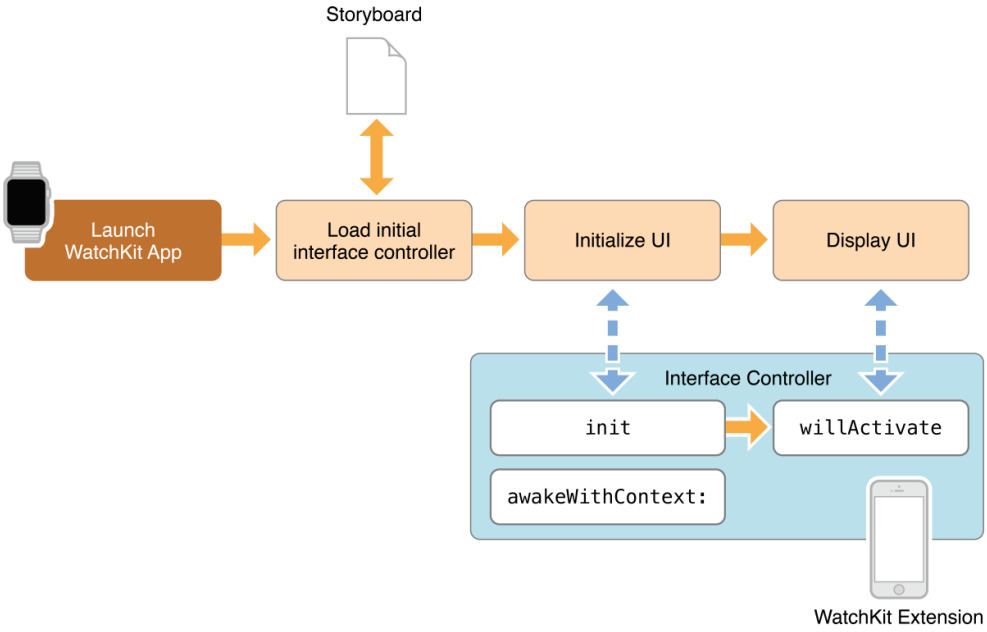

Figura 12 – Ciclo de vida de uma aplicação WatchKit aquando da sua inicialização [43]

A seguinte lista apresenta os principais métodos desencadeados nas diferentes transições entre estados de uma aplicação WatchKit.

i. **Init** - Este método informa a aplicação que acabou de iniciar, representando a primeira oportunidade para inicializar a interface;
ii. **awakeWithContext** – Este método informa que a aplicação iniciou com contexto associado. É habitualmente utilizado para passar informação entre diferentes interfaces;
iii. **willActivate** – Este método é desencadeado no momento em que a aplicação irá ser brevemente exibida. Deverá ser utilizado apenas para realizar pequenas alterações sobre a interface;
iv. **didDeactivate** – Este método é desencadeado no momento em que a interface irá ser desativada. Poderá, entre diferentes ações, ser utilizado para limpar a interface, deixando-a num estado padrão para mais tarde ser reutilizada.

6.3. Solução SeeWhatISee

Dois dos três módulos desenvolvidos foram integrados com a aplicação SeeWhatISee. Estes desenvolvimentos representaram desafios de compreensão da plataforma de modo a realizar o correto desenvolvimento das funcionalidades requeridas.

De forma a perceber a solução na qual os desenvolvimentos foram integrados, será de seguida realizada uma apresentação da solução SeeWhatISee e sua plataforma de suporte.

6.3.1. Visão geral da plataforma

A solução SeeWhatISee, desenvolvida pela empresa WIT-Software para operadores de televisão e TELCO, tem como objetivo a partilha de UGC com a televisão. Esta solução foi idealizada para permitir a partilha de vídeos, fotos e música, a partir de uma aplicação móvel, com a televisão. Está pensada para permitir a captura e envio de vídeo em tempo real da câmara do dispositivo, conteúdo previamente armazenado no dispositivo ou conteúdo disponível em fontes como Youtube, Instagram ou Facebook.

Esta solução está igualmente pensada para funcionar com uma vasta gama de dispositivos como smartphones e tablets Android e iOS, smartwatches, Google Glass, etc. Do lado do televisor a solução está desenhada para comunicar com set-top boxes que corram o *midleware* Ericsson Mediaroom. Está nos planos futuros da empresa para esta solução alargar a compatibilidade a outras plataformas de televisão.

6.3.2. Ericsson Mediaroom

O Ericsson Mediaroom representa da oferta na área televisiva fornecida pela Ericsson. Originalmente o Mediaroom foi desenvolvido pela Microsoft, sendo adquirido pela Ericsson no ano de 2013 [44].

Esta plataforma apresenta-se como líder no mercado IPTV *end-to-end* [45], permitindo aos operadores de televisão fornecer um conjunto diversificado de serviços televisivos aos seus clientes.

A plataforma disponibiliza um ambiente de desenvolvimento, Applications Development Kit (ADK), que permite a criação de aplicações. Estas aplicações representam uma extensão do ambiente televisivo tradicional, permitindo a integração de produtos e serviços com a plataforma. As aplicações desenvolvidas para a plataforma são denominadas Presentation Framework Applications (PF Apps).

A solução SeeWhatISee utiliza ainda uma PF App de forma a permitir a abertura e reprodução do vídeo partilhado com a televisão.

6.3.3. Plataforma SeeWhatISee

A plataforma SeeWhatISee, desenvolvida até ao momento, foca-se na partilha de vídeo em direto a partir de *smartphones* ou *tablets* de forma síncrona e imediata. O vídeo é possível de partilhar com contatos que tenham suporte à sua infraestrutura. A recente oferta por parte da Vodafone Portugal, LiveOnTv [46], é o exemplo mais notório solução desenvolvida pela WIT-Software.

A arquitetura geral da plataforma pode ser visualizada na seguinte figura 13.

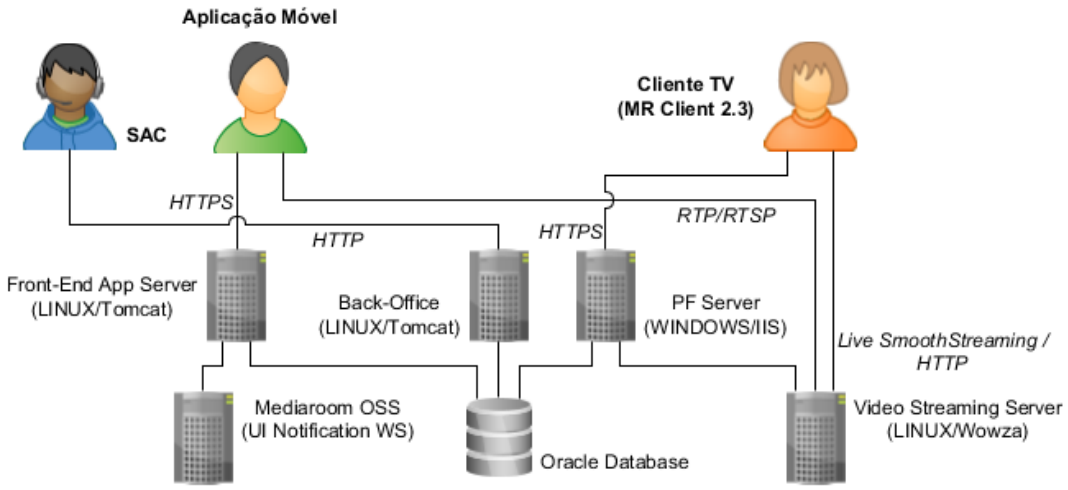

Figura 13 – arquitetura geral da aplicação SeeWhatISee

Para suportar a solução existe a necessidade de instalar alguns componentes na infraestrutura de *back-end* dos operadores.

A aplicação móvel SeeWhatISee, aquando da sua inicialização, contata do servidor *front-end*, de forma a obter configurações e sincronizar a lista de contatos presentes do dispositivo. Esta sincronização obtém como resposta a lista de contatos que têm suporte para a solução SeeWhatISee na plataforma Ericsson MediaRoom associada ao contato.

O vídeo em direto registado a partir da aplicação móvel é enviado no formato Real Time Streaming Protocol (RTSP) ao servidor de *streaming* de vídeo, ficando este responsável pela conversão do formato do vídeo inicial RTSP em Live Smooth Streaming.

Aquando de uma partilha de vídeo em direto com um contato o servidor front-end é responsável pelo envio de uma notificação do servidor PF, para este avisar o cliente final foi iniciada uma partilha de vídeo consigo.

O contacto final, através do seu televisor com suporte ao *midleware* Ericsson Mediaroom, pode aceitar ou recusar a visualização da partilha de vídeo em direto. Se aceitar é aberta uma aplicação PF que irá apresentar o *streaming* de vídeo. O vídeo partilhado estará também disponível no histórico de partilhas recebidas na aplicação disponível para a televisão.

A aplicação móvel SeeWhatISee é suportada pelas plataformas móveis iOS com a versão mínima 6.0 e Android com a versão 3.0 ou superior. O cliente de televisão disponibilizado pelo operador necessita da versão 2.3 do Mediaroom.

O mediaroom OSS é o *webservice* responsável pela gestão de notificações no *midleware* Mediaroom. Graças a ele é possível lançar uma nova notificação na televisão com a informação de uma nova partilha de vídeo.

Para servidor de *streaming* de vídeo foi escolhida a solução Wowza Media systems, permitindo a reprodução de vídeo em direto ou on-demand.

As máquinas responsáveis pelo Front-End e Back-End utilizam sistema operativo baseado em Linux com apache tomcat, estando responsável pelos diferentes *servlets* de suporte à plataforma.

O sistema de base de dados utilizado recorre à plataforma Oracle, ficando responsável pelo registo de partilhas entre utilizadores. A sua informação serve de suporte ao envio de notificações nos dois sentidos, na televisão para informar que chegou uma nova partilha, ou no *smartphone* para notificar se os destinatários estão ou não a visualizar o conteúdo partilhado.

6.3.3.1. Fluxo de vídeo

Uma das partes mais importantes da solução desenvolvida são as operações realizadas ao vídeo transmitido. Entre a captura, através do *smartphone* ou *tablet*, e visualização na televisão o vídeo transmitido, este sofre diversas transformações que serão analisadas em detalhe seguidamente.

Na seguinte figura 14 podemos visualizar o fluxo de vídeo desde a captura à receção. Seguidamente serão abordados em maior detalhe as suas diferentes fases.

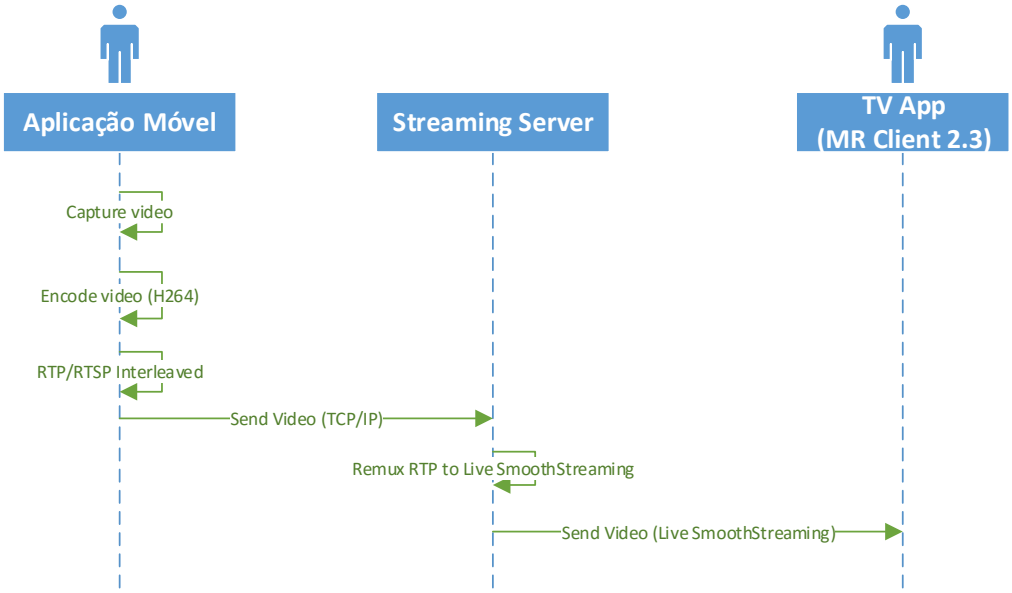

Figura 14 - Fluxo de vídeo entre aplicação, servidor de streaming e televisão

Na aplicação móvel o vídeo é capturado da câmara do dispositivo, sendo realizado o *encoding* do vídeo com codec H.264. O vídeo é seguidamente enviado via ligação TCP ao servidor de vídeo wowza no formato RTSP.

