Mestrado em Engenharia Informática

Estágio

Relatório Final

TV Everywhere: aplicação iOS para acesso ao conteúdo TV

Estagiário:

David Manuel Carvalho Rodrigues

dmcr@student.dei.uc.pt

Orientador:

Dr. Pedro Furtado

pnf@dei.uc.pt

Data: 3 de Julho de 2013

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

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Pólo II, Pinhal de Marrocos, 3030-290 Coimbra
Pinhal de Marrocos, 3030-290 Coimbra
☏ +351239790000
🖷 +351239701266
✉ info@dei.uc.pt

WIT-Software S.A.
EN1/IC2, Km 185,6, Banhos Secos, Santa Clara
3030-032 Coimbra
☏ +351239801030
🖷 +351239801039
✉ info@wit-software.com

Estagiário

David Manuel Carvalho Rodrigues
N° estudante: 2007183496
E-mail: dmcr@student.dei.uc.pt

Orientador da Faculdade, Departamento de Engenharia Informática

Pedro Furtado, PhD
E-mail: pnf@dei.uc.pt

Orientador da Empresa, WIT-Software S.A.

Eng.° Nuno Carvalho

E-mail: nuno.carvalho@wit-software.com

Agradecimentos

Aos meus pais a quem devo tudo que sou e alcancei até hoje, pela confiança que depositaram em mim e o esforço que realizaram para me oferecer a oportunidade de expandir os meus conhecimentos.

À minha namorada Ana Bento, por todo o amor e dedicação, por todos os conselhos, revisões e paciência, por ter estado sempre ao meu lado ao longo deste percurso.

Ao Nuno Carvalho, meu orientador na WIT-Software por toda a ajuda, suporte e confiança depositada em mim e por acreditar no sucesso deste projeto.

Ao meu orientador Pedro Furtado pela ajuda e suporte que me disponibilizou ao longo do estágio.

À empresa WIT-Software que me ofereceu a oportunidade de trabalhar num ambiente fantástico, me permitiu aplicar e expandir os meus conhecimentos técnicos e pelo conjunto de pessoas fenomenais que me permitiu conhecer.

Aos meus colegas e à minha equipa por toda a paciência, conselhos e suporte ao longo deste percurso.

Resumo

Nos dias de hoje vivemos a era da computação móvel com a proliferação em larga escala dos *smartphones* e dos *tablets* que se tornaram dispositivos presentes no nosso dia-a-dia. Criou-se uma nova experiência de utilização com os recentes ecrãs *multi-touch*, o acesso à informação tornou-se mais simples e rápido, surgiram os mercados de aplicações móveis que cada vez mais registam números astronómicos de downloads e vendas. O sector de televisão sofreu igualmente uma evolução com o surgimento do sector IPTV que ofereceu uma melhoria de serviço assim como um novo conjunto de funcionalidades que tem vindo a ser expandidas para um conjunto de novos dispositivos, como é o caso dos *smartphones* e dos *tablets* acompanhando assim a tendência tecnológica que temos vindo a registar.

Este estágio visa atingir a convergência plena entre estes dois ambientes, dispositivos móveis e plataformas IPTV, dando resposta a uma tendência gradual de usarmos estes dispositivos enquanto assistimos televisão, funcionando como um segundo ecrã onde frequentemente se pesquisa informação contextual. Tirando partido destes dispositivos é-nos possível ainda revolucionar a experiência de utilização através de uma nova interface acessível nos nossos dispositivos do dia-a-dia, a qualquer momento e em qualquer lugar.

O presente estágio visa completar este tipo de sistemas fornecendo o suporte para o sector móvel, acompanhando as tendências tecnológicas e oferecendo uma oferta apelativa, inovadora e completa aos operadores deste sector.

Palavras-Chave

"tv everywhere" "companion applications" "connected tv" "contextual tv" "video on demand" "electronic programming guide" "personal video recorder"

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

Acrónimo	Termo
ADK	Applications Development Kit
API	Application Programming Interface
BSS	Business support system
DRM	Digital rights management
DUID	Device Unique Identifier
EPG	Electronic program guide
HD	High-definition
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IPC	Inter-process communication
IPTV	Internet Protocol television
MOV	QuickTime File Format
MVC	Model–View–Controller
OSS	Operations support system
PSK	Pre-shared key
PVR	Personal video recorder
RCS	Rich Communications Services
REST	Representational State Transfer
SD	Standard-definition
SDK	Software development kit
SOAP	Simple Object Access Protocol
STB	Set-top box
SVN	Apache Subversion
TVE	TV Everywhere
UI	User Interface
URL	Uniform resource locator

UUID	Universally Unique Identifier
UX	User Experience
VOD	Video on Demand
XML	Extensible Markup Language

Glossário

Termo	Descrição
Agile	Conjunto de metodologias de desenvolvimento de software que visam uma rápida adaptação as mudanças e garantem a satisfação do cliente com entregas frequentes e continuas de software
Back-end	Parte da aplicação relativa ao servidor onde se encontra a camada lógica do negócio invocada pelo front-end.
Back-office	Interface de gestão e controlo de um conjunto de serviços relativos ao sistema por parte dos administradores, não visível pelo utilizador final
Branch	Ramificação paralela de desenvolvimento, criada com o intuito de preservar a integridade do código existente no *trunk*
Callback	Conjunto de código executável cuja referência é fornecida a outra entidade de modo a esta proceder à sua invocação a um dado momento conveniente
Commit	Persistência de eventuais alterações efectuadas no repositório
Event-driven	Paradigma de programação onde o fluxo de um programa é determinado por eventos ou mensagens
Event-loop	Mecanismo que recepciona e emite eventos ou mensagens num programa
Front-end	Interface gráfica do sistema direcionada para o utilizador final com o qual este interage diretamente
Gateway	Máquina intermediária geralmente destinada a interligar redes (ex: routers)
Jailbreak	Adicionar recursos extra a um dispositivo através da instalação de software não autorizado
Keychain	Sistema de armazenamento dos dispositivos Apple que permite o armazenamento de credenciais de forma segura
Local Play	Mecanismo de envio de fotografias e vídeos do dispositivo móvel para sua visualização na televisão
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 efectuadas num mesmo ficheiro
Middleware	Agente mediador entre o software e as demais aplicações visando fornecer formas de comunicações entre diferentes componentes
Operation	Objecto que encapsula toda a lógica e dados necessários à execução de uma tarefa de execução única. Representa uma macro-tarefa
Operation Queue	Instância responsável por proceder à execução das *operations* em threads secundárias
Product Owner	Elemento que define quais os itens que farão parte do *Product Backlog* e que os prioriza nas *Sprint Planning Meetings*
Proxy	Servidor intermediário responsável por reencaminhar as requisições de serviços ou arquivos efectuadas por um cliente a um determinado servidor
Sandbox	Mecanismo de segurança cujo objectivo é criar um ambiente separado e restrito às aplicações de modo a garantir um isolamento na sua execução
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
Subscriber	Representa um assinante do serviço de televisão do operador
Subscriber Group	Representa um conjunto de *subscribers* que possui uma determinada categorização. O objetivo destes conjuntos passa por permitir gerir o acesso ao conteúdo com base nestes grupos, ex.: disponibilização de canais *premium* ao conjunto de *subscribers* assinantes deste pacote
Subscription VOD	Consumidor (*subscriber*) paga um valor definido (geralmente mensal) por uma assinatura de contéudo *video on demand*
Thread	Fluxo de controlo sequencial isolado dentro de um processo
Thread-pool	Coleção de *threads* usadas para executar uma série de tarefas em background (ou segundo-plano)
Transactional VOD	Consumidor é cobrado pela visualização individual de cada conteúdo televisivo
Trunk	"Linha principal" de desenvolvimento num repositório SVN

VOD Package	Consumidor compra um conjunto de conteúdos fixo que corresponde por exemplo a uma série ou um conjunto de filmes
Waterfall	Modelo de desenvolvimento de software sequencial e estruturado
Wireframe	Protótipo (ou desenho básico) de uma interface que deverá retratar toda a arquitetura de informação e usabilidade do sistema desejada pelo cliente. No entanto, este desenho deve ser muito simples e resumido, devendo conter apenas o estritamente necessário

Tabela de anexos

ID	Descrição
A	Planeamento
B	Especificação de requisitos
C	Design e Arquitetura
D	Especificação de UI/UX
E	Benchmark de parsers JSON e XML
F	Resultados do benchmark de parsers XML e JSON
G	Análise de ferramentas de suporte ao negócio
H	Análise de ferramentas de controlo de qualidade
I	MediaProxy

Capítulo 1
Introdução

O presente estágio encontra-se inserido no âmbito da disciplina de estágio do Mestrado em Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, tendo decorrido na empresa WIT-Software, S.A.[1] sob orientação do professor Doutor Pedro Furtado e Engenheiro Nuno Carvalho.

1.1. A empresa

A WIT-Software é uma empresa especializada na criação de aplicações e serviços inovadores para os operadores de telecomunicações, operadores de televisão e internet móvel. Fundada em 2001 como *spin-off* da Universidade de Coimbra, estabeleceu desde logo uma parceria com a Vodafone Portugal, cliente com o qual continua a trabalhar nos dias de hoje. Ao longo dos anos a empresa tem registado um rápido e sustentado crescimento no sector que lhe permitiu marcar uma posição no mercado mundial, contando com um portfólio de clientes onde se destacam nomes como: Grupo Vodafone, Telefónica, TeliaSonera, Deutsche Telekom, Orange, Unitel, CenturyLink.

A empresa do ponto de vista organizacional encontra-se dividida por *Business Units* que se encontram especializadas num segmento específico de negócio da empresa como o são a área das telecomunicações, *Telco Business Unit*, a área das aplicações móveis, *Mobile Business Unit*, e a área dos conteúdos para operadores IPTV, *TV Business Unit*.

O sucesso da empresa constata-se não só pelo portfólio de clientes que possui mas igualmente pelo reconhecimento que lhe é concebido sendo considerada desde 2009 uma empresa PME Líder distinguida sempre com excelência, resultado dos seus resultados financeiros assim como da estabilidade financeira alcançada. A empresa encontra-se igualmente certificada nas normas de Qualidade (ISO 9001)[2], Ambiente (NP EN ISO 14001)[3] e Inovação (NP EN 4457)[4].