O servidor de vídeo Wowza está encarregue de receber a ligação RTSP enviada pela aplicação móvel, efetuar o *remuxing* para o formato Live Smooth Streaming, e disponibilizar um *streaming* para a aplicação no mediaroom aceder.

Consoante o dispositivo e a qualidade da ligação utilizada, são escolhidas diferentes resoluções:

- 352x288 – resolução mínima suportada, utilizada em ligações com baixa qualidade;
- 640x480 – resolução definida por defeito;
- 720x480 – resolução utilizada em ligações de boa qualidade.

Independentemente da resolução utilizada, fruto do ajuste correspondente à qualidade da ligação, o vídeo é sempre capturado e enviado a 25 *frames* por segundo.

6.4. Partilha de vídeo em direto oriundo das câmaras GoPro

A adição de compatibilidade à aplicação SeeWhatISee das câmaras GoPro foi o primeiro módulo a ser desenvolvido do decorrer do período do estágio. Estava inerente no plano de trabalhos a investigação de comunicação com este tipo de câmaras, implementação de camada lógica para suportar a comunicação, e camada gráfica tendo em conta as definições de UI/UX.

No plano inicial de desenvolvimentos o suporte às câmaras GoPro estava restringido à terceira geração, sendo que após a sua implementação, foi decidido adicionar o suporte aos mais recentes modelos da quarta geração. Este suporte obrigou à adaptação da estrutura previamente utilizada, de forma a torna-la genérica para os diferentes modelos suportados.

Este bloco de desenvolvimentos envolveu a captura, tratamento e envio de um *streaming* de vídeo.

6.4.1. Comunicação com câmara

A empresa por detrás do desenvolvimento das câmaras GoPro não disponibiliza qualquer tipo de documentação que permita o desenvolvimento de aplicações que possam interagir com as suas câmaras. Dado este facto foi necessário proceder à investigação da estrutura de comunicação utilizada na aplicação oficial para iOS.

Da seguinte análise foi possível concluir a sua estrutura de comunicação. Nas câmaras é possível ativar o seu *wifi*, iniciando assim o seu modo Access Point por defeito. A câmara disponibiliza endereços IP da classe A na gama 10.5.5.x, sendo o primeiro IP a ser atribuído é iniciado no 100, com as seguintes propriedades.

Default gateway 10.5.5.9

DNS Primário 8.8.8.8

DNS Secundário 4.4.4.4

Quando a aplicação é iniciada é efetuada uma ligação HTTP ao endereço da câmara. Se o porto estiver aberto conclui-se que o iPhone está ligado via wifi a uma GoPro. Após esta verificação é iniciada uma série de pedidos HTTP que permitem obter informações acerca da câmara, como estado da bateria ou definições de imagem utilizadas. Apesar de se manter uma estrutura idêntica de comunicação, os endereços diferem estre as duas gerações da câmara.

A seguinte figura 15 apresenta o principal fluxo de pedidos durante uma previsualização. A cada pedido HTTP é esperada a receção de uma resposta a confirmar a correta execução do pedido realizado.

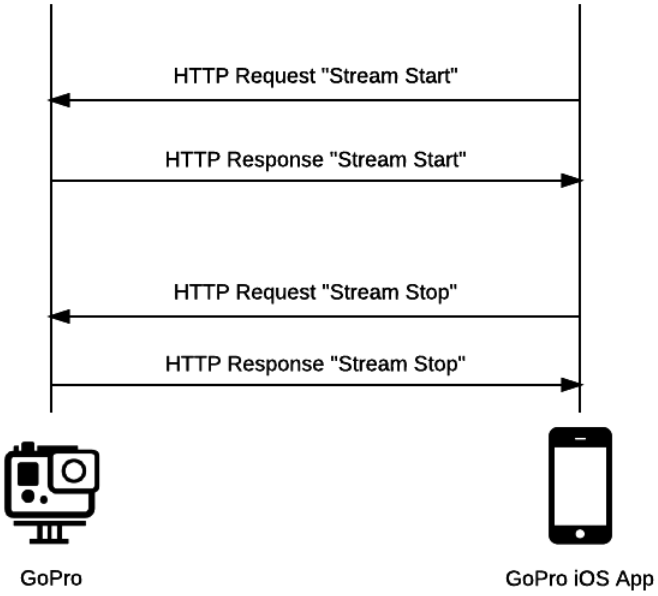

Figura 15 – Principal fluxo de pedidos durante uma previsualização

6.4.1.1. KeepAlive

Um dos problemas identificados durante os desenvolvimentos efetuados, foi a grande instabilidade da previsualização, após um curto espaço temporal do envio do comando de início do *stream*. Este problema levou a uma nova análise de comunicação entre a GoPro e a sua aplicação oficial iOS. Foi então aqui identificado um padrão, o envio e um comando repetidamente, com um intervalo de alguns segundos, desde o início e o fim da previsualização.

Na terceira geração das câmaras, o keepalive implementado, foi conseguindo com resultados enviando o comando de início de previsualização a cada três segundos.

i. http://10.5.5.9/camera/PV?t=<password>&p=%02

Para a quarta geração foi implementado um procedimento diferente da versão anterior, recorrendo ao padrão Wake-on-Lan (Wol). Este padrão permite que um dado computador seja ligado ou "acordado" enviando uma sequência de 6 octetos 0xFF seguida de 16 repetições do endereço MAC do nó que se pretende acordar, denominado por Magic Packet [47], podendo a sua estrutura ser visualizada na seguinte figura 16. Foi verificado e GoPro ignora o endereço MAC enviado, conseguindo o mesmo resultado do keepalive com um endereço MAC diferente do utilizado pela câmara.

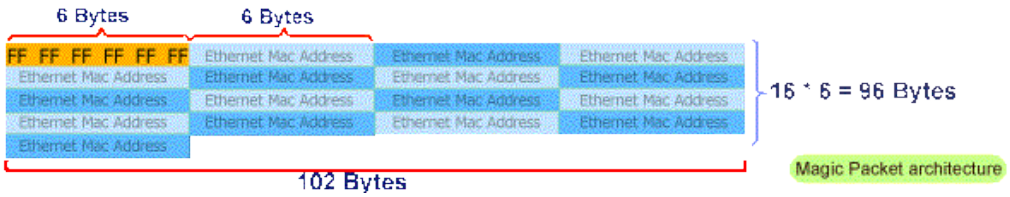

Figura 16 – Estrutura utilizada pelo Magic Packet [48]

6.4.2. Captura de previsualização

Para poder gerar um *streaming* de vídeo que possa ser enviado a um ou mais contactos é necessário primeiro proceder a sua captura. As câmaras GoPro de terceira e quarta geração geram a previsualização quando ativadas pelos comandos apresentados no capitulo anterior. Entre as duas gerações, terceira e quarta, existem grandes diferenças relativamente às especificações das previsualizações geradas.

A terceira geração, quando enviado o pedido para iniciar o stream, inicia a geração de um ficheiro no formato Http Live Streaming (HLS), gerando no cartão de memória o ficheiro amba.m3u8. Este ficheiro é possível de ser acedido na diretoria "/live/amba.m3u8". A previsualização gerada apresenta um atraso de 3 segundos em relação à imagem capturada inicialmente pela GoPro.

Na seguinte figura 17 podemos visualizar as características do *stream* gerado pela GoPro 3.

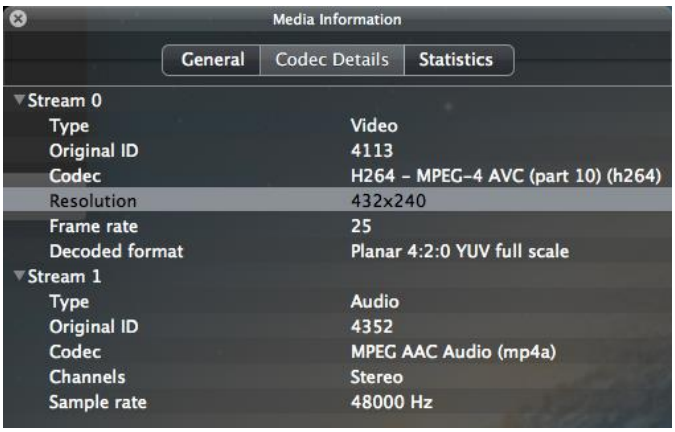

Figura 17 – Propriedades previsualização na GoPro 3

A quarta geração das câmaras GoPro difere da estrutura existente na geração anterior, deixando de estar disponível o ficheiro no formato HLS no cartão de memória da câmara. Em alternativa, esta geração recorre ao protocolo RTSP, permitindo assim a disponibilização de um *stream* com menor atraso e relação à imagem capturada, cerca de 1 segundo de diferença. Após enviar o pedido de início de geração da previsualização, o cliente deverá abrir o porto UDP 8554 no seu dispositivo, local este onde será recebido a previsualização gerada.

Na seguinte imagem 18 podemos visualizar as características do *stream* gerado pela GoPro 4.

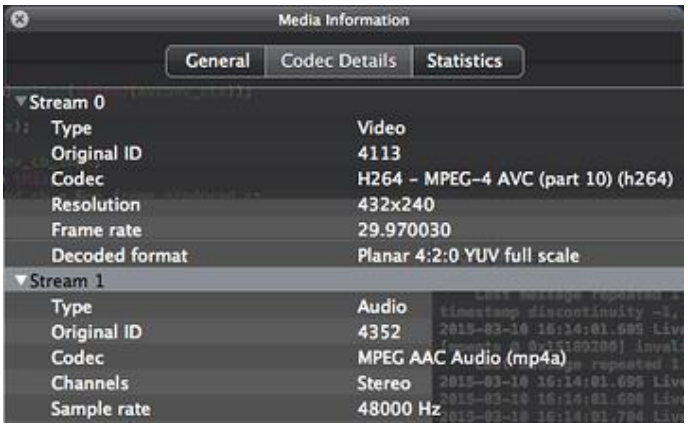

Figura 18 - Propriedades previsualização na GoPro 4

6.4.3. Exibição e *streaming* de vídeo

Pelas diferenças abordadas até agora entre as duas gerações das câmaras GoPro foi necessário implementar diferentes abordagens para cada câmara. Pela hierarquia cronologia pela qual foram lançadas, será primeiro efetuada a análise na terceira geração. Para efetuar a previsualização do *stream* capturado foi utilizado o elemento nativo da plataforma iOS, MPMoviePlayerController.

O *player* de vídeo do tipo MPMoviePlayerController gere a reprodução de um vídeo disponível na internet ou em ficheiro local. Na implementação efetuada este apenas tem de aceder ao endereço do ficheiro amba.m3u8 disponível no cartão de memória da GoPro. Para evitar redundâncias criadas pela proximidade entre a GoPro e iPhone o áudio foi desativado no *player*.