Recentemente foi considerada como uma das 15 empresas, no mercado das tecnologias, que mais cresceu no nosso país nos últimos 5 anos num estudo levado a cabo pela empresa de consultadoria Deloitte[5].

1.2. Enquadramento do estágio

A unidade de TV da WIT-Software, *TV Business Unit*, teve a sua criação no ano de 2009 com o intuito de dinamizar o sector de televisão recorrendo a uma componente fortemente social e disponibilizando um conjunto de serviços de forma simples e eficaz, tornando assim a experiência televisiva dos utilizadores mais completa e familiar através incorporação das novas tendências dos nossos dias.

Neste sentido foram iniciados desenvolvimentos que visassem criar uma solução completa e fortemente inovadora que permitisse completar a atual oferta dos operadores IPTV, tendo como objetivo direto a plataforma IPTV e os seus respetivos clientes diretos, as *set-top boxes*. Ao fim de 3 anos de desenvolvimentos foi criada uma

plataforma com fortes componentes de inovação tendo uma oferta ao nível de aplicações para a televisão, denominadas *TV Widgets*, uma loja completa de venda de conteúdos de vídeo, *VOD Storefront*, a convergência das comunicações assim como o desenvolvimento de aplicações contextuais para a televisão disponibilizando um complemento ao tradicional conteúdo televisivo. Foi igualmente desenvolvida uma plataforma de notificações para os operadores poderem proceder ao envio de notificações aos seus clientes, notificações como ofertas de conteúdo disponíveis, publicidade, entre outros *use cases*, encontrando-se neste momento em fase de testes para entrada em produção.

Com o desenvolvimento de toda a infraestrutura de suporte à atual oferta e tendo em consideração a predominância dos dispositivos móveis nos dias que correm, existia a necessidade de complementar a oferta com soluções *TV Everywhere* de modo a fornecer este conjunto de serviços neste tipo de plataformas, alcançando assim uma abrangência que passa pelas *set-top boxes*, smartphones, tablets e sem esquecer o sector web. Esta aposta dinamiza a atual oferta assim como comprova a aposta por parte da empresa nas últimas tecnologias mantendo sempre em mente um espírito de inovação nos produtos que desenvolve.

O presente estágio inseriu-se nos desenvolvimentos de soluções *TV Everywhere* da unidade de TV focado especificamente na plataforma iOS culminando no desenvolvimento do cliente *TV Everywhere* para iPhone e iPad.

A aposta neste tipo de soluções por parte da empresa apresentou-se como um caso de sucesso, o projeto está neste momento envolvido em vários concursos com vários operadores tendo já conquistado o concurso da Vodafone Alemanha e que culminará no lançamento da oferta *TV Everywhere* deste operador.

1.3. Motivação

O sector de IPTV tem vindo a registar um forte crescimento nos últimos anos, facto comprovado pelo aumento do número de *subscribers* destes serviços cujas estimativas preveem atingir a fasquia dos 75 milhões durante o ano de 2013 originando assim receitas na ordem dos 30 mil milhões de dólares. Tendo em conta o forte crescimento deste mercado a empresa decidiu apostar neste sector tendo como objectivo a disponibilização de soluções emergentes e inovadoras que complementem os tradicionais serviços das plataformas IPTV.

A aposta neste sector requereu a escolha de um *middleware* IPTV que possibilitasse a realização dos desenvolvimentos para este segmento, tendo essa escolha recaído sobre o *middleware* da Microsoft[6], Microsoft Mediaroom[7], pela forte adopção registada por parte dos grandes operadores como AT&T[8], Deutsche Telekom[9], Telefonica[10], Telus[11]. A importância deste *middleware* foi recentemente comprovada pela anunciada compra da unidade IPTV da Microsoft por parte da Ericsson, que a tornará a empresa líder no fornecimento deste tipo de soluções com um *market share* superior a 25%.

Os desenvolvimentos realizados possibilitaram a elaboração de um extenso e completo conjunto de serviços que constituem a atual oferta da empresa, estando no entanto, mais focada nos típicos clientes, ou seja, as *set-top boxes*. O forte crescimento de dispositivos móveis, smartphones e tablets, torna-os perfeitos candidatos a futuros

clientes deste tipo de plataformas. Alguns dados de Janeiro deste ano indicaram alguns factos interessantes:

- 50% dos utilizadores de tablets assistem a conteúdo vídeo nestes dispositivos de forma mensal, 24% dos quais paga pelo conteúdo a que assiste;
- 85% dos utilizadores de tablets e smartphones usam estes dispositivos enquanto assistem à televisão de forma mensal, havendo mesmo 40% que o fazem de forma diária.

Estes números demonstram a importância que estes dispositivos têm vindo a adquirir no quotidiano das pessoas sendo usados a qualquer momento e em qualquer lugar não só numa vertente mais profissional, mas também em momentos de lazer. Representam uma forma fácil de comunicação com os amigos e família, de rápido acesso a conteúdos de entretenimento bem como fontes de informação. Estes aparelhos desempenham um papel fundamental no dia-a-dia dos seus utilizadores, registando-se assim a necessidade de fornecer um conjunto de aplicações como forma de acompanhar a evolução que temos vindo a observar nos últimos anos, enquadrando-se totalmente neste panorama as plataformas IPTV.

Assim foram iniciados os desenvolvimentos de soluções *TV Everywhere* por parte da empresa com o objectivo de disponibilizar todo o conjunto de ofertas já existentes na televisão, agora em smartphones e tablets, consequentemente oferecendo aos utilizadores a comodidade de acesso a estes serviço a partir de qualquer lugar. O presente estágio insere-se nestes desenvolvimentos estando focado concretamente na implementação do cliente iOS, iPhone e iPad, que procura não só disponibilizar este tipo de funcionalidades de forma inovadora mas também procura atingir uma excelente experiência de utilização por parte dos seus consumidores finais. Em paralelo está a ser desenvolvido o cliente Android, também em âmbito de estágio, assim como o cliente Web permitindo desta forma abranger um conjunto significativo de plataformas igualmente bastante populares neste tipo de soluções.

1.4. Estrutura do documento

Este documento apresenta a seguinte estrutura:

- **Capítulo 2 (Apresentação do estágio)** – apresentação dos objetivos do estágio, os riscos associado, o planeamento e os resultados alcançados.
- **Capítulo 3 (Estado da Arte)** – apresentação e análise das soluções existentes no mercado dentro do âmbito do presente estágio.
- **Capítulo 4 (Desenho da aplicação)** – especificação dos requisitos da aplicação e consequente arquitetura que permite alcançar os objetivos traçados.
- **Capítulo 5 (Implementação)** – descrição do trabalho desenvolvido fundamentando as várias opções tomadas nos desenvolvimentos.
- **Capítulo 6 (Controlo de qualidade)** – processos realizados visando o controlo da qualidade do produto desenvolvido.
- **Capítulo 7 (Conclusões e trabalho futuro)** – considerações finais do trabalho desenvolvido com uma reflexão acerca da continuidade do projeto.

Capítulo 2
Apresentação do estágio

Neste capítulo será feita uma apresentação do estágio através do conjunto de objetivos pretendidos, a apresentação da equipa e da metodologia e finalmente os resultados alcançados pelo estágio.

2.1. Objectivos

O presente estágio visa estender a solução *TV Everywhere* da empresa com o desenvolvimento da aplicação iOS, iPhone e iPad, plataforma cujo peso no mercado é bastante significativo, cerca de 20.9%[12], tornando-a assim mais completa e apelativa para os operadores.

O objectivo da aplicação iOS prende-se por proporcionar a interação com os serviços de televisão da plataforma a partir de qualquer lugar, em qualquer momento e em qualquer dispositivo levando assim o mundo da televisão para lá dos limites das casas dos utilizadores. De forma complementar, permite igualmente completar a experiência de televisão por parte dos seus utilizadores ao permitir a interação com o conteúdo nos nossos dispositivos diários, podendo desta forma controlar a *set-top box* diretamente do nosso *smartphone* ou tablet sem necessitar recorrer ao tradicional comando.

Assim, a aplicação visa fornecer:

i. acesso ao conteúdo vídeo do operador, o denominado *Video-on-Demand* (VOD), possibilitando realizar a compra e a respetiva visualização do conteúdo diretamente nestes dispositivos;

ii. consultar a grelha de programação do operador, o denominado *Electronic Programming Guide* (EPG), permitindo realizar o agendamento de gravações de conteúdo a partir de qualquer local sem interagir com a *set-top box;*

iii. consultar a lista de gravações agendadas assim como as gravações concluídas respeitantes ao denominado *Personal Video Recorder* (PVR).

O cliente deverá igualmente suportar a comunicação com a *set-top box* de modo a disponibilizar a funcionalidade de controlo remoto permitindo desta forma complementar a experiência de utilização por parte dos utilizadores quando estão a assistir conteúdo diretamente na televisão.

O objetivo geral do projeto passa por completar a atual oferta *TV Everywhere* da empresa dotando-a de suporte para a plataforma iOS permitindo assim torná-la numa solução mais abrangente e em conformidade com as expectativas do mercado. O resultado final do estágio levará à integração da aplicação na oferta da empresa.

2.2. Riscos

Qualquer projeto de desenvolvimento de software está sujeito a riscos. Os riscos identificados são descritos de seguida.

2.2.1. Curva de aprendizagem

A familiarização com a tecnologia requerida pelo projeto representa sempre um risco no desenvolvimento do mesmo podendo levar a desvios no planeamento ou mesmo ao insucesso em algumas componentes do projeto. O desenvolvimento para este tipo de dispositivos representa igualmente um risco pelas limitações físicas associadas, poder de processamento, memória disponível, fonte de alimentação limitada, embora estejamos a assistir a uma estonteante evolução nestes dispositivos é necessário existir um pleno conhecimento da arquitetura e modo de funcionamento do sistema de modo conceber aplicações perfeitamente adaptadas e que retiram o melhor partido do dispositivo sobre o qual irão correr.

Este risco foi progressivamente minimizado ao longo do estágio com o permanente estudo da linguagem assim como da arquitetura destes dispositivos.