A biblioteca Ablib foi utilizada na GoPro3 para realizar a conversão entre formatos de vídeo, passando de HLS a RTSP. Nesta conversão foi realizada a cópia ao *streaming* de vídeo sem alterações e ajustado o *bitrate* de áudio para 44100Hz. A mesma biblioteca realiza o envio o *streaming* convertido, via ligação TCP, para o endereço do serviço de *streaming* de vídeo wowza.

Todo o fluxo anteriormente explicado é representado na seguinte figura 19:

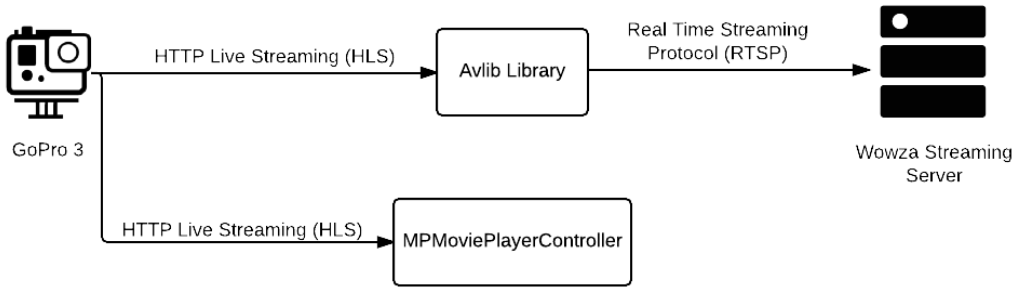

Figura 19 - Fluxo de vídeo desde a captura ao envio do vídeo ao servidor de streaming com câmara GoPro 3

Na quarta geração da GoPro foi necessário implementar uma estrutura diferente, recorrendo igualmente a biblioteca Avlib. Ao formato RTSP inicial foi ajustado o *bitrate* do *stream* de áudio.

Não existindo nenhum elemento nativo na plataforma iOS capaz de reproduzir um *streaming* no formato RTSP foi necessário delegar esta tarefa à biblioteca Avlib. Por cada segundo de vídeo recebido são geradas 15 *frames* com uma resolução de 512 de altura por 512 de largura. As 15 *frames* por segundo representam um valor inferior ao limite percetível pelo olho humano, podendo causar ligeiros problemas de fluidez a utilizados mais exigentes. A qualidade do vídeo enviado não é comprometida pois a baixa nas *frames* por segundo apenas tem impacto na previsualização realizada no iPhone.

Pela sua redundância, o áudio na previsualização foi desativado, não representando um novo elemento que era necessário processar. O *stream* RTSP enviado ao servidor Wowza foi sujeito à alteração do valor de *bitrate* da áudio para 44100Hz. Este foi o valor encontrado com melhores resultados na qualidade de áudio, após a realização de diferentes testes.

Na seguinte figura 20 podemos visualizar o fluxo de vídeo anteriormente descrito.

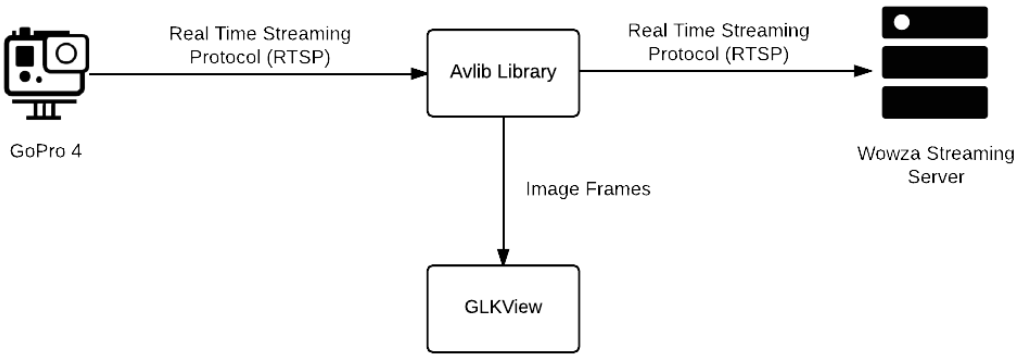

Figura 20 – Fluxo de vídeo desde a captura ao envio do vídeo ao servidor de streaming com câmara GoPro 4

6.4.5. UI/UX

De forma a possibilitar o suporte às câmaras GoPro na aplicação SeeWhatISee, foram introduzidos diferentes elementos na aplicação, sendo agora apresentados neste capítulo.

Na seguinte figura 21 podemos visualizar o menu principal da aplicação, onde foi introduzida a nova opção de partilha com câmaras GoPro. Ao selecionar é exibido ao utilizador *spinner* enquanto a aplicação procura uma câmara na rede.

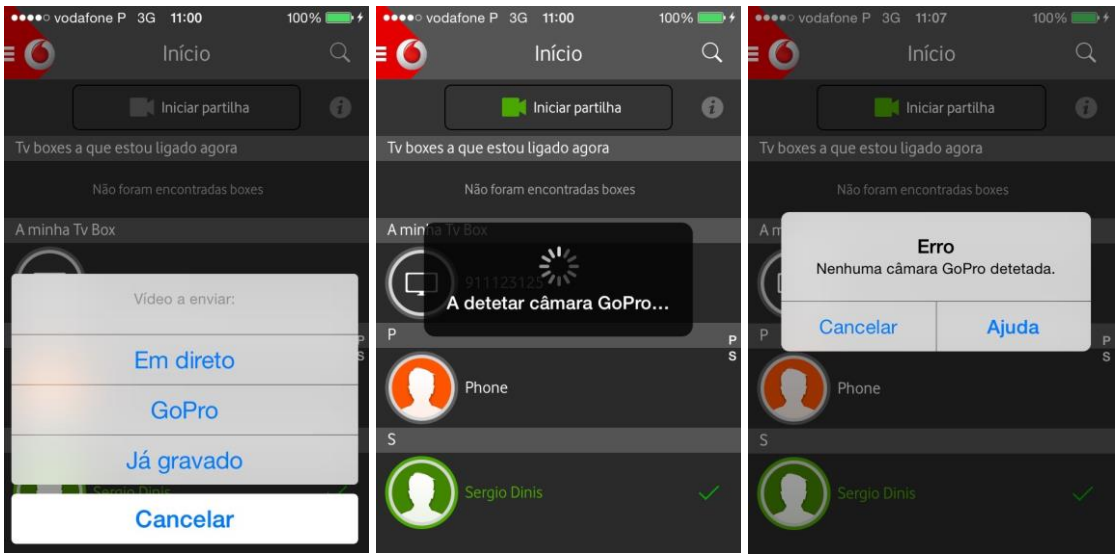

Figura 21 – Ecrã inicial da aplicação SeeWhatISee com integração do módulo GoPro

Quando não é possível detetar uma câmara GoPro na rede *wifi* à qual o iPhone está conectado é apresentada uma menagem de erro. Se o utilizador selecionar a opção de ajuda é exibido o seguinte ecrã, presente na figura 22. Este conjunto de ecrãs, com suporte a *swipe* horizontal, mostra os diferentes passos para efetuar uma partilha de vídeo a partir de uma GoPro.

Figura 22 - Ecrã de ajuda para efetuar partilhas com câmaras GoPro

Em diferentes momentos de uma partilha, a partir de uma câmara GoPro, vários fatores podem afetar a estabilidade da ligação à câmara. Na seguinte figura 23 podemos visualizar um conjunto de alertas com informações acerca do estado da ligação ou nível de bateria restante na câmara.

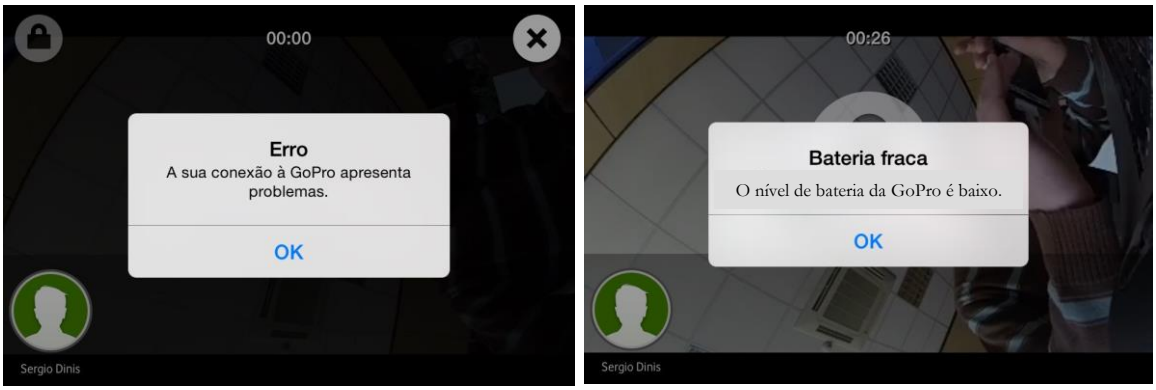

Figura 23 – Ecrã de alerta durante uma partilha na aplicação SeeWhatISee

A pensar na verdadeira essência inerente a uma câmara GoPro, foi implementado o modo de bloqueio, que permite o bloqueio do ecrã da aplicação. Desta forma o iPhone poderá ser colocado com segurança, sem receios de toques indevidos, não terminando a partilha em curso. Para desbloquear a aplicação, o utilizador necessita de realizar swipe horizontalmente, da esquerda para a direita. Este modo é apresentado na seguinte figura 24.

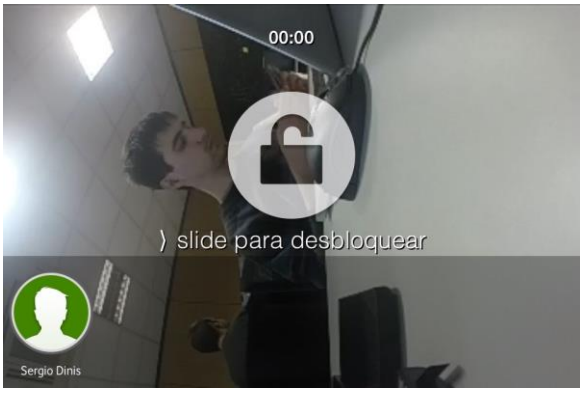

Figura 24 – ecrã de bloqueio durante uma partilha com a câmara GoPro

6.5. Efeitos de vídeo em direto

O desenvolvimento do módulo de efeitos de vídeo em direto, a adicionar à aplicação SeeWhatISee, foi o segundo a ser realizado durante o período de estágio. Antes de iniciar os trabalhos de implementação existiu uma análise de tecnologias e plataformas de suporte a efeitos de vídeo em direto. Os resultados desta análise podem visualizados no capítulo 4.5 - análise de bibliotecas com suporte a efeitos de vídeo.

Os desenvolvimentos efetuados por mim, na aplicação iOS, foram efetuados em simultâneo com os desenvolvimentos realizados na aplicação para Android, tendo estes sido realizados por outro membro da equipa de televisão.

Tendo em vista a partilha de lógica e reutilização de código, foi adota uma estrutura desenvolvida na linguagem C++, partilhada entre as duas versões da aplicação, iOS e Android. Esta estrutura já antes tinha sido seguida nos desenvolvimentos efetuados com a GoPro, sendo analisada em detalhe no seguinte capítulo 7.1.1 - código partilhado.

A lógica multiplataforma foi desenvolvida por mim, tendo como objetivo o suporte à adição de efeitos de vídeo ao *streaming* processado. Para além da lógica multiplataforma foi necessário implementar a estrutura para integrar esta lógica com o código nativo de cada aplicação.

Resultante da análise às plataformas de edição de vídeo em direto, abordado no capítulo 3.5 - análise de bibliotecas com suporte a efeitos de vídeo, foi decidido efetuar os desenvolvimentos recorrendo a biblioteca GStreamer. Esta escolha foi baseada no seu suporte multiplataforma, existência de documentação, suporte a *plugins*, e vasto leque de efeitos pré-existentes. Seguidamente irá ser analisada a arquitetura desta biblioteca.

6.5.1. Estrutura da biblioteca

Fundada em 1999, a biblioteca multimédia Gstreamer permite a reprodução, criação e manipulação de principalmente áudio e vídeo. É possível utilizar esta biblioteca para efetuar a reprodução de áudio ou vídeo, editar conteúdo pré-existente, e gravar o conteúdo em novos formatos.

A estrutura seguida por esta biblioteca é baseada em pipelines, tornado o processo de compreensão do fluxo de dados mais simples. Disponibilizada ainda uma estrutura de suporte ao desenvolvimento de *plugins*, como podemos visualizar na seguinte figura 25, sendo que estes podem ser classificados em:

i. Manipulação de protocolos;
ii. Origem de conteúdo;
iii. Manipulação de formatos
iv. Codecs: *Decoding* e *Encoding*;
v. Filtros;
vi. Destino de conteúdo.

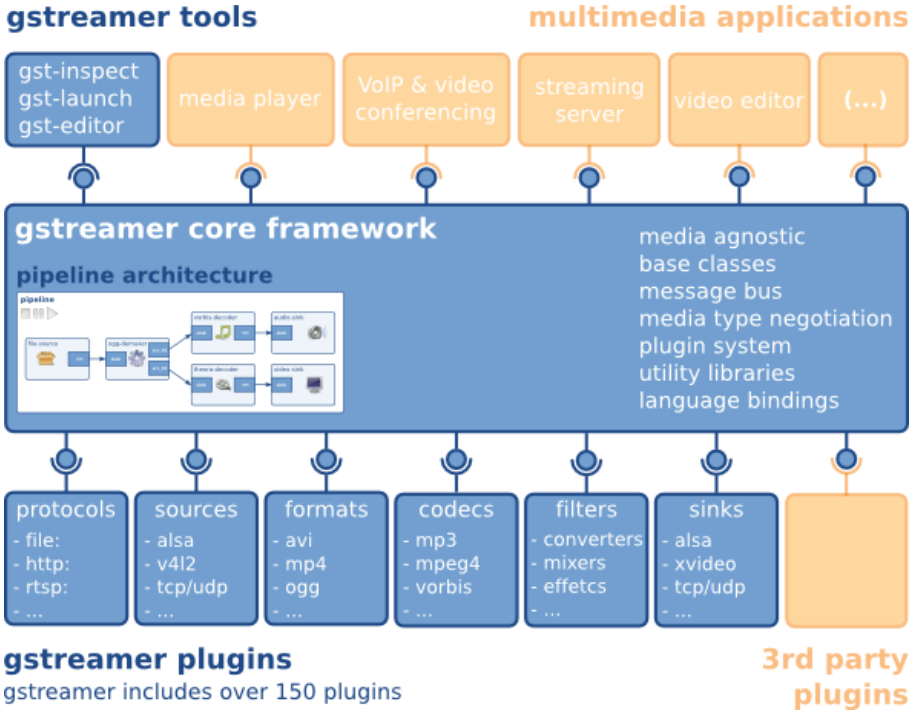

Figura 25 – Estrutura da biblioteca Gstreamer [49]

A biblioteca apresenta-se dividida em diferentes conjuntos, permitindo uma organização de elementos disponibilizados para desenvolvimento. Seguidamente são apresentados os diferentes conjuntos existes e breve contexto de cada um.

i. **gstreamer:** núcleo da biblioteca;
ii. **gst-plugins-base:** conjunto de elementos essenciais ao desenvolvimento;
iii. **gst-plugins-good**: conjunto de *plugins* de boa qualidade com licença LGPL;
iv. **gst-plugins-ugly:** conjunto de *plugins* de boa qualidade sem licença LGPL;
v. **gst-plugins-bad:** conjunto de *plugins* instáveis, que necessitam de melhoramentos;
vi. **gst-libav:** conjunto de *plugins* que fazem uso da biblioteca Avlib para realizar *decoding* e *encoding* de conteúdo.

Nos desenvolvimentos efetuados neste módulo de efeitos de vídeo foi utilizado um conjunto de *plugins* pré-existente. Tendo como preocupação o suporte em diferentes ambientes e linguagens, a biblioteca utiliza o GObject, uma camada de orientação a objetos para linguagem C.

6.5.2. Fundamentos

De forma a descrever os desenvolvimentos efetuados são apresentados neste subcapítulo alguns dos componentes básicos utilizados no Gstreamer.

6.5.2.1. Elements

Um elemento é o objeto mais importante no desenvolvimento utilizando o Gstreamer. Nos desenvolvimentos efetuados foram utilizados cadeias de elementos, ligados entre si, de forma a permitir o fluxo de dados.

Cada elemento tem uma função especifica, como ler vídeo de um endereço, converter o formato, aplicar um efeito, etc. Graças à junção de vários elementos, é possível criar uma pipeline, permitindo realizar diferentes tarefas.

6.5.2.2. Pads

Os pads são a entrada e saída de cada elemento, permitindo a ligação do fluxo de dados entre outros elementos. São utilizados para permitir a negociação do fluxo de dados entre os vários elementos. Nos desenvolvimentos realizados foi necessário utilizar pads para definir o formato de imagem na entrada e saída da pipeline.

A ligação entre dois elementos só é possível quando os caps de saída de um são suportados nos caps de entrada do outro. De forma a tornar possível a ligação entre elementos sem caps em comum foi necessário recorrer a um elemento de conversão de vídeo entre eles.

6.5.2.3. Bins

Um bin é um contentor para uma coleção de elementos. Este pode ser controlado da mesma forma que controlamos um elemento individual. O seu uso permite simplificar a complexidade inerente a uma atualização de estado, elemento a elemento, combinando um grupo de elementos num único elemento lógico.

Na seguinte figura 26 podemos visualizar um exemplo da sua utilização, em que o emento 1, 2 e 3, ligados entre si, passam a fazer parte de um único elemento representado pelo bin criado.

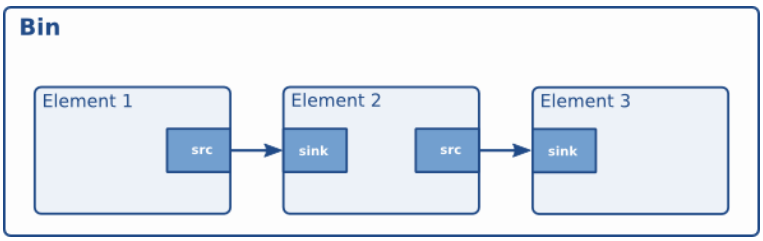

Figura 26 - Utilização de bin para agrupar elementos na pipeline [50]

Na aplicação desenvolvida, e de forma a simplificar a sua complexidade, apenas foi utilizado o Bin associado a toda a pipeline criada.

6.5.2.4. Pipeline

A pipeline representa um *top-level* Bin, gerindo a sincronização dos elementos que contém. Os seus diferentes estados podem ser:

i. Play – os elementos contidos na pipeline estão em reprodução;
ii. Pause – os elementos contidos na pipeline estão em pausa
iii. Stop - os elementos contidos na pipeline estão em parados.

6.5.2.5. Comunicação

A biblioteca disponibiliza diferentes mecânicos de comunicação e troca de informação entre a aplicação e a pipeline. Estes são agora apresentados na seguinte lista e representados visualmente na seguinte figura 27:

i. **Buffers:** permitem a passagem de dados de *streaming* entre os diferentes elementos presentes na pipeline;
ii. **Eventos:** objetos enviados entre elementos ou entre a aplicação e elementos. Nos desenvolvimentos efetuados foram utilizados, na maioria dos casos, para notificar alterações de estados na pipeline e/ou elementos individuas;

iii. **Mensagens:** objetos colocados pelos elementos no bus de comunicação da pipeline. São ser utilizados para transmitir informação entre os elementos e a aplicação, não permitindo no entanto a comunicação direta entre elementos.
iv. **Queries:** permitem o pedido de informação por parte da aplicação a um dado elemento e/ou o pedido de informação entre elementos.

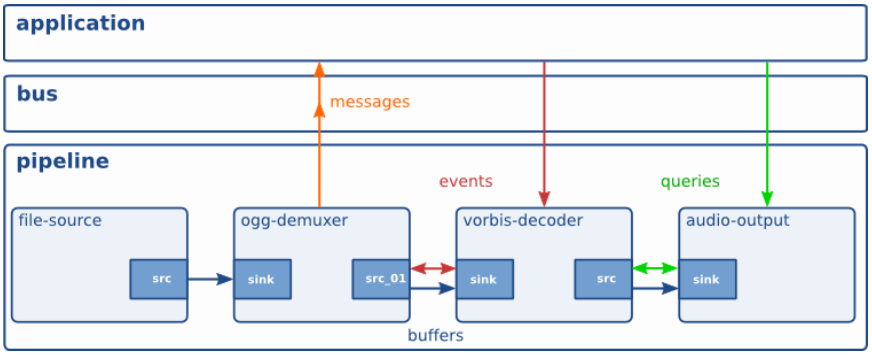

Figura 27 – Representação dos diferentes fluxos de comunicação na pipeline [51]

Estes mecanismos foram de grande importância nos desenvolvimentos efetuados, permitindo gerir diferentes acontecimentos durante o ciclo de vida da aplicação.

6.5.3. Efeitos de vídeo selecionados

Antes de avançar na integração da biblioteca na aplicação SeeWhatISee, foram realizadas prototipagens rápidas que visaram o conhecimento dos diferentes *plugins* existentes. Estas prototipagens permitiram realizar demonstrações internas, que levaram a seleção de uma lista de efeitos presentes nos grupos de *plugins* gst-plugins-good e gst-plugins-bad.

6.5.4. Pipeline desenvolvida

Para poder adicionar efeitos de vídeo em direto utilizando a biblioteca GStreamer foi necessário construir uma pipeline composta por diferentes elementos.

A captura de imagem a partir da câmara é intercetada e passada ao primeiro elemento da pipeline, o app source [52]. Este elemento disponibiliza toda a estrutura de suporte à injeção de buffers na pipeline, permitindo definir caps com as propriedades de imagem, *callbacks* para eventos entre outros parâmetros.

Seguidamente foi necessário utilizar um elemento do tipo videoconvert [53]. Este permite a negociação de um formato comum entre a saída do fluxo de dados de um elemento e a entrada o elemento seguinte. Desta forma garantimos suporte para qualquer formato de imagem obtido através da câmara, e qualquer formato de entrada no elemento de efeitos de vídeo.

O elemento seguinte é alterado dinamicamente, estando responsável pela aplicação do módulo de efeitos de vídeo selecionado pelo utilizador. Seguidamente foi necessário aplicar um novo elemento de videoconvert, permitindo negociar, com o elemento responsável pela aplicação de efeitos, o formato de saída utilizado.

De forma a manter à sua saída o mesmo formato de imagem que na entrada, foi necessário aplicar um novo elemento videoconvert, contendo as mesmas propriedades presentes nos caps do elemento app source.

Para permitir a divisão do fluxo de dados por dois elementos em simultâneo foi utilizado o elemento tee [54]. Um dos fluxos é enviado a um elemento do tipo app sink [55], permitindo o envio das imagens processadas com o efeito de vídeo de volta à aplicação, de forma a continuar o normal fluxo de vídeo da solução SeeWhatISee.