2.2.2. UI/UX

Tendo em consideração que o utilizador final do produto serão os clientes dos operadores, é de particular relevância ter em especial atenção o design gráfico assim como a respectiva usabilidade da aplicação, não só para garantir que o produto segue as linhas de design comuns às aplicações iOS mas igualmente por ser necessário vender o produto aos olhos dos consumidores, sendo essencial apresentar um produto cuidado e com boa apresentação gráfica. Ao longo dos desenvolvimentos houve a constante preocupação de analisar o mercado e conceber um conjunto de *wireframes* para mitigar este risco e conceber um produto apelativo do ponto de vista de design e experiência de utilização.

Outro aspecto a ter em conta prende-se pela possibilidade de customização de toda a aplicação com base nas pretensões do operador, existindo o risco de toda a interface poder ser reformulada. Durante os desenvolvimentos foram tomadas medidas para minimizar o impacto de eventuais reformulações da interface gráfica da aplicação.

2.2.3. Divergências e alterações de prioridade

Por vezes podem existir divergências entre as escolhas do estagiário e a visão da empresa, o que pode originar conflitos e afetar o percurso e os objetivos do estágio. Alterações de prioridade dos requisitos devido a demonstrações ou novos requisitos podem afetar igualmente a linha dos desenvolvimentos.

Este risco provou-se real tendo em conta uma alteração de requisitos da solução *TV Everywhere* que afetou diretamente o desenvolvimento do projeto tendo sido conduzida uma reformulação da arquitetura de modo a dar resposta ao novo conjunto de requisitos. Foram igualmente adicionados um conjunto de requisitos de modo a enriquecer a componente de VOD da aplicação o que levou a um reajustar do

planeamento do projeto. De modo a minimizar o impacto deste risco foi conduzida uma monitorização e um acompanhamento constantes de modo a garantir uma sincronização das necessidades da empresa e os objetivos do estágio.

2.3. Equipa

O presente estágio decorreu inserido na unidade de TV da empresa tendo sido desenvolvido o cliente iOS, iPhone e iPad, em equipa juntamente com o colega Pedro Silva. Os desenvolvimentos envolveram igualmente uma comunicação e colaboração com a equipa de *back-end* de modo a agilizar o processo de integração do cliente na plataforma. Os sucessivos desenvolvimentos foram acompanhados e validados por um elemento da equipa de Controlo de Qualidade, especialmente alocado aos projetos da unidade de TV e que se encontra responsável por assegurar os padrões de qualidade esperados e exigidos pela empresa WIT-Software. Ao longo dos desenvolvimentos houve igualmente um acompanhamento crucial por parte do Eng.º Nuno Carvalho, *Product Owner* do produto, gerindo e avaliando constantemente a evolução do projeto.

Na Figura 1 podemos visualizar os elementos da equipa que contribuíram para o presente estágio.

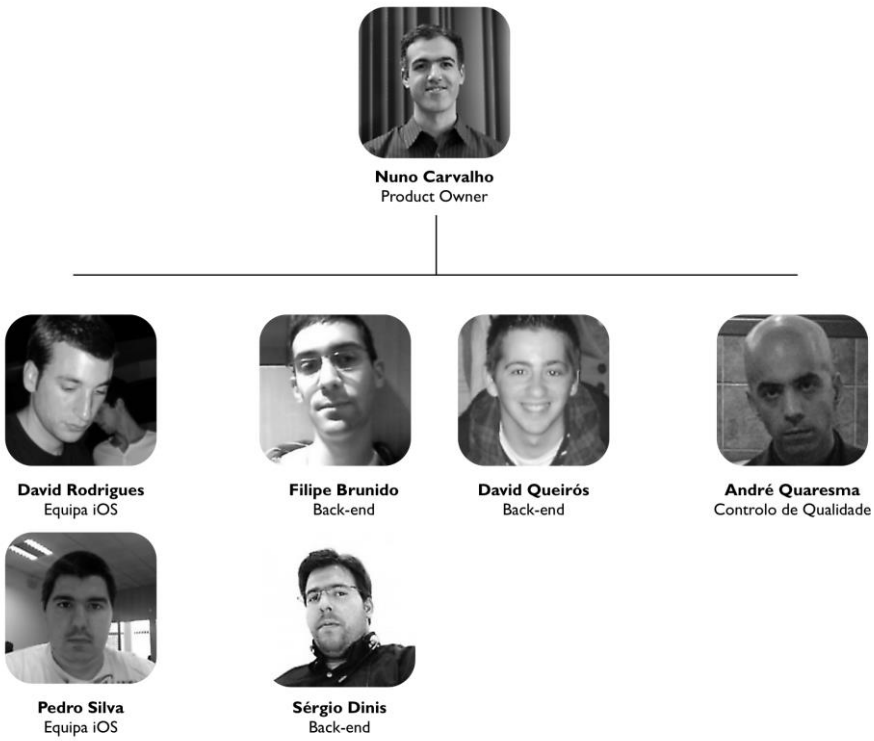

Figura 1 – Elementos da equipa com contributo no presente estágio

2.4. Metodologia

A empresa WIT-Software tem vindo a adoptar processos de desenvolvimento iterativos e incrementais baseados em metodologias *Agile* em detrimento do modelo *Waterfall*, modelo mais tradicional onde é feito um planeamento exaustivo e completo de todo o desenvolvimento do projeto durante as suas diferentes fases. A adopção destes modelos ágeis visa dar resposta às frequentes alterações de requisitos realizadas ao longo do projeto assim como permite igualmente um melhor acompanhamento da evolução do projeto permitindo realizar sucessivas avaliações durante o seu desenvolvimento e assim realizar possíveis ajustamentos necessários.

No desenvolvimento do projeto *TV Everywhere* foi adoptada uma metodologia *Agile* baseada em *Scrum* de modo a permitir um melhor acompanhamento e avaliação do decurso do estágio. A empresa realizou um pequeno *workshop* que permitiu uma primeira familiarização com esta metodologia de trabalho tendo sido feito posteriormente um estudo deste tipo de processos por parte do estagiário.

A metodologia usada com sucesso ao longo do estágio é caraterizada de seguida nas suas componentes chave.

- **Sprint** – um *sprint* consiste num período de tempo, 1 semana, onde a equipa está empenhada e concentrada no desenvolvimento de um conjunto de funcionalidades bem definido. No final do *sprint* é esperada a total concretização das funcionalidades definidas resultando num incremento funcional do produto final a atingir;
- **Product backlog** – representa o conjunto de requisitos a atingir no âmbito do projeto podendo existir uma definição de prioridades. Este conjunto de requisitos apresenta-se iterativo e incremental, em linha com a metodologia ágil adoptada sendo definido pelo *Product Owner* do projeto.
- **Sprint backlogs** – no inicio de cada *sprint*, é feito um levantamento de itens do *product backlog* a serem realizados. Para cada item é realizada uma análise e estimativa de duração permitindo assim especificar o conjunto de itens a constituir no presente *sprint* transitando os restantes para os seguintes.
- **Sprint plan meetings** – esta reunião visa analisar o *sprint* decorrido, se tudo correu como inicialmente planeado assim como analisar o resultado. Feita esta análise, é então especificado o *sprint backlog* do próximo *sprint*.
- **Daily meetings** – diariamente é realizada uma reunião com uma duração muito curta, entre 5-10 minutos, onde é feito um levantamento do que foi realizado até ao momento, obstáculos encontrados e se existem impedimentos no decurso do sprint tentando nesta eventualidade encontrar uma resolução desde logo. O *sprint backlog* é também atualizado permitindo ver o seu estado e respectiva evolução.

Na Figura 2 é possível visualizar o processo da metodologia usada ficando evidenciadas as componentes base que a suportam.

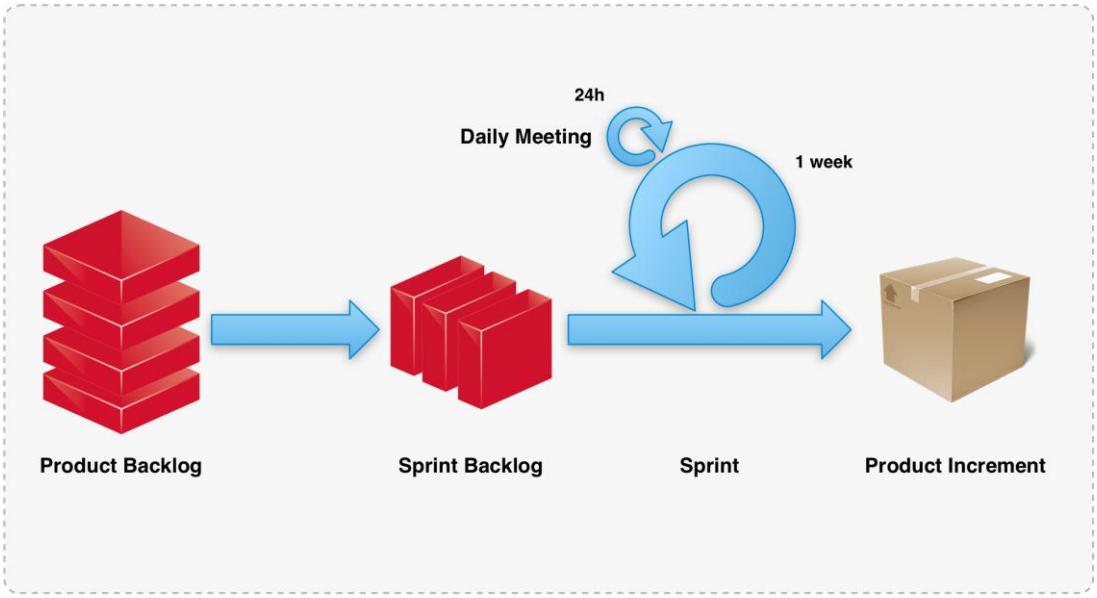

Figura 2 – Metodologia utilizada no desenvolvimento do projeto

2.5. Processos

No desenvolvimento do projeto foi usado o sistema de controlo de versões SVN que se revelou fundamental nos desenvolvimentos pelo trabalho desenvolvido em equipa. Todos os desenvolvimentos foram concebidos no *trunk* até atingir a versão 1.0 do produto, tendo depois sido iniciados ciclos de desenvolvimento em *branches* de modo a dotar o produto de novas funcionalidades terminando com o respetivo *merge* do *branch* para o *trunk*.

Nos desenvolvimentos foi adoptado um processo de *code review* contínuo por ambos os elementos da equipa de modo a proceder a uma análise das sucessivas alterações e desenvolvimentos evitando assim, dentro do possível, problemas de performance, comportamento incorreto, funcionalidades não-implementadas. Este processo de *code review* permitiu igualmente manter um profundo acompanhamento do projeto nas suas várias componentes bem como cimentar os conhecimentos técnicos na plataforma por ambos os elementos da equipa.

Periodicamente, de duas em duas semanas, foi igualmente conduzido um *code review* a mais alto nível com um elemento sénior da unidade de TV, tendo o objetivo passado por avaliar e discutir as sucessivas opções tomadas nos desenvolvimentos, o que se apresentou benéfico por fornecer uma visão mais externa sobre o projeto e permitiu por várias vezes fornecer uma nova perspectiva para os desenvolvimentos.

2.6. Planeamento

O planeamento disponível na Tabela 1 reflete a concepção dos vários módulos da aplicação, estando disponível no Anexo A o diagrama de Gantt que detalha com maior precisão as tarefas elaboradas ao longo do estágio.

Setembro	Documentação e análise de requisitos
Outubro	Arquitetura
Novembro	Módulo VOD
Dezembro	Módulo VOD; Módulo EPG
Janeiro	Módulo EPG; Relatório de estágio
Fevereiro	Módulo PVR; Autenticação
Março	Módulo Local Play (Suporte vídeo)
Abril	Reestruturação da arquitetura
Maio	Reestruturação da arquitetura; Extensão do módulo VOD
Junho	Extensão do módulo VOD; Relatório de estágio

Tabela 1 - Planeamento do estágio

2.7. Desvios

Qualquer projeto de software encontra-se suscetível a desvios relativamente ao que foi definido na sua concepção e especificação devido a alterações de prioridades, especificações incorretas entre outros factores.

Durante os desenvolvimentos houve algumas alterações relativamente ao inicialmente planeado sobretudo derivado à conquista do concurso do grupo Vodafone Alemanha, tendo sida conduzidas algumas análises técnicas pela empresa que resultaram em alterações significativas na plataforma e consequentemente nos clientes. Estas alterações na plataforma levaram a uma reestruturação de arquitetura da aplicação para dar resposta aos novos requisitos. Foi igualmente decidido dar um maior foco no módulo de VOD em detrimento da solução RCS para o sector da televisão, pela necessidade de investir continuamente no produto até ao seu lançamento pelo operador. Assim a solução RCS@TV foi adiada de forma temporária e não será passível de ser incluída no âmbito do estágio, devendo assim realizar-se um maior investimento no módulo de VOD que se apresenta como uma das componentes chave do projeto.

2.8. Resultados alcançados

Um estágio por natureza representa um caminho que visa alcançar um conjunto de objectivos específicos, objectivos esses que devem ser devidamente identificados e

definidos desde o seu início. Ao concluir este caminho, o estágio permitiu completar a solução *TV Everywhere* da empresa através de uma aplicação iOS:

1. que permite o acesso a conteúdo de vídeo provisionado pelo operador, *VOD*, possibilitando desde a sua navegação, pesquisa, compra e visualização, permitindo assim fornecer aos seus clientes uma oferta de conteúdo de fácil acesso;
2. que permite o acesso à grelha de programação do operador assim como a pesquisa de programas, permitindo o seu acesso neste tipo de dispositivos em qualquer lugar;
3. que procede ao agendamento de gravações e respetivo cancelamento a partir de qualquer local sem requerer a interação direta com a *set-top box* instalada na casa do cliente;
4. que possibilita o controlo da *set-top box* diretamente a partir destes dispositivos substituindo assim o tradicional controlo remoto;
5. que permite a visualização de conteúdo de vídeo e imagem armazenados no dispositivo diretamente na televisão recorrendo à ligação com a *set-top box*;
6. que possibilita aos operadores restringir as componentes da aplicação que poderão ser acedidas pelos clientes fornecendo-lhe controlo sobre um conjunto de estados, desde o tipo de utilizador, a localização do utilização assim como o atual perfil de rede.

A aplicação foi igualmente especificada e modelada para possibilitar um total redesenho por parte da sua interface gráfica sem com isso afectar todo o seu funcionamento fornecendo assim à empresa um determinado nível de flexibilidade no desenvolvimento de diferentes clientes para diferentes operadores.

Na Figura 3 é possível visualizar o resultado final alcançado para o módulo de VOD, apresentando a *homepage* desenvolvida na sua versão iPhone e iPad.

Figura 3 - Módulo VOD da aplicação na versão iPhone e iPad

Na Figura 4 é possível visualizar o módulo EPG apresentando a grelha de programação contínua na versão iPad e a programação por canal na versão iPhone.

Figura 4 - Módulo EPG da aplicação na versão iPhone e iPad

A Figura 5 apresenta o módulo de PVR apresentando a informação relativa aos programas gravados e agendados nas versões iPhone e iPad.

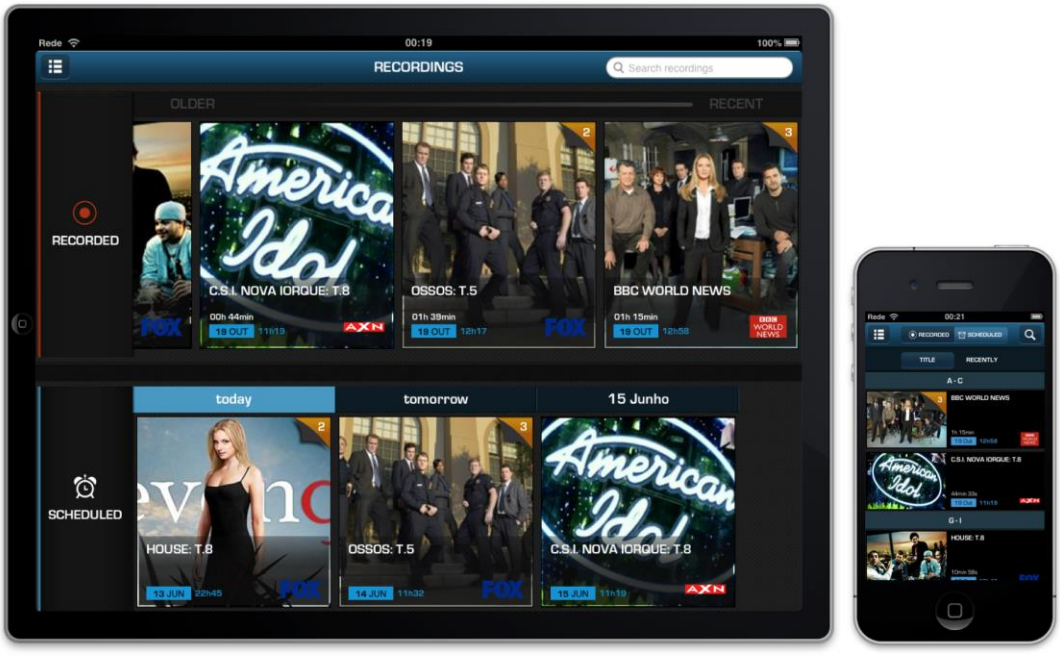

Figura 5 - Módulo PVR da aplicação na versão iPhone e iPad

Capítulo 3
Estado da Arte

O levantamento do estado da arte é uma etapa de elevada importância no desenvolvimento de um projeto por permitir avaliar a viabilidade do projeto através da análise de soluções existentes, identificando os pontos fortes e as falhas que estes possuem, assim como analisar o nível de saturação do mercado no sector em questão. É com base nesta análise que é possível determinar o nível de risco do projeto e respetiva probabilidade de sucesso.

O presente estágio insere-se no sector da televisão, mais concretamente no sector IPTV, sendo por isso efectuada uma análise acerca das soluções IPTV e aplicações móveis mais amplamente conhecidas e utilizadas, de forma a explorar e enriquecer os conhecimento sobre esta área e consequentemente melhorar o produto final. Neste capítulo é ainda possível visualizar uma análise das tecnologias relativas ao desenvolvimento do presente estágio.

3.1. Soluções IPTV

A definição de IPTV fornecida pela Alliance for Telecommunications Industry Solutions (ATIS)[13] e que a descreve na sua plenitude é apresentada de seguida.

> *"IPTV is defined as the secure and reliable delivery to subscribers of entertainment video and related services. These services may include, for example, Live TV, Video On Demand (VOD) and Interactive TV (iTV). These services are delivered across an access agnostic, packet switched network that employs the IP protocol to transport the audio, video and control signals. In contrast to video over the public Internet, with IPTV deployments, network security and performance are tightly managed to ensure a superior entertainment experience, resulting in a compelling business environment for content providers, advertisers and customers alike."*

Hoje em dia as plataformas IPTV fornecem o serviço de televisão de uma grande percentagem de casas tendo um impacto muito relevante neste sector pelo conjunto de funcionalidades que oferece ao operador para disponibilizar aos seus clientes. É assim de particular importância a escolha do *middleware* de suporte a este conjunto de serviços.

A atual plataforma, TV Everywhere, encontra-se desenvolvida e devidamente integrada com o Microsoft Mediaroom. A escolha desta solução foi realizada após a criação da unidade TV da empresa, tendo por base uma análise de mercado que visou identificar qual o *middleware* mais utilizado pelos grandes operadores. O presente estágio prevê a integração com a atual plataforma e respectivo *middleware* usado pela empresa, não abrangendo assim a escolha a este nível tendo, no entanto, sido realizada uma breve contextualização a este nível por parte do estagiário.

Nas secções seguintes é feita uma breve apresentação das soluções com maior expressão neste sector.

3.1.1. Microsoft Mediaroom

Microsoft Mediaroom apresenta-se com uma plataforma *end to end* da Microsoft para o sector de televisão. Conta com um conjunto de funcionalidades que incluem *content-protected Live TV*, *Video on demand* (VOD), *Personal video recorder* (PVR) e aplicações que correm no respectivo cliente associado a estes sistemas, *Set-top Box*. É uma plataforma proprietária e paga sendo utilizada por um grande conjunto de grandes operadores. A nível nacional a MEO[14] usa este *middleware* como base para a sua plataforma.