Para permitir que o utilizador tenha uma previsualização com o resultado da aplicação do efeito de vídeo, o fluxo de dados é também enviado a um elemento do tipo gl image sink [56]. Este elemento realiza o envio direto das imagens para uma vista de openGL.

A seguinte figura 28 representa o fluxo de dados entre os diferentes elementos, descrito anteriormente.

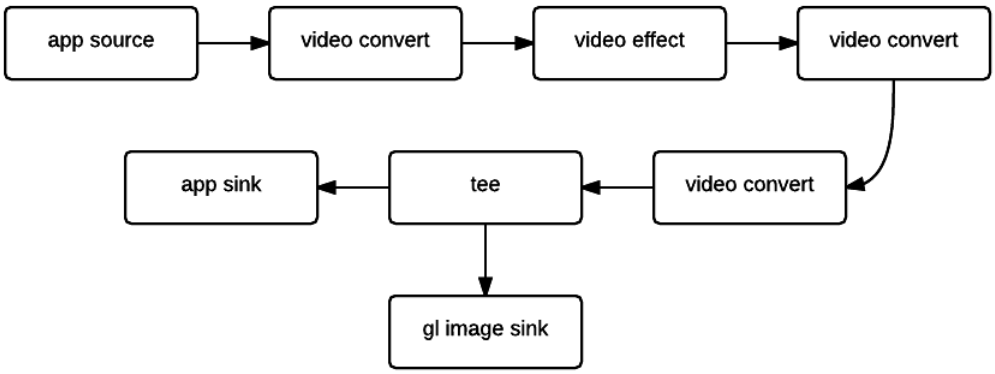

Figura 28 – Elementos presentes na pipeline desenvolvida para aplicação de efeitos de vídeo

6.5.5. UI/UX

Este tratou-se de um módulo com muito maior vertente lógica do que gráfica. Ainda assim, foi necessário introduzir na aplicação principal, no ecrã de partilha, um botão para ativar e desativar a aplicação de efeitos de vídeo, bem como o suporte ao *swipe* horizontal para alternar entre diferentes efeitos.

Na seguinte imagem 29 podemos visualizar o resultado do módulo de efeitos de vídeo com a aplicação do efeito *agingtv*.

Figura 29 – Aplicação SeeWhatISee com módulo de efeitos de vídeo – aplicação do efeito agingtv

6.6. Controlo de conteúdo em STBs via Apple Watch

O último módulo desenvolvido neste estágio curricular foi a aplicação de controlo de conteúdo em set-top boxes, utilizando no novo Apple Watch. Antes de serem iniciados os trabalhos de implementação foi realizada uma análise de concorrência, reunindo as principais aplicações móveis de controlo remoto existentes no mercado. Os resultados desta análise podem visualizados no capítulo 4.6 - análise de aplicações para controlo de set-top Boxes.

Ao contrário dos desenvolvimentos efetuados nos módulos anteriores, este corresponder ao desenvolvimento de uma aplicação de raiz, composta por aplicação para iOS e Apple Watch.

A aplicação para iPhone tem por objetivo realizar login e emparelhamento com uma set-top box, bem como servir de intermediária aos pedidos realizados pela aplicação no Apple Watch. No outro extremo, a aplicação no relógio tem por objetivo a exibição de informação acerca de programação no ar, videoclube e controlo remoto.

Na seguinte figura 30 podemos visualizar os diferentes fluxos de comunicação entre a aplicação no Apple Watch, a aplicação no iPhone e comunicação com servidor OTT e *midleware* Mediaroom. Os diferentes fluxos de comunicação são abordados nos três seguintes capítulos

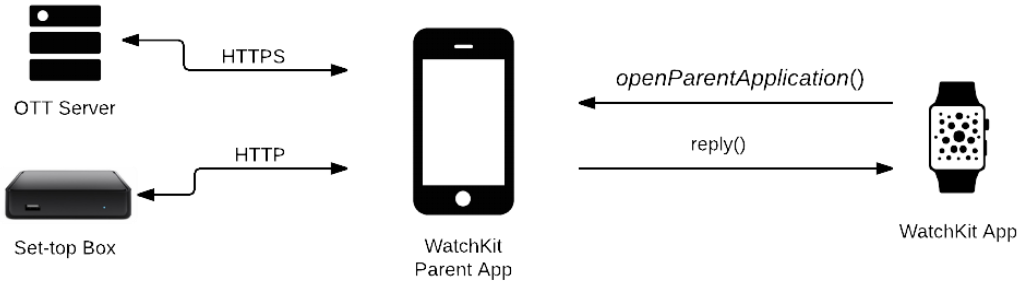

Figura 30 - Fluxos de comunicação entre os vários elementos da aplicação WatchKit

6.6.1. Comunicação com iOS App

O servidor OTT permite obter a informação dos diferentes canais, bem como a informação presente no videoclube. De forma a permitir a comunicação com set-top boxes existentes na rede *wifi* e OTT, foi necessário o estabelecimento de ligação à internet. A versão atual do SDK do WatchKit não disponibiliza qualquer meio para permitir a ligação à internet a partir do relógio. Foi por isso necessário analisar as diferentes formas de comunicação com a aplicação pai no iPhone, podendo ser encontrados mais detalhes no capítulo 7.2 - controlo de conteúdo em STBs via Apple Watch.

Dado que o aplicativo a desenvolver necessitou de uma estreita colaboração com a aplicação no iPhone, a fim de realizar os diferentes pedidos à internet, foi utilizado o método *openParentApplication:reply:*. Este permite o envio de pedidos à aplicação no iPhone, recebendo a resposta de forma síncrona.

Este método apenas permite o início a comunicação a partir do Apple Watch, não sendo permitido à aplicação pai no iPhone realizar a comunicação inversa. Se a aplicação no iPhone estiver suspensa ou terminada aquando o recebimento de um pedido, o sistema irá iniciar em background o recebimento do pedido, a fim de processar e enviar a resposta. Para o utilizador

final, este é um processo que se realiza invisivelmente, podendo estar a realizar qualquer outra tarefa em simultâneo no iPhone.

6.6.2. Comunicação com STB

Para realizar a comunicação com set-top boxes a correr o *midleware* Mediaroom foi utilizada a CompanionLib, uma biblioteca previamente desenvolvida pela WIT-Software. Esta biblioteca permite o envio de comandos ou eventos bem como a realização de emparelhamento com a set-top box selecionada.

O dispositivo comunica com a set-top box selecionada usando um POST HTTP composto com a informação necessária a cada ação requerida, recendo um status code que identifica o estado do pedido efetuado.

O emparelhamento, realizado na aplicação principal no iPhone, recorre a um serviço desenvolvido pela empresa que realiza a leitura e envio do código de emparelhamento presente set-top box. Com este código a aplicação pode pedir o emparelhamento daquele dispositivo, passando a poder interagir com a set-top box.

Na seguinte figura 31 podemos visualizar os diferentes ecrãs da aplicação a correr no iPhone durante o processo de emparelhamento com uma set-top box.

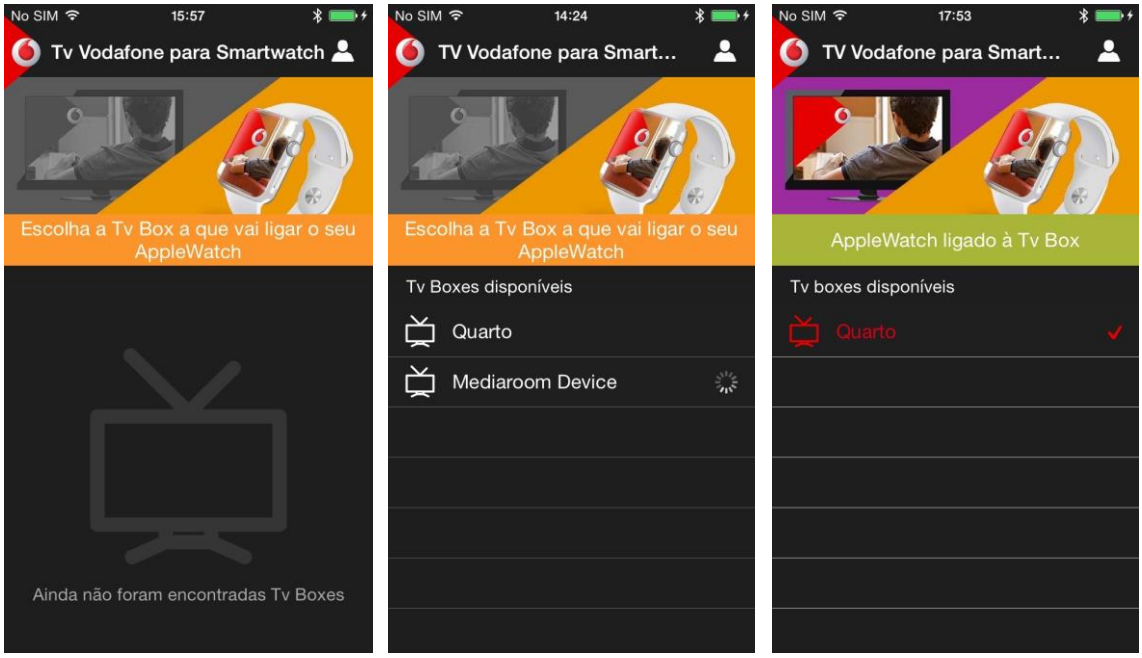

Figura 31 – Conjunto de ecrãs da aplicação principal durantes diferentes fases do processo de emparelhamento

O envio de comandos é utilizado para simular as ações presentes num normal controlo remoto, composto por botões como mudar de canal ou nível de volume. Recorre ao mapeamento da ação selecionada com a *key* correspondente no *midleware* Mediaroom.

Na seguinte figura 32 podemos visualizar os diferentes ecrãs de controlo remoto disponíveis no Apple Watch.

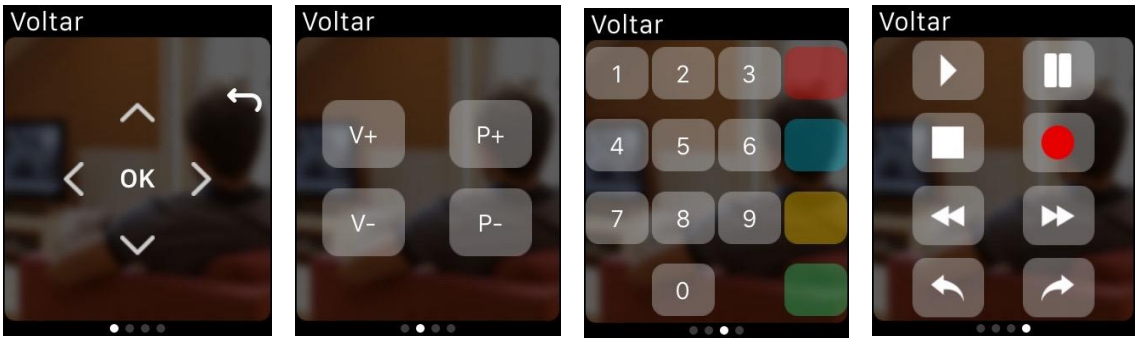

Figura 32 – Ecrãs de controlo remoto na aplicação no Apple Watch

De forma a obter a informação acerca do canal em reprodução na set-top box é utilizado o pedido de *info*, obtendo a resposta em XML, composta pela informação da posição do canal. Esta informação é mais tarde utilizada para pesquisar a programação do canal no OTT.

É ainda utilizado um método da CompanionLib que permite a abertura de uma página específica na set-top box, permitindo abrir a página de detalhes associada a determinado filme no videoclube.