3.1.2. NDS

A empresa NDS[15] possui igualmente uma oferta IPTV *end to end* particionada por módulos. Tendo em conta os vários módulos, disponibiliza um conjunto de funcionalidades que vão desde *content-protected Live TV*, *Video on demand* (VOD), *Personal video recorder* (PVR), aplicações que correm na *Set-top Box* e assim como um cliente multi-ecrã para PC, *smartphones* e *tablets*. É uma plataforma proprietária e paga, recentemente adquirida pela Cisco, que possui uma expressividade considerável no leque de operadores do sector. A nível nacional a ZON[16] usa este *middleware* como base para a sua plataforma.

3.1.3. OpenTV

OpenTV[17] apresenta-se como uma oferta IPTV *end to end* que disponibiliza os serviços caraterísticos desta plataforma: *content-protected Live TV*, *Video on demand* (VOD), *Personal video recorder* (PVR). Disponibiliza também a possibilidade de desenvolvimento de aplicações para a STB. É uma plataforma proprietária e paga contando com um considerável conjunto de clientes a nível global.

3.2. Aplicações móveis IPTV

Uma das componentes de uma plataforma IPTV consiste nos dispositivos cliente que possibilitam o acesso à plataforma, e respetivo conjunto de serviços, que até à data era feito exclusivamente recorrendo às *Set-Top Boxes* instaladas nas casas dos clientes do operador. Estes dispositivos apresentam no entanto uma desvantagem, são dispositivos que não apresentam qualquer mobilidade podendo apenas ser usadas na casa do respectivo cliente. Existe assim a necessidade de soluções que permitam oferecer mobilidade assim como o rápido acesso aos serviços IPTV dando resposta à constante e estonteante evolução tecnológica que assistimos hoje em dia, nomeadamente, a massificação dos dispositivos móveis.

O presente estágio visa fornecer completar a solução *TV Everywhere* da empresa com o desenvolvimento dos clientes iOS, possibilitando desta forma tornar a oferta apelativa e enquadrada com a procura existente no sector sem nunca descurar o sentido de inovação característico da empresa. Nesse sentido foi realizado uma análise de mercado que permitisse identificar qual a oferta existente e permitisse comparar objectivamente o cliente *TV Everywhere* da empresa relativamente aos seus concorrentes.

No Anexo A encontra-se uma matriz comparativa onde é visível os pontos fortes de cada aplicação, sendo igualmente estabelecida uma comparação com o cliente *TV Everywhere* da empresa WIT-Software.

3.2.1. MEO GO!

O MEO GO![18] é uma aplicação da MEO que permite aos seus clientes usufruírem dos mais variados conteúdos oferecidos por esta operadora, em qualquer ecrã oferecendo assim uma verdadeira experiência multi-ecrã, e em qualquer lugar, dentro e fora de casa. A aplicação permite a visualização de um conjunto de canais de televisão, o acesso ao conteúdo do MEO Videoclube e finalmente aceder ao guia TV onde pode ser consultada a programação assim como agendar gravações de programas.

3.2.2. MEO Remote

O MEO Remote é uma aplicação da MEO que faculta ao cliente o controlo remoto da sua *Set-top Box* diretamente no *smartphone* e na *tablet*. Esta aplicação permite executar toda a lógica do tradicional comando diretamente na aplicação bastando para isso realizar o processo de emparelhamento com a STB. A aplicação permite ainda o acesso à programação televisiva, agendamento de gravações, partilhar o conteúdo que está a dar na televisão nas redes sociais Facebook e Twitter, partilha de fotos do *smartphone* e *tablet* diretamente na TV assim como pesquisar e agendar gravações de programas de uma forma remota.

3.2.3. ZON Remote

O ZON Remote[19] é uma aplicação da ZON que faculta ao cliente o controlo remoto da sua *Set-top Box* diretamente no *smartphone* e na *tablet*. A aplicação executa um emparelhamento com a STB do cliente passando a disponibilizar toda a lógica disponível no tradicional comando. A aplicação disponibiliza ainda o acesso à programação televisiva, agendamento de gravações e finalmente ao videoclube da ZON.

3.2.4. ZONline

O ZONline é a aplicação que a ZON fornece aos seus clientes de modo a disponibilizar o seu leque de serviços em qualquer local. Esta aplicação conta com a visualização em direto de um conjunto de canais, a consulta do guia de programação, agendamento de gravações para além do acesso ao videoclube da ZON.

3.2.5. Netflix

O Netflix[20] consiste numa plataforma online de acesso a filmes, séries e programas de TV, que permite o rápido e instantâneo acesso ao conteúdo recorrendo a técnicas de *streaming*. É um videoclube na linha do que os operadores oferecem mas apresenta algumas diferenças como o acesso total ao conteúdo com base numa mensalidade fixa deixando de existir o processo de compras das outras aplicações, possui um famoso sistema de recomendações com base nos *ratings* dos utilizadores e oferece a funcionalidade de pausar a reprodução de um conteúdo num dispositivo e terminar noutro dispositivo diferente oferecendo assim uma verdadeira experiência *multi-device*.

3.2.6. HULU

O HULU[21] é um concorrente direto do Netflix existindo um número bastante relevante de semelhanças nas ofertas de ambos. O HULU funciona igualmente com a oferta de todo o seu conteúdo tendo como contrapartida uma mensalidade fixa, recorrendo igualmente a técnicas de *streaming* para visualização do seu conteúdo. A nível de caraterísticas possui também um motor de recomendações baseado nas preferências do utilizador assim como oferece a possibilidade de pausar a reprodução de um conteúdo num dispositivo e terminar noutro dispositivo diferente oferecendo, também ele, assim uma verdadeira experiência *multi-device*.

3.2.7. AT&T U-verse Live TV

AT&T U-verse Live TV é a aplicação da AT&T que oferece aos seus clientes o serviço *Live TV*. A sua aposta é muito focada e objetiva oferecendo aos seus clientes um conjunto de canais a que podem assistir em direto e conteúdo *on demand*. Esta aplicação é em tudo semelhante às aplicações Sprint TV e T-Mobile TV, todas elas desenvolvidas pela empresa, MobiTV[22].

3.2.8. Conclusões

Analisando as soluções existentes é possível constatar que estamos perante um mercado em evolução existindo uma oferta limitada tendo em conta as funcionalidades possíveis de alcançar. É ainda possível verificar que os operadores com plataformas IPTV estão a apostar neste tipo de soluções gradualmente tendo em conta apenas uma pequena parte já possuir soluções deste tipo, mostrando que este é um mercado que não está neste momento saturado existindo assim a possibilidade de dar resposta à procura dos operadores.

O cliente *TV Everywhere* comparativamente com a sua concorrência apresenta suporte para as funcionalidades nucleares desta área tendo ainda funcionalidades que oferecem um incremento de valor ao produto. A única exceção prende-se pela componente de *Live TV* que será suportada no futuro a quando da integração com a plataforma do operador, esta opção passa pelos custos associados assim como não é totalmente viável integrar com as várias soluções disponíveis no mercado e utilizadas pelos operadores. Assim este requisito será devidamente cumprido a quando da integração na plataforma do operador.

Na Tabela 2 na página seguinte é possível visualizar um comparativo do conjunto de funcionalidades disponíveis nas várias aplicações móveis IPTV.

Aplicação	Live TV	VOD	EPG	PVR	Local Play	Controlo Remoto	Motor de Recomendações
MEO GO!	✓	✓	✓	✓	×	×	×
MEO Remote	✓	×	✓	✓	✓	✓	×
ZON Remote	×	✓	✓	✓	×	✓	×
ZONline	✓	✓	✓	✓	×	×	×
Netflix	×	✓	×	×	×	×	✓
HULU	×	✓	×	×	×	×	✓
AT&T U-Verse Live TV	✓	✓	×	×	×	×	×
Sprint TV	✓	✓	×	×	×	×	×
T-Mobile TV	✓	✓	×	×	×	×	×

Tabela 2 - Comparativo de clientes móveis IPTV

Capítulo 4
Desenho da Aplicação

Neste capítulo é apresentado o processo de desenho da aplicação iOS que requereu uma análise da solução atual, nomeadamente da plataforma, de modo a estruturar e desenhar a arquitetura da aplicação para completar com sucesso o processo de integração com a plataforma. Foi conduzido igualmente um processo de levantamento e estudo de requisitos para o cliente de modo a cumprir os objetivos do projeto.

De seguida é feita uma introdução à atual solução *TV Everywhere* de modo a fornecer uma contextualização do ambiente com o qual a aplicação integra, avançando depois para o conjunto de requisitos da aplicação concluindo com a apresentação da arquitetura definida para a aplicação iOS.

4.1. Solução TV Everywhere

O desenvolvimento do cliente de uma plataforma apresenta sempre desafios na sua concepção por requerer uma plena percepção da plataforma de modo a disponibilizar o leque de funcionalidades disponíveis permitindo a sua utilização em pleno como pretendido.

De seguida é feita uma breve apresentação da plataforma que suporta a solução *TV Everywhere* da empresa e que representa a base para os clientes TVE.

4.1.1. Visão geral da plataforma

A plataforma *TV Everywhere* tem como objetivo integrar com os sistemas IPTV do operador fornecendo depois uma interface agnóstica para os seus clientes possibilitando assim atingir um determinado nível de abstração destes sistemas. Com esta abstração o custo de integração de um novo *middleware* é reduzido significativamente por não afectar os vários clientes da plataforma garantindo a interoperabilidade entre sistemas.

Esta plataforma para além de integrar com o *middleware* IPTV e subsistemas do operador, fornecendo o conjunto de funcionalidades base destes sistemas, permite ainda complementar essa plataforma com um conjunto extra de funcionalidades tornado assim a plataforma geral mais rica e completa. Entre estas funcionalidades encontra-se por exemplo o motor de recomendações desenvolvido pela empresa que recorre a abordagens de *Content-based filtering* e *Collaborative filtering* permitindo acrescentar valor aos conteúdos do operador.

A plataforma apresenta-se assim como uma camada de nível superior que conduz o processo de integração com a plataforma do operador. Na Figura 6 é possível visualizar as relações entre os clientes e as plataformas.

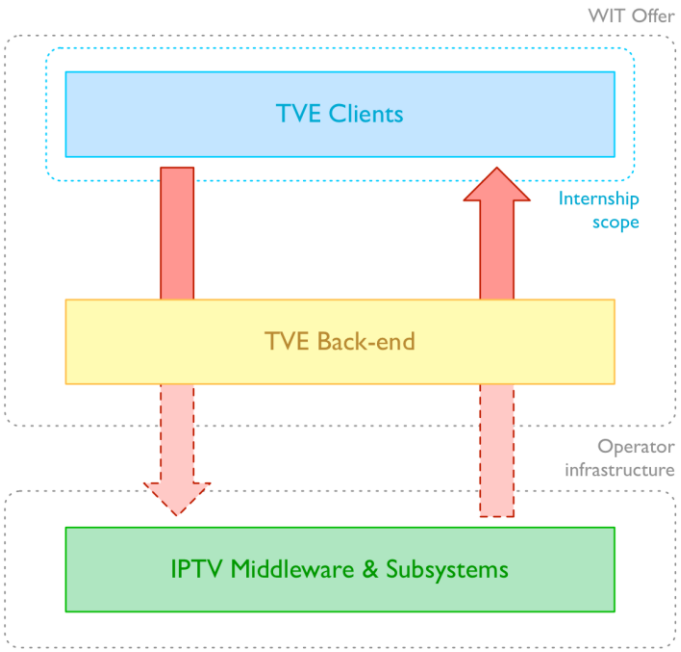

Figura 6 – Relações entre os clientes e as plataformas

Atualmente a plataforma encontra-se devidamente integrada com o *middleware* Microsoft Mediaroom para efeitos de ambiente de desenvolvimento assim como para efeitos de produto. De seguida é feita uma breve apresentação deste *middleware*.

4.1.2. Microsoft Mediaroom

O Microsoft Mediaroom representa a oferta na área de TV da Microsoft, recentemente adquirido pela Ericsson, apresentando-se como uma plataforma IPTV *end-to-end* líder a nível global que permite aos operadores de televisão oferecer um diverso conjunto de serviços de televisão ao seus clientes assinantes, os denominados *subscribers*.

É uma plataforma composta por uma série de servidores responsáveis por fornecer um conjunto de serviços de televisão disponibilizados aos seus clientes como o são as clássicas *set-top boxes,* consolas Xbox 360, computadores Windows e dispositivos com Windows Phone. É uma solução *out of the box* para operadores, sendo porventura um dos factores da sua elevada adopção pelo mercado.

Do ponto de vista funcional apresenta suporte a um leque de funcionalidades características das plataformas IPTV:

- Live TV
- Video on Demand (VOD)
- Electronic Program Guide (EPG)
- Personal Video Recordings (PVR)

O Mediaroom é constituído por toda uma infraestrutura própria, disponibilizando um conjunto de *back-offices* para proceder à sua gestão assim como um conjunto de web services que possibilitam integrações com outros sistemas. A sua última versão veio trazer alterações significativas, sendo agora disponibilizado sob a forma de um verdadeiro ambiente que recorre a virtualização para alcançar um isolamento dos seus vários serviços, visando garantir não só um aumento de estabilidade e escalabilidade da plataforma mas também simplificar o processo de instalação nas plataformas dos operadores.

A integração com outros sistemas, como é o caso da plataforma *TV Everywhere*, é possibilitada através de um conjunto de web services (SOAP) disponibilizados para o efeito e que se subdividem nas seguintes categorias:

- **Business Support Systems (BSS) Web Services** – estes serviços visam a integração com as várias operações de suporte ao negócio como a gestão de *subscribers*, questões de facturamento e as várias modalidades de pagamento de um determinado *asset*.
- **Operations Support Systems (OSS) Web Services** – estes serviços visam a disponibilização de uma API para que sistemas externos possam integrar e invocar um conjunto de operações da plataforma passando desde operações administrativas para gestão dos serviços televisivos da plataforma até ao conjunto de funcionalidades que os clientes dispõem como o processo de agendamento de gravações (PVR), consultar o guia de programação (EPG) entre outras funcionalidades.

A Microsoft disponibiliza ainda um *Applications Development Kit* (ADK) que visa fornecer um ambiente de desenvolvimento de aplicações para as STB da plataforma, possibilitando a expansão destes dispositivos com novas funcionalidades assim como integração com outros sistemas. Estas aplicações são denominadas *Presentation Framework Applications (PF Applications)* ou mais vulgarmente *widgets*.

4.1.3. Plataforma TV Everywhere

A plataforma *TV Everywhere* surge como uma evolução da plataforma criada inicialmente para os primeiros desenvolvimentos na área de TV, tendo sido dotada de suporte para este tipo de soluções.

A arquitetura da plataforma e relação com os demais sistemas pode ser visualizada na Figura 7.

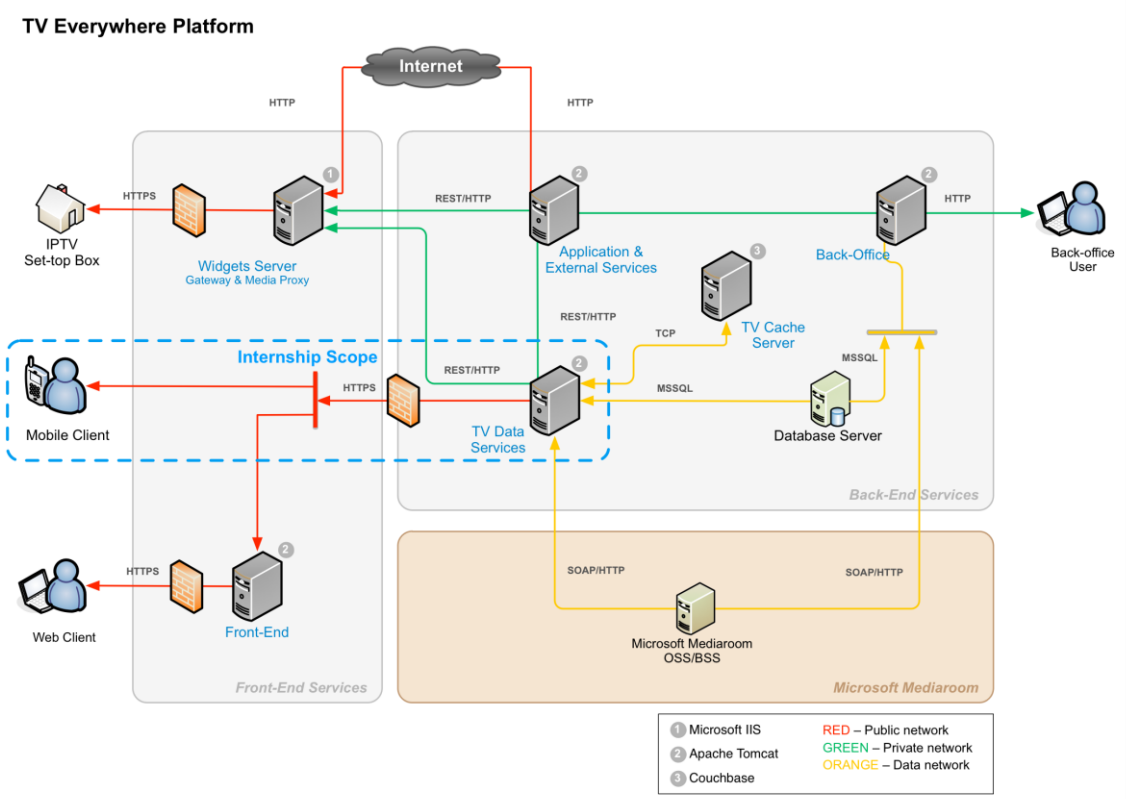

Figura 7 - Arquitetura da plataforma *TV Everywhere*

O núcleo da plataforma encontra-se no nó *TV Data Services*. É neste nó onde podemos encontrar a camada de acesso a dados que realiza a integração com o *middleware* do operador, no caso da plataforma atual integra com os web services OSS/BSS do Mediaroom. Esta camada é responsável pela disponibilização de um conjunto de interfaces, sob a forma de uma API REST, que conduz à invocação das respetivas operações de acesso a dados assim como operações de negócio na plataforma do operador independentemente do tipo de *middleware*.

Esta camada de acesso a dados apresentou um desafio do ponto de vista de concepção e estruturação funcionando como uma *bridge* entre os clientes da plataforma e o respectivo *middleware* e subsistemas do operador. O seu propósito base enquadra-se com o objetivo da *pattern Bridge*:

> *"Decouple an abstraction from its implementation so that the two can vary independently."*[1]

Este nó apresenta assim um papel fundamental para o sucesso da plataforma tendo sido trabalhado progressivamente ao longo dos últimos quatro anos dotando-o da robustez necessária para atingir o seu propósito.

[1] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

Existem ainda um conjunto de nós de suporte à plataforma como é o caso do *Database Server* e do *TV Cache Server*. O nó *Applications & External Services* reúne toda a lógica de acesso a dados externos à plataforma, por exemplo o acesso às redes sociais que fornecem informação ao motor de recomendações desenvolvido pela empresa.

4.1.3.1. Integração com o Mediaroom

A integração da plataforma com o *middleware* Mediaroom é feita com base num conjunto de web services SOAP disponibilizados para o efeito, Business Support Systems e Operations Support Systems.

Na Tabela 3 e Tabela 4 apresentam-se os web services usados no processo de integração assim como uma breve descrição do seu propósito.

BSS Web Services	Descrição
Principal Management Web Service	Web Service responsável pela provisionamento e gestão de contas, dispositivos, *subscribers* e *subscribers group*
Billing Record Management Web Service	Web Service que permite realizar a gestão de registos de faturação possibilitando obter um histórico de compras relativo às várias contas
Offer Management Web Service	Web Service responsável pela gestão das várias *offers* do Mediaroom incluindo preço, moeda e data de expiração da oferta

Tabela 3 - Web Services BSS utilizados

OSS Web Services	Descrição
VOD Branch Management Web Service	Web Service responsável gestão do conteúdo *Video on demand* (VOD) provisionado no Mediaroom
EPG Web Service	Módulo responsável pela consulta e gestão da informação relativa aos programas pertencentes ao *Electronic Program Guide* (EPG)
Channel Management Web Service	Web Service responsável pela gestão dos canais da plataforma e respetivo mapeamento pelos *subscribers groups* assim como um gestão de um conjunto de definições relativas aos *subscribers groups*.
Remote Recording Web Service	Web Service responsável por realizar os agendamentos de programas assim como consultar e gerir as diferentes gravações relativas às STB.