6.6.3. Comunicação com OTT

Para preencher os ecrãs da aplicação no Apple Watch, correspondentes aos menus de programação e videoclube, é necessário obter informação presente no OTT da Vodafone. Todas as ações apresentadas neste subcapítulo necessitaram de ser implementadas de raiz.

Os pedidos realizados ao OTT utilizam um POST HTTPS para requisitar diferentes tipos de informação. O servidor envia a resposta em xml composta pelo estado do pedido e informação requerida.

A autenticação de um utilizador permite a obtenção de credenciais para realizar futuros pedidos ao OTT. É por isso o primeiro ecrã da aplicação, passo obrigatório para controlo da televisão com o relógio, podendo ser visualizado na seguinte figura 33.

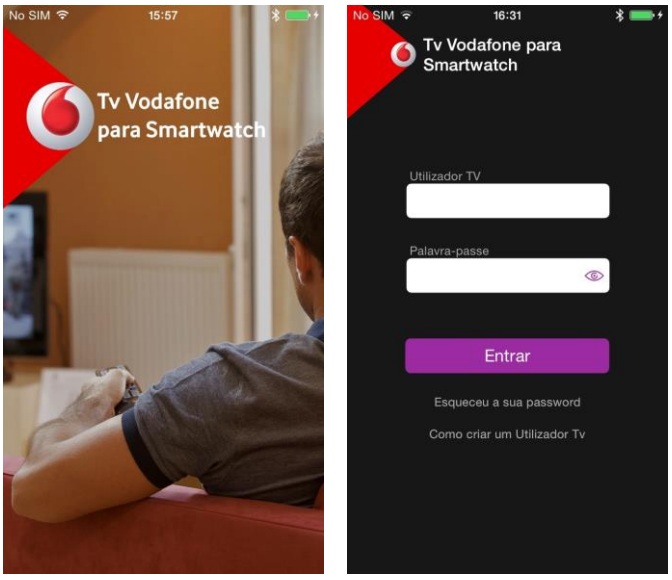

Figura 33 – Ecrã inicial de login da aplicação no iPhone

A informação do canal em reprodução, obtida em resposta ao pedido realizado à set-top box, é utilizada para realizar o pedido de programação ao OTT. Esta informação permite a composição dos ecrãs de programação "no ar", apresentados na seguinte imagem 34.

Figura 34 - Ecrãs de programação no ar na aplicação no Apple Watch

O videoclube apresentado na aplicação é composto por dois níveis de categorias, capas e detalhes dos filmes. Cada um destes ecrãs representam diferentes pedidos efetuados ao OTT, seguindo a estrutura de pedidos previamente apresentada.

Na seguinte imagem 35 podemos visualizar a composição dos diferentes ecrãs de videoclube na aplicação para Apple Watch.

Figura 35 - Ecrãs de videoclube na aplicação no Apple Watch

Capítulo 7
Desafios de Implementação

Neste capítulo são apresentados os mais relevantes detalhes e decisões de implementação realizados nos desenvolvimentos ao longo do estágio.

7.1. Estrutura

Este capítulo aborda decisões de implementação comuns a todos ou a mais que um dos módulos desenvolvidos.

7.1.1. Código partilhado

Os desenvolvimentos efetuados nos dois primeiros módulos, integração com câmaras GoPro e efeitos de vídeo na aplicação SeeWhatISee, seguiram uma estrutura pensada para possibilitar a partilha de código entre diferentes plataformas.

Foi assim possível criar lógica na linguagem c++, partilhada entre os sistemas operativos iOS e Android. Na plataforma iOS foi implementado um *wrapper* entre a camada de interface e camada de código partilhado. Apesar de tudo, em iOS, a camada de interface tem a capacidade de chamar diretamente um método c++ presente no módulo de código partilhado, podendo por isso ser evitada a implementação de uma camada de abstração entre a interface e código partilhado.

Na plataforma Android, implementada por outro membro da equipa, foi necessário desenvolver um *wrapper* entre a camada de interface, em linguagem java, e a camada de código partilhado. Utilizando a Java Native Interface (JNI) é possível a integração de código Java com código C e C++.

A seguinte imagem 36 apresenta a estrutura de código partilhado seguida nos desenvolvimentos efetuados.

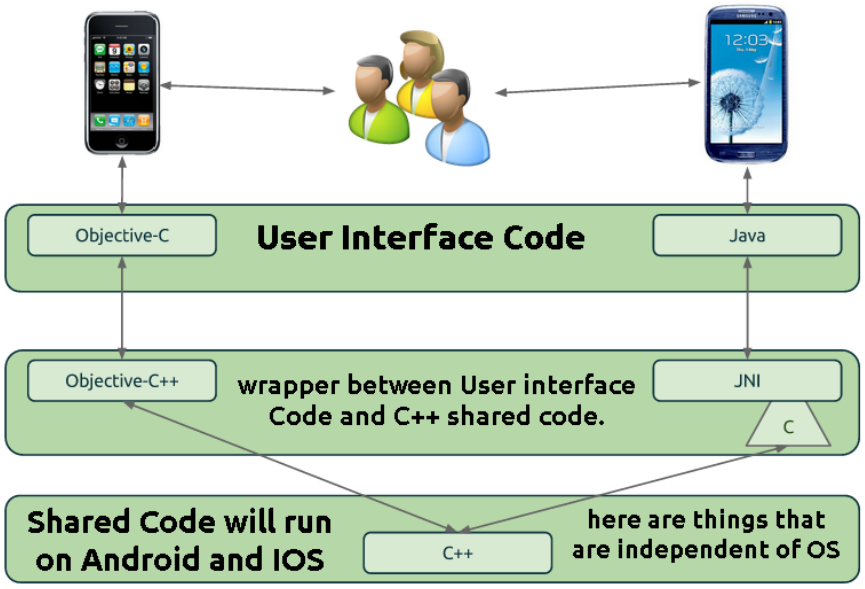

Figura 36 – Estrutura de código partilhado entre as plataformas iOS e Android [57]

7.1.2. CocoaPods

O uso de bibliotecas de terceiros é uma opção comum no desenvolvimento de *software*, poupando tempo de implementação ao usar lógica já implementada em bibliotecas externas. O uso de novas bibliotecas cria diferentes dependências no projeto ao qual é necessário resolver. Por vezes é normal encontrar dependências de diferentes versões da mesma biblioteca, tornando caótica a gestão tradicional de dependências.

Para a plataforma iOS, existe a ferramenta CocoaPods que tem vindo a torna-se padrão nos seus projetos, motivada pela dificuldade em importar e disponibilizar bibliotecas no Xcode [58]. Para suportar esta gestão o CocoaPods utiliza um arquivo, chamado Podfile, que lista todas as dependências e as informações necessárias ao projeto. A partir de aqui o CocoaPods fica responsável pela gestão de dependências de cada biblioteca, realizando a transferência, atualização ou remoção de dependências necessárias.

O uso desta ferramenta nos desenvolvimentos realizados facilitou todo o processo de importação de bibliotecas externas ao projeto, promovendo uma estrutura de dependências melhor organizada quando comparado com o processo tradicional.

7.2. Controlo de conteúdo em STBs via Apple Watch

Este capítulo aborda pormenores de implementação relativos ao terceiro módulo de desenvolvimentos realizado, controlo de conteúdo em STBs via Apple Watch.

7.2.1. Escolha de Comunicação entre Apple Watch e iPhone

O Apple Watch, na sua primeira versão do SDK, apresenta inúmeras lacunas de funcionalidades. Uma delas, a falta de ligação à internet diretamente a partir do relógio, causou algumas dificuldades ao desenvolvimento. Foi necessário por isso realizar uma análise às diferentes formas de conseguir enviar dados do iPhone para a aplicação no relógio.

7.2.1.1. OpenParentApp

O método *openParentApplication:reply:* permite a criação de um sistema de comunicação nos dois sentidos, podendo apenas ser iniciado a partir da extensão Watchkit no Apple Watch. Este método permite a partilha de um dicionário composto por objetos nativos do iOS.

Existe uma grande vantagem no uso deste método quando comparado com as alternativas existentes. A vantagem reside no fato de a aplicação no iPhone ter a capacidade de responder, mesmo que não esteja em execução no momento do pedido. Apesar de poder ser considerada uma vantagem pode igualmente ter os seus contras. Se a aplicação no iPhone não estiver ativa, e um pedido recebido do relógio pretender abri-la e alterar a sua interface, não irá conseguir, pois será apenas executada em *background*. Só em caso de a aplicação já estar previamente aberta é que se torna possível alterar conteúdo em *foreground*.

7.2.1.2. App Groups

Desde o lançamento das extensões na versão 8 do iOS foi introduzido o conceito de App Groups [59]. Tentando simplificar alguma complexidade no seu uso, é basicamente uma forma de partilhar dados entre múltiplas execuções no iPhone.

Uma das suas principais vantagens prende-se pela possibilidade de serialização de objetos, facilitando todo o processo de troca de dados. No seu inverso, apresenta algumas

desvantagens. Não é possível adicionar um *key-value-observer,* não tendo forma de notificar potenciais interessados das alterações efetuadas aos dados.

7.2.1.3. Darwin Notification Center

O Darwin Notification Center [60] é um mecanismo, disponível em iOS e OS X, que permite o envio de sinais entre processos. A biblioteca MMWormhole [61] apresenta-se como um *wrapper* em torno do Darwin Notification Center, permitindo a simplificação no seu uso.

A forma do seu uso assemelha-se ao App Groups, apresentado como vantagem a possibilidade de subscrição de alterações. Este mecanismo é ideal para a troca de informação entre duas aplicações, em execução em simultâneo. Caso a aplicação não esteja em execução torna-se impossível responder a um qualquer pedido.

7.2.1.4. Conclusão

Para o caso da aplicação desenvolvida, era uma mais-valia a possibilidade de realização de pedidos à internet sem necessidade de execução da aplicação no iPhone. Dado este fato foi escolhida a utilização método *openParentApplication*. A obrigatoriedade do uso de dicionários compostos por objetos nativos aumentou a complexidade na troca de informação, tendo neste campo a serialização de objetos possibilitado uma maior simplificação no processo.

7.2.2. Atualização de vistas

A aplicação desenvolvida para o Apple Watch necessita, em muitas das suas vistas, de informação externa à aplicação, acessível por comunicação à internet. Utilizando mecanismo *openParentApplication* foi possível disponibilizar informação acessível na internet à aplicação no relógio.

A sua execução não impede o normal fluxo do ciclo de vida da aplicação, apresentado no capítulo 6.1.2 - ciclo de vida, criando um problema de inconsistência na informação apresentada nas vistas do relógio. O método que finaliza as atualizações na vista *application:willFinishLaunchingWithOptions* poderá ser chamado antes que o pedido de informação tenha recebido resposta com dados para atualizar a vista, exibindo ao utilizador um ecrã incompleto, sem toda informação necessária.

De forma a contornar este problema foi implementado uma estrutura de vistas que exibe um indicador de atividade, baseado no projeto *JBWatchActivityIndicator* [62], até receber com sucesso os dados necessários à atualização da vista. Na seguinte figura 37 podemos visualizar os dois estados possíveis, correspondendo ao antes e depois da receção da informação.

Figura 37 - Diferentes estados das vistas na aplicação para Apple Watch

Capítulo 8
Controlo de qualidade

Para garantir o sucesso e cumprimento dos objetivos do projeto deverá existir um constante controlo de qualidade. Neste capítulo são apresentados os vários processos utilizados para garantir a qualidade final do *software* desenvolvido.

8.1. Static Analyzer

As aplicações desenvolvidas para a plataforma móvel da Apple iOS são compiladas recorrendo ao seu compilador por defeito, Apple LLVM Compiler. Este garante uma primeira validação do código através de uma análise semântica e sintática.