Tabela 4 - Web Services OSS utilizados

Na Figura 8 encontra-se ilustrada o fluxo de comunicação entre os clientes e as plataformas sendo evidenciado o protocolo usado assim como o formato dos dados transferidos.

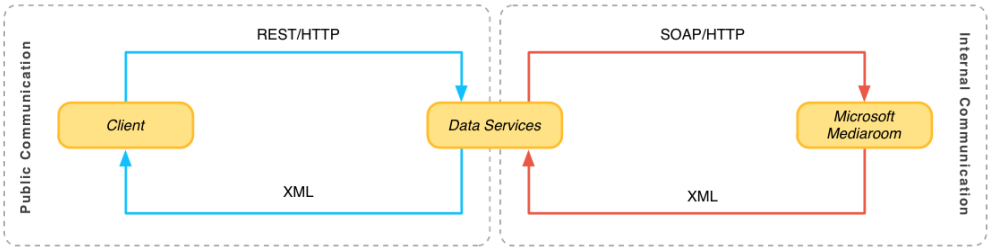

Figura 8 – Fluxo de comunicação envolvendo o *Microsoft Mediaroom*

4.1.3.2. Media Proxy

O Media Proxy existente na plataforma representa um proxy para conteúdo multimédia, acesso a imagens externas e internas à plataforma. Foi desenvolvido com o intuito de solucionar a limitação de formatos e tamanhos suportados pelas STB tendo sido progressivamente evoluído com novas funcionalidades. Atualmente possuí mecanismo de *image transcoding*, operações de *re-scale* de imagens com opções como manter o *aspect ratio* e realizar um auto-zoom, assim como contém uma componente de *caching* para promover a reutilização de processamento. É de particular relevância o uso deste nó por permitir estabelecer um controlo sobre o conteúdo transferido por parte dos clientes, possibilitando uma redução da largura de banda consumida assim como reduzir as operações sobre as imagens realizadas pelos clientes, permitindo assim libertar algum poder de computação.

4.1.3.3. Integração dos clientes

A integração dos clientes com a plataforma é feita recorrendo à API REST dos *data services* sendo o acesso a este nó realizado de forma direta por parte dos clientes. Esta foi uma das alterações de requisitos definidos para a plataforma tendo estado inicialmente previsto o desenvolvimento de uma *gateway* que visava ser o ponto de acesso dos clientes e lidar com mecanismos da plataforma como a gestão de sessões, *caching*, entre outras caraterísticas.

A plataforma foi sujeita a um novo conjunto de requisitos o que alterou substancialmente o processo de integração por parte dos clientes, sendo agora necessário cumprir um conjunto de condições para proceder à integração. Estas condições são enumeradas de seguida:

i. Qualquer dispositivo que comunique com a plataforma deve proceder ao seu registo previamente, sendo-lhe atribuído um *Device Unique Identifier* (DUID) que deve ser guardado de forma permanente pelo dispositivo;

ii. Qualquer acesso à plataforma requer o estabelecimento prévio de uma sessão assim como a sua renovação quando esta expirar;

iii. Todos os acessos à plataforma deverão ser autenticados recorrendo ao algoritmo de autenticação especificado e que faz recurso ao DUID atribuído;

iv. A obtenção de dados internacionalizáveis deve ser especificados no respetivo pedido à plataforma.

Os clientes de modo a procederem à devida integração com a plataforma devem garantir o cumprimento destes condições na sua totalidade.

A plataforma usa o formato XML para transmissão dos dados sendo igualmente necessário o suporte a este formato de dados por parte dos clientes de modo a garantir uma correta integração. As respostas da plataforma poderão igualmente especificar na resposta a respetiva validade associada aos dados recebidos de modo a dar suporte a um mecanismo de *caching* por parte dos clientes.

4.2. Requisitos

O processo de especificação de requisitos apresenta-se como uma das etapas cruciais no desenvolvimento de um projeto de software, permitindo identificar quais os objectivos pretendidos e a forma como devem ser atingidos, possibilitando não só definir uma linha orientadora para os desenvolvimentos assim como uma forma de avaliar objectivamente o sucesso e concretização do projeto.

Nesta secção são apresentados os resultados do processo de especificação de requisitos realizado pelo estagiário em colaboração com a empresa. Os resultados encontram-se divididos por módulos de modo a representar as diferentes componentes do projeto podendo os detalhes ser consultados no Anexo B.

4.2.1. Requisitos gerais

Multi-dispositivo	A aplicação deverá suportar os dispositivos iPhone assim como os dispositivos iPad fixando a orientação do ecrã verticalmente (*portrait*) no iPhone e horizontalmente (*landscape*) na versão iPad
Registo do dispositivo	A aplicação deve proceder ao registo do dispositivo na plataforma de modo a poder ser possível identificar os dispositivos univocamente e limitar o nº de dispositivos associados a uma conta
Armazenamento seguro do identificador único do dispositivo (DUID[2])	A plataforma requer que cada dispositivo possa ser identificado univocamente, gerando um identificador que deverá ser guardado no dispositivo e deverá ser mantido entre instalações
Obtenção e gestão da sessão do cliente	A utilização da plataforma pressupõe a existência de uma sessão associada, independentemente do tipo de utilizador, sendo necessário proceder à sua criação e renovação quando esta ficar inválida
Autenticação dos pedidos à plataforma	Todos os pedidos realizados à plataforma necessitam ser assinados com um *token* de autenticação gerado através do DUID

[2] DUID – Device Unique Identifier

Suporte a ligações seguras	A comunicação com a plataforma deverá ser estabelecida de forma segura recorrendo ao protocolo HTTPS de modo a garantir a segurança dos dados a circular na rede
Suporte a respostas comprimidas	A aplicação deverá suportar respostas comprimidas (GZIP) de modo a reduzir o tempo de transferência dos dados e consequentemente aumentado a responsividade da aplicação
Suporte a cache	De modo a reduzir os acessos à rede e atingir uma redução da largura de banda consumida deverá existir suporte a uma cache aplicacional promovendo a reutilização de dados já transferidos resultando num aumento da responsividade da aplicação e consequente melhoria na experiência de utilização
Suporte a alterações de configurações	A aplicação deverá suportar a obtenção remota de um conjunto de configurações de modo a conduzir a um ajustamento do seu funcionamento interno sem requerer a geração de uma nova versão
Verificar se o dispositivo foi comprometido	A aplicação deverá verificar se o dispositivo foi comprometido através da técnica de *jailbreak*, não devendo nesse caso arrancar a mesma por esta técnica poder comprometer a segurança da aplicação
Suporte à navegação de utilizadores não-autenticados	A aplicação deverá permitir a sua utilização por parte de utilizadores não autenticados de modo a permitir o acesso ao conteúdo TV
Aplicação de políticas de restrição ao conteúdo	O operador deverá poder controlar quais as componentes da aplicação ativas através de um conjunto de regras definidas com base no tipo de utilizador, o perfil de rede e a localização atual devendo a aplicação proceder à validação das regras de modo a determinar se as operações podem ou não ser executadas
Suportar internacionalização	A aplicação deverá suportar internacionalização no conteúdo textual pertencente à própria aplicação assim como dos dados fornecidos pela plataforma

Tabela 5 - Requisitos gerais

4.2.2. Video on Demand (VOD)

Nesta secção encontram-se representados os requisitos pretendidos para o módulo VOD visando a disponibilização do acesso a conteúdo de vídeo do operador por parte dos utilizadores.

Homepage	A aplicação cliente deve fornecer um ecrã de entrada onde apresentará conteúdo selecionado pelo operador
Navegação por categorias	Deve existir um mecanismo de navegação por categorias e subcategorias (nível-N)
Navegação, pesquisa e compra de VOD Packages	A aplicação deverá fornecer suporte à consulta, pesquisa e compra de PVOD

Acesso aos VOD Packages adquiridos (PVOD)	A aplicação deverá suportar a consulta de PVODs adquiridos pelos utilizadores
Navegação, pesquisa e compra de Transactional VOD (TVOD)	A aplicação deverá fornecer suporte à consulta, pesquisa e compra de TVOD
Acesso aos Transactional VOD adquiridos (TVOD)	A aplicação deverá suportar a consulta de TVODs adquiridos pelos utilizadores
Navegação, pesquisa e compra de Subscription VOD (SVOD)	A aplicação deverá fornecer suporte à consulta, pesquisa e compra de SVOD
Acesso aos Subscription VOD adquiridos (SVOD)	A aplicação deverá suportar a consulta de SVODs adquiridos pelos utilizadores
Partilha de conteúdo para a rede social Facebook	A aplicação deverá suportar a partilha do conteúdo VOD na rede social

Tabela 6 - Requisitos de Video on Demand

4.2.3. Electronic Program Guide (EPG)

Nesta secção encontram-se representados os requisitos pretendidos para o módulo EPG visando a disponibilização de funcionalidades de consulta da programação do operador.

Programação Diária	Apresentação da programação diária por canal (Exclusivo da versão iPhone)
Programação no formato de grelha de programação	Apresentação de uma grelha de programação que apresenta a programação de forma dinâmica para os vários canais do operador assim como para os vários dias da semana
Sintonização de canais	Realizar a sintonização de canais diretamente a partir da grelha de programação
Pesquisa de canais e programas	Pesquisar canais assim como programas diretamente na plataforma
Canais favoritos	Lista de canais favoritos do utilizador permitindo a sua edição e remoção possibilitando a visualização da grelha de programação apenas com este conjunto de canais
Visualização de detalhes dos programas	Apresentação da informação dos programas
Agendar e cancelar o agendamento de programas	Proceder ao agendamento e cancelamento de programas a partir da grelha de programação
Partilha de conteúdo para a rede social Facebook	Partilha dos programas na rede social permitindo fornecendo uma componente social à grelha de programação

Tabela 7 - Requisitos de Electronic Program Guide

4.2.4. Personal Video Recorder (PVR)

Nesta secção encontram-se representados os requisitos pretendidos para o módulo PVR pretendendo-se fornecer uma consulta e controlo dos programas agendados e gravados pelos utilizadores.