Em diferentes fases dos desenvolvimentos efetuados nas aplicações foram realizadas análises ao código com recurso ao static analyzer, fornecido nas ferramentas de desenvolvimento do Xcode. Este procede a uma análise de todo o código desenvolvido, permitindo identificar *memory-leaks*, variáveis não utilizadas, incumprimento de *guidelines* de código especificadas pela Apple e outros aspetos que auxiliam o desenvolvimento.

8.2. Testes de *software*

No final de cada módulo desenvolvido foram realizados testes ao *software* desenvolvido, permitindo avaliar o cumprimento de requisitos e o correto funcionamento das funcionalidades.

As diferentes ferramentas de testes ao *software* desenvolvido fornecidas pela aplicação Xcode, revelaram-se de grande importância, permitindo avaliar problemas de performance ou identificar elementos causadores de problemas de resposta ou até mesmo um *crash* da aplicação.

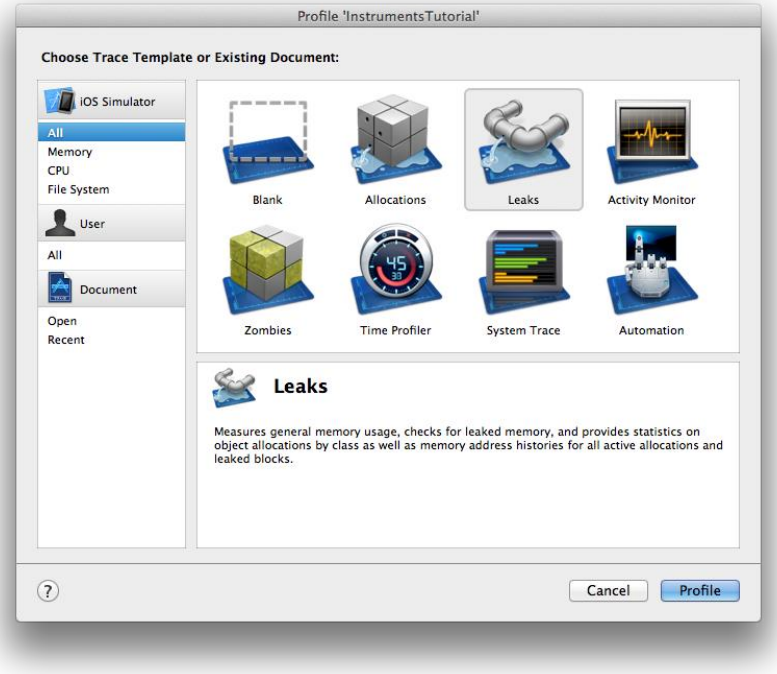

Figura 38 – Ferramentas de análise de performance e de teste fornecidas pelo Instruments

A anterior figura 38 apresenta algumas das várias ferramentas disponibilizadas pelo Instruments. Durante os desenvolvimentos foram principalmente utilizadas as seguintes:

i. **Allocations** – Análise de alocações de memória, permitindo identificar os diferentes blocos de memória alocados e desalocados ao longo da execução da aplicação;
ii. **Leaks** – Identificação de blocos de memória não desalocados ou alocação de memória abusiva. Este recurso foi largamente utilizado durante os desenvolvimentos efetuados no processamento de vídeo;
iii. **Time profiler** – Análise da utilização de tempo de processamento realizado pelas diferentes *threads*. Este recurso foi muito importante na identificação de causas em alguns problemas de performance identificados ao longo dos desenvolvimentos.

8.3. Continuous Integration

Quando um dado projeto partilha código com outros projetos/plataformas torna-se evidente a importância de garantir e validar todo o processo de integração.

A utilização de um sistema de controlo de versões garantiu um mecanismo que permite a integração dos desenvolvimentos dos vários componentes do projeto. Durante os desenvolvimentos este processo foi de grande relevância, permitindo a partilha de lógica entre diferentes elementos da equipa.

Foi também adotado um mecanismo de integração continua através de um *build server*, garantindo o processo de compilação do projeto assim como a geração das diferentes versões. O Jenkins [63] foi a plataforma escolhida para o *build server*, tendo a sua escolha sido baseada na sua popularidade e uso prévio em outros projetos pela equipa.

O script criado realiza *checkout* da versão mais recente no repositório do projeto, procedendo ao processo de compilação da sua nova versão. Sempre que exista uma falha que impeça a geração de uma nova versão do projeto, os responsáveis são notificados por email acerca da falha ocorrida.

Esta geração automática de versões revelou-se de grande importância para garantir a estabilidade geral do projeto e suas implicações em outros projetos com código partilhado. Foi também importante para permitir realizar a geração de novas versões das aplicações sem necessidade de todo o ambiente de desenvolvimento.

As diferentes versões geradas ao longo do estágio permitiram realizar várias demonstrações do estado geral dos desenvolvimentos do projeto.

8.4. Testes de internacionalização

A pensar nas diferentes nacionalidades dos possíveis clientes dos desenvolvimentos efetuados, existiu a preocupação de garantir a internacionalização das várias mensagens utilizadas na aplicação.

Para o garantir foi utilizado um mecanismo nativo do sistema operativo da Apple, iOS, que usa ficheiros para cada língua suportada com um conjunto de itens chave-valor. A escolha do idioma final apresentado pela aplicação é efetuado tendo por base a linguagem utilizada no

dispositivo, e em caso de falta de suporte para a linguagem utilizada é utilizada a linguagem padrão definida na aplicação.

A plataforma I18n revelou-se de extrema importância para a partilha de mensagens entre diferentes plataformas de desenvolvimento. Desta forma foram definidas um conjunto de chave-valor correspondendo às diferentes mensagens utilizadas nas diferentes plataformas da aplicação.

8.5. Demonstrações

Ao longo de todo o processo de desenvolvimento os diferentes módulos foram sujeitos a avaliações no seio da empresa. Para isso foram realizadas demonstrações a funcionalidades implementadas e a diferentes abordagens para chegar ao mesmo objetivo, permitindo recolher feedback que possibilitou um contínuo melhoramento dos desenvolvimentos efetuados.

Por vezes, fruto de interesse demostrado por clientes nos desenvolvimentos em curso, foram realizadas demonstrações a clientes finais do produto. Mais uma vez estas demonstrações permitiram obter informações que permitiram melhorar o produto final desenvolvido.

Capítulo 9
Conclusões

Este capítulo marca do término deste relatório e da conclusão de dez meses de estágio na empresa WIT-Software. À semelhança de capítulos anteriores, este está dividido em diferentes secções.

A primeira seção apresenta um *overview* de estágio curricular e conhecimentos adquiridos durante o seu período. A segunda seção refere os resultados alcançados com a realização do projeto de estágio, fruto dos diferentes desenvolvimentos efetuados. A terceira secção apresenta um plano de trabalhos futuro, reunindo informações acerca de possíveis melhoramentos a realizar nas aplicações desenvolvidas.

Por último, são apresentadas algumas considerações pessoais acerca do estágio curricular.

9.1. Estágio

O estágio curricular decorreu sem problemas de maior, tendo os desenvolvimentos sido realizados com acompanhamento contante por parte dos orientadores envolvidos, prevenindo assim potenciais falhas de orientação.

Durante o decorrer do estágio foram analisados diversos aspetos tecnológicos no desenvolvimento de *software*, bem como de integração com equipa de trabalho.

O projeto em que estive envolvido representou uma introdução à plataforma móvel de desenvolvimento da Apple, não tendo no seu início quaisquer conhecimentos na área. Foi possível, após dez meses de estágio, desenvolver competências que permitem o desenvolvimento de aplicações para a plataforma iOS.

Para além dos conhecimentos de desenvolvimento móvel adquiridos, foram igualmente desenvolvidas outras competências em diferentes áreas. A temática abordada no projeto permitiu desenvolver conhecimentos nas áreas de edição de vídeo e comunicação com set-top boxes Mediaroom.

O ambiente em contexto empresarial permitiu o primeiro contato com o processo de desenvolvimento de *software* para clientes finais.

9.2. Resultados alcançados

Durante o período de estágio foi possível o desenvolvimento de três diferentes aplicações móveis para a plataforma móvel iOS.

A primeira aplicação desenvolvida permite a integração das câmaras de ação GoPro, de terceira geração, com o produto SeeWhatISee. O interesse da Vodafone Portugal na prova de conceito levou a um pedido de alargamento de compatibilidade com os últimos modelos no mercado, de quarta geração.

A segunda aplicação desenvolvida permitiu adicionar um módulo de efeitos de vídeo em direto ao produto SeeWhatISee. Esta personalização de conteúdo em modo vídeo permite a criação de UGC com forte fator diferenciador e maior potencial de popularidade.

Por último, a terceira aplicação desenvolvida permite a comunicação com set-top boxes Mediaroom via Apple Watch. Este produto, recentemente introduzido no mercado, representou um desafio de implementação, tanto a nível de limitações de especificações, como a nível de documentação e exemplos de implementação existentes. Apesar de tudo, foi possível concluir com sucesso o desenvolvimento de uma aplicação que permite o controlo e visualização de informação presente em STBs.

Como complemento, foi igualmente possível desenvolver um módulo de videoclube no *smartwatch*, permitindo a navegação na galeria e a visualização de mais informações sobre o filme na televisão associada à STB emparelhada.

As três aplicações desenvolvidas representam uma prova de conceito realiza pela empresa para avaliar o resultado final e interesse por parte de potenciais clientes. O seu desenho seguiu uma estrutura modelar, permitindo a incorporação ou remoção de funcionalidades, fornecendo desta forma flexibilidade no desenvolvimento de novas soluções.

9.3. Trabalho futuro

Os desenvolvimentos efetuados de suporte às câmaras de ação GoPro são vítimas da inexistência de suporte ao desenvolvimento por parte dos seus criadores. À data da realização deste estágio curricular, a comunicação com as câmaras apresenta-se funcional, mas nada garante que mais tarde a empresa lance novas atualizações para as suas câmaras, e altere a forma de comunicação com as mesmas.

Um contato direto com a empresa criadora questionando acerca de suporte ao desenvolvimento poderia obter resultados, conseguindo suporte e garantias de comunicação otimizadas. Poderia igualmente informar sobre novas alterações na comunicação com as câmaras, fruto de novas atualizações de *firmware*.

A implementação de um sistema de configurações remoto, atualizado a cada inicialização da aplicação, permitiria a obtenção de configurações dos pedidos de comunicação com a GoPro. Esta funcionalidade permitia a adição de suporte a novos modelos sem obrigar ao desenvolvimento e atualização de uma nova versão da aplicação.

Os desenvolvimentos de suporte à quarta geração de câmaras GoPro e efeitos de vídeo em direto recorreram à utilização de componentes openGL. Devido à falta de experiência no desenvolvimento com a tecnologia, os elementos carecem de possíveis otimizações, tendo impacto no consumo de recursos realizados pela aplicação.

A aplicação de efeitos de vídeo desenvolvida, na sua pipeline, utiliza diversos elementos de conversão de formato de vídeo. Esta implementação favorece a compatibilidade entre sistemas, com suporte a diferentes formatos de imagem. Por outro lado representa um impacto acrescido no consumo de recursos, levando a problemas de performance em dispositivos com menores especificações. A utilização de um único formato de vídeo, desde o início ao final do fluxo de vídeo da pipeline, representaria uma otimização no consumo de recursos efetuados pela aplicação.

A Apple, na conferência para *developers* realizada no mês de setembro, deverá realizar a apresentação da versão do sistema operativo para o seu relógio, o WatchOS2 [64]. Esta nova versão irá representar um novo paradigma no desenvolvimento de aplicações para o Apple Watch, tornando as aplicações desenvolvidas por terceiros completamente independentes de um iPhone. Uma possível adaptação da estrutura implementada, para fazer uso das novas

funcionalidades disponíveis, tornaria a aplicação mais poderosa, com uma experiencia de utilização mais imersiva.

A nova possibilidade de realização de pedidos à internet a partir do relógio permitirá reduzir os tempos de espera entre ações. A implementação de um sistema de cache contribuirá igualmente para otimizar os tempos de resposta entre ecrãs.

9.4. Considerações finais

Para concluir este relatório, é agora tempo para apresentar as minhas opiniões e experiências acerca do estágio.

Estes dez meses de estágio representaram uma nova experiencia para mim, com contato direto com o mundo empresarial, cheio de desafios e lições a aprender. Durante todo o percurso tive liberdade, sendo incentivado até, para propor novas ideias que pudessem trazer valor para as aplicações desenvolvidas. Achei extremamente motivante este espaço criativo oferecido pela empresa, com a realização de *brainstormings* com a equipa, a fim de apresentar e discutir as novas propostas.

Fiquei igualmente fascinado pelo companheirismo e espirito de partilha de conhecimento presente na equipa de trabalho, permitindo-me crescer, a nível pessoal e profissional. Os resultados, agora alcançados neste estágio curricular, não teriam iguais se não tivesse inserido nesta fantástica equipa!

Bibliografia

[1] J. Heggestuen, "One In Every 5 People In The World Own A Smartphone, One In Every 17 Own A Tablet," 15 12 2013. [Online]. Available: http://www.businessinsider.com/smartphone-and-tablet-penetration-2013-10. [Acedido em 10 08 2015].

[2] "Are Young People Watching Less TV," 30 06 2015. [Online]. Available: http://www.marketingcharts.com/television/are-young-people-watching-less-tv-24817/. [Acedido em 10 08 2015].

[3] "Iterative and Incremental Development: A Brief History," [Online]. Available: http://www.computer.org/csdl/mags/co/2003/06/r6047-abs.html. [Acedido em 14 08 2015].

[4] "Waterfall model," [Online]. Available: http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/. [Acedido em 26 1 2015].

[5] "IBM Rational Unified Process," [Online]. Available: http://www-01.ibm.com/software/br/rational/. [Acedido em 26 1 2015].

[6] "Extreme programming," [Online]. Available: http://www.extremeprogramming.org/. [Acedido em 26 1 2015].

[7] "Slack," [Online]. Available: https://slack.com/. [Acedido em 14 08 2015].

[8] C. Williams, "How young viewers are abandoning television," [Online]. Available: http://www.telegraph.co.uk/finance/newsbysector/mediatechnologyandtelecoms/media/11146439/How-young-viewers-are-abandoning-television.html. [Acedido em 20 08 2015].

[9] "Netflix," [Online]. Available: https://www.netflix.com. [Acedido em 12 08 2015].

[10] "Hulu," [Online]. Available: http://www.hulu.com/. [Acedido em 12 8 2015].

[11] "Amazon prime," [Online]. Available: http://www.amazon.com/Amazon-Prime-One-Year-Membership/dp/B00DBYBNEE. [Acedido em 12 08 2015].

[12] "Youtube," [Online]. Available: http://youtube.com. [Acedido em 12 08 2015].

[13] "Twitch," [Online]. Available: www.twitch.tv. [Acedido em 12 08 2015].

[14] "Talking to Strangers: Millennials Trust People over Brands," [Online]. Available: http://resources.bazaarvoice.com/rs/bazaarvoice/images/201202_Millennials_whitepaper.pdf. [Acedido em 26 1 2015].

[15] "Chromecast," [Online]. Available: https://www.google.pt/chrome/devices/chromecast/. [Acedido em 12 08 2015].

[16] "Nexus Player," [Online]. Available: http://www.google.com/nexus/player/. [Acedido em 12 08 2015].

[17] P. Store. [Online]. Available: https://play.google.com/store?hl=pt_BR. [Acedido em 12 08 2015].

[18] "Apple TV," [Online]. Available: http://www.apple.com/pt/appletv/. [Acedido em 12 08 2015].

[19] "Roku," [Online]. Available: https://www.roku.com/. [Acedido em 12 08 2015].

[20] "Amazon Fire TV," [Online]. Available: http://www.amazon.com/Fire-TV-streaming-media-player/dp/B00CX5P8FC. [Acedido em 12 08 2015].

[21] "Amazon Fire Stick," [Online]. Available: http://www.amazon.com/Amazon-W87CUN-Fire-TV-Stick/dp/B00GDQ0RMG. [Acedido em 12 08 2015].

[22] "GoPro App," [Online]. Available: http://shop.gopro.com/EMEA/softwareandapp/gopro-app/GoPro-App.html. [Acedido em 12 08 2015].

[23] R. Dillet, "GoPro And Red Bull Should Merge And Become A Full-Time Media Company," 07 07 2015. [Online]. Available: http://techcrunch.com/2015/07/07/gopro-and-red-bull-should-merge-and-become-a-full-time-media-company/. [Acedido em 14 08 2015].

[24] "Magisto," [Online]. Available: http://en.wikipedia.org/wiki/Magisto#Technology. [Acedido em 26 1 2015].

[25] "Instagram," [Online]. Available: http://instagram.com/. [Acedido em 26 1 2015].

[26] "Vine," [Online]. Available: http://bambuser.com/. [Acedido em 26 1 2015].

[27] "Gstreamer plugins," [Online]. Available: http://gstreamer.freedesktop.org/documentation/plugins.html. [Acedido em 14 08 2015].

[28] "Banshee," [Online]. Available: http://banshee.fm/. [Acedido em 14 08 2015].

[29] "Rhythmbox," [Online]. Available: https://wiki.gnome.org/Apps/Rhythmbox. [Acedido em 14 08 2015].

[30] "FFmpeg," [Online]. Available: https://www.ffmpeg.org/. [Acedido em 14 08 2015].

[31] "Avlib," [Online]. Available: https://libav.org/. [Acedido em 14 08 2015].

[32] "Libopenshot," [Online]. Available: https://launchpad.net/libopenshot. [Acedido em 14 08 2015].

[33] "GPUImage," [Online]. Available: https://github.com/BradLarson/GPUImage. [Acedido em 14 08 2015].

[34] "CyberAgent/android-gpuimage," [Online]. Available: https://github.com/CyberAgent/android-gpuimage. [Acedido em 14 08 2015].

[35] "Meo Remote," [Online]. Available: https://play.google.com/store/apps/details?id=pt.ptinovacao.iad.meoremote&hl=pt_PT. [Acedido em 14 08 2015].

[36] "Iris Remote," [Online]. Available: http://www.nos.pt/particulares/ajuda/equipamentos-servicos/televisao/iris/aplicacoes/Pages/Iris-remote.aspx. [Acedido em 14 08 2015].

[37] "Tv Vodafone App," [Online]. Available: http://www.vodafone.pt/main/particulares/apps/tv-vodafone.html. [Acedido em 14 08 2015].

[38] "Peel Smart Remote," [Online]. Available: https://play.google.com/store/apps/details?id=tv.peel.app. [Acedido em 14 08 2015].

[39] "Model View Controller," [Online]. Available: http://theapplady.net/getting-started/how-an-ios-app-work-on-an-idevice/.

[40] "The App Life Cycle," [Online]. Available: https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html. [Acedido em 12 08 2015].

[41] "Concurrent Programming: APIs and Challenges," [Online]. Available: https://www.objc.io/issues/2-concurrency/concurrency-apis-and-pitfalls/. [Acedido em 12 08 2015].

[42] "Android Wear," [Online]. Available: https://www.android.com/wear/. [Acedido em 15 08 2015].

[43] "Developing for Apple Watch," [Online]. Available: https://developer.apple.com/library/prerelease/ios/documentation/General/Conceptual/WatchKitProgrammingGuide/DesigningaWatchKitApp.html#//apple_ref/doc/uid/TP40014969-CH3-SW1. [Acedido em 12 08 2015].

[44] S. Tibken, "Microsoft to sell Mediaroom IPTV business to Ericsson," CNet, 08 04 2013. [Online]. Available: http://www.cnet.com/news/microsoft-to-sell-mediaroom-iptv-business-to-ericsson/. [Acedido em 16 08 2014].

[45] D. Simmons, "Microsoft Mediaroom Buy Gives Ericsson Leadership in Fast-Growing IPTV Set-Top Box Middleware Biz," [Online]. Available: https://technology.ihs.com/431875/microsoft-mediaroom-buy-gives-ericsson-leadership-in-fast-growing-iptv-set-top-box-middleware-biz. [Acedido em 15 08 2015].

[46] "Live on TV," [Online]. Available: http://www.vodafone.pt/main/particulares/tv-net-voz/televisao/apps-tv/live-on-tv.html. [Acedido em 15 08 2015].

[47] ". O. L. (WOL), "http://www.dei.isep.ipp.pt/~andre/documentos/wol.html," [Online]. Available: http://www.dei.isep.ipp.pt/~andre/documentos/wol.html. [Acedido em 10 08 2015].

[48] H. Mosavi, "Wake On LAN (WOL)," 27 08 2007. [Online]. Available: http://www.codeproject.com/Articles/11469/Wake-On-LAN-WOL. [Acedido em 12 08 2015].

[49] "Chapter 1. What is GStreamer?," [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/chapter-gstreamer.html. [Acedido em 12 08 2015].

[50] "Chapter 6. Bins," [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/chapter-bins.html. [Acedido em 12 08 2015].

[51] "Communication Gstreamer," [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/section-intro-basics-communication.html. [Acedido em 12 08 2015].

[52] "appsrc: GStreamer Base Plugins," [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-base-libs/html/gst-plugins-base-libs-appsrc.html. [Acedido em 22 08 2015].

[53] "videoconvert - GStreamer," [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-base-plugins/html/gst-plugins-base-plugins-videoconvert.html. [Acedido em 22 08 2015].

[54] "tee: GStreamer Core Plugins," [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gstreamer-plugins/html/gstreamer-plugins-tee.html. [Acedido em 22 08 2015].

[55] "appsink: GStreamer Base Plugins," [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-base-libs/html/gst-plugins-base-libs-appsink.html. [Acedido em 22 08 2015].

[56] "glimagesink - GStreamer," [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-gl-plugins/html/gst-plugins-gl-plugins-glimagesink.html. [Acedido em 22 08 2015].

[57] "How to use the same C++ code for Android and iOS," [Online]. Available: http://stackoverflow.com/a/18334548. [Acedido em 12 08 2015].

[58] M. Buss, "Great CocoaPods Every Project Should Use," 25 01 2014. [Online]. Available: http://mikebuss.com/2014/01/25/great-cocoapods/. [Acedido em 18 08 2015].

[59] "App Extension Programming Guide - Apple Developer," [Online]. Available: https://developer.apple.com/library/prerelease/ios/documentation/General/Conceptual/ExtensibilityPG/ExtensionScenarios.html. [Acedido em 18 08 2015].

[60] "Darwin Notification Concepts - Apple Developer," [Online]. Available: https://developer.apple.com/library/mac/documentation/Darwin/Conceptual/MacOSXNotifcationOv/DarwinNotificationConcepts/DarwinNotificationConcepts.html. [Acedido em 14 08 2015].

[61] "MMWormhole," [Online]. Available: https://github.com/mutualmobile/MMWormhole. [Acedido em 18 08 2015].

[62] "JBWatchActivityIndicator," [Online]. Available: https://github.com/mikeswanson/JBWatchActivityIndicator. [Acedido em 18 08 2015].

[63] "Jenkings," [Online]. Available: https://jenkins-ci.org/. [Acedido em 18 08 2015].

[64] "WatchOS 2 Preview," [Online]. Available: http://www.apple.com/watchos-2-preview/. [Acedido em 18 08 2015].