Listagem dos programas gravados e agendados na STB	Acesso e navegação sobre o conteúdo gravado e agendado para gravação pela STB
Pesquisa de programas gravados e agendados	Pesquisa de programas gravados e agendados permitindo assim realizar uma rápida filtragem de conteúdo
Visualização de detalhes de programas singulares e séries	Apresentação da informação dos programas assim como a listagem de outros episódios caso se trate de uma série
Gerir os programadas agendados e gravados	Permitir o cancelamento de programas agendados assim como o apagamento de programas já gravados pela STB
Reprodução dos programas gravados na TV	Iniciar a reprodução do conteúdo gravado pela STB diretamente na TV a partir do dispositivo

Tabela 8 - Requisitos de Personal Video Recorder

4.2.5. Remote Control

Nesta secção apresentam-se os requisitos pretendidos para o desenvolvimento da funcionalidade de controlo remoto pela aplicação.

Informação contextual do canal sintonizado	Apresentação do canal atualmente sintonizado assim como o programa que está no ar e respetivo progresso
Canais favoritos	Deverá ser possível adicionar e remover facilmente um canal da lista de favoritos assim como navegar nesta lista
Sintonizar canal	Sintonizar um canal na STB diretamente a partir da aplicação cliente
Funcionalidade de comando remoto	Controlar o volume da televisão assim como permitir a navegação pelos menus, acesso aos tradicionais botões do controlo remoto

Tabela 9 - Requisitos do Remote Control

4.2.6. Local Play

Esta secção apresenta os requisitos pretendidos para o módulo de partilha de conteúdo para a TV.

Partilha de imagens para a TV	Com recurso à comunicação com a STB, deverá ser possível o envio e visualização de fotografias armazenadas no dispositivo diretamente na TV

Partilha de vídeos para a TV	Com recurso à comunicação com a STB, deverá ser possível o envio e visualização de vídeos armazenados no dispositivo diretamente na TV

Tabela 10 - Requisitos do Local Play

4.2.7. Políticas de restrição

Esta secção apresenta os requisitos pretendidos para a aplicação de políticas de restrição ao conteúdo de modo a fornecer um controlo sobre as funcionalidades disponibilizadas consoante um conjunto de condições.

Aplicação de políticas de restrição a módulos	Os vários módulos da aplicação estão sujeitos a políticas restritivas por parte do operador com base num conjunto de condições: i) tipo de utilizador; ii) perfil de rede; iii) localização atual; ficando bloqueados e alertando o utilizador sempre que determinado componente esteja restringida
Aplicação de políticas de restrição a funcionalidades	As várias funcionalidades da aplicação, e.g. reprodução de conteúdo, estão igualmente susceptíveis a políticas restritivas por parte do operador com base num conjunto de condições: i) tipo de utilizador; ii) perfil de rede; iii) localização atual; ficando bloqueadas e alertando o utilizador sempre que determinada funcionalidade esteja restringida

Tabela 11 - Requisitos para as políticas de restrição

4.3. Desafios e dificuldades

Analisando os requisitos podemos verificar que existe um leque vasto de funcionalidades pretendidas para a aplicação cujos dispositivos alvo apresentam ainda um poder de computação algo limitado, tendo existido a necessidade constante de monitorizar os recursos utilizados de modo a alcançar a performance e fluidez pretendidas para a aplicação. A falta de experiência de desenvolvimento para estes dispositivos pelo estagiário contribuiu igualmente para este desafio, tendo sido minimizado com um forte investimento em investigação e pesquisa da plataforma e da própria linguagem de modo a conseguir retirar o máximo daquilo que disponibiliza.

Outro dos desafios do projeto passou pelo suporte multi-dispositivo, iPhone e iPad, tendo sido necessário realizar um investimento na procura de mecanismos que permitissem diminuir não só o esforço de desenvolvimento para as duas plataformas mas principalmente promover a reutilização da lógica envolvida.

Do ponto de vista motivacional estes desafios apresentaram-se aliciantes e atrativos pela componente de pesquisa que envolveu que permitiu enriquecer significativamente os conhecimentos nesta área assim como conceber uma arquitetura na sua totalidade. Os capítulos seguintes apresentam as soluções encontradas para resolução destes desafios.

4.4. Arquitetura da aplicação iOS

Um dos requisitos da aplicação passa pelo suporte dos dispositivos iPhone e iPad o que requer a definição de uma arquitetura que promova a partilha de código entre ambas as plataformas de modo a diminuir o custo de desenvolvimento e garantir uma manutenção do projeto mais eficiente através da propagação de correções e melhorias por ambos os dispositivos. Com este requisito em mente foi especificada a separação total da camada de apresentação de toda a lógica de negócio de modo a reunir e centralizar toda esta lógica numa biblioteca partilhada pelas aplicações iPhone e iPad, *TVE Client Library*.

A arquitetura foi moldada com base na arquitetura de três camadas com o objetivo de fornecer uma abstração à camada de negócio relativamente ao acesso dos dados, ficando sim responsável por processar os dados recebidos, definir os fluxos de execução assim como coordenar o troca de dados entre as duas camadas, a camada de apresentação e a camada de acesso a dados. Esta abstração fornece flexibilidade à camada de acesso a dados permitindo conduzir alterações sem com isso afetar a restante lógica. Alguns exemplos passam pela alteração do *parser* usado para descodificação dos dados assim como o formato dos dados poder ser alterado sem com isso afetar a camada de negócio ou a camada de apresentação.

Na Figura 9 é possível visualizar a arquitetura de três camadas usada ficando evidenciada a separação entre a aplicação e a biblioteca.

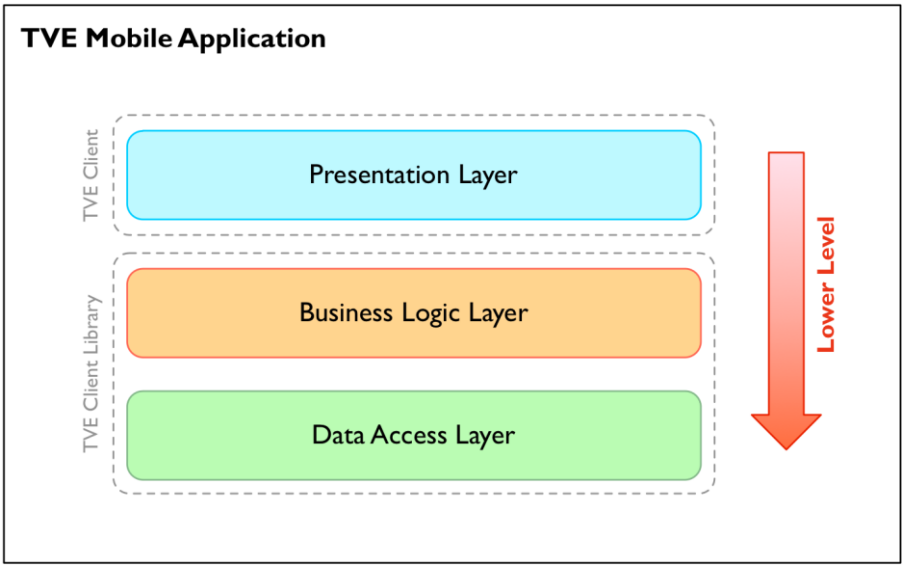

Figura 9 - Arquitetura de 3 camadas da aplicação TVE

Uma consequência positiva desta arquitetura passa pela possibilidade de suportar múltiplas aplicações com camadas de apresentação diferentes, adoptadas aos requisitos do operador, reutilizando toda a lógica existente na biblioteca promovendo a partilha de funcionalidades entre as aplicações o que permite diminuir os custos de desenvolvimento assim como atingir uma maior maturação e consolidação da biblioteca e uma possível extensão com novas funcionalidades.

Um exemplo da sua aplicabilidade poderá ser o desenvolvimento no futuro da aplicação TVE para a plataforma Mac OS X que requer obrigatoriamente uma camada de apresentação específica da plataforma, podendo no entanto ser reutilizada a biblioteca *TVE Client Library* para toda a sua lógica interna derivado da forte partilha de *Frameworks* e APIs entre as duas plataformas, sendo apenas necessário lidar com algumas das especificidades existentes.

4.4.1. Biblioteca cliente

A biblioteca cliente surgiu com o propósito de garantir o isolamento das camadas de negócio e acesso a dados da camada de apresentação de modo a assegurar um elevado nível de desacoplamento possibilitando alterações nas respetivas camadas sem com isso requerer alterações do lado das aplicações, especializadas na apresentação da aplicação.

A biblioteca consiste numa biblioteca estática escrita em Objetive-C que é incluída no processo de linkagem da respectiva aplicação, sendo em tudo semelhante às dependências de projetos C/C++ que fazem recurso a bibliotecas externas, sendo neste caso estáticas e não dinâmicas visto o iOS não suportar a linkagem dinâmica de bibliotecas que não estejam presentes nativamente na plataforma por políticas de segurança.

A integração da aplicação com a biblioteca é realizada através da exposição de uma API pública, implementada de uma forma abstracta de modo a não expor nem requerer lógica interna da biblioteca. Esta API permite fornecer um ponto central aos dispositivos iPhone e iPad atingindo assim o objetivo de evitar a duplicação de código assim como promover a sua reutilização.

Tendo em consideração as convenções[23] da Apple todas as classes pertencentes à biblioteca serão precedidas do prefixo WTV[3].

Com a adopção do modelo de três camadas a biblioteca ficou diretamente responsável pela implementação da camada de negócio e da camada de acesso a dados, sendo que este desacoplamento tem como objetivo abstrair o cliente dos vários detalhes técnicos fornecendo assim uma API simples e abstracta que disponibiliza todo o suporte necessário pelas aplicações. Esta abstração fornece uma flexibilidade sem paralelo por permitir alterações na implementação tanto da aplicação como da biblioteca sem que com isso se afetem mutuamente, implementando conceptualmente a *pattern Bridge*.

Tendo isto em conta a biblioteca é constituída por três camadas ao invés de duas de modo a disponibilizar a API pública usada pelas aplicações.

[3] WIT-TV

Na Figura 10 ilustra a arquitetura da biblioteca assim como as várias componentes que a constituem.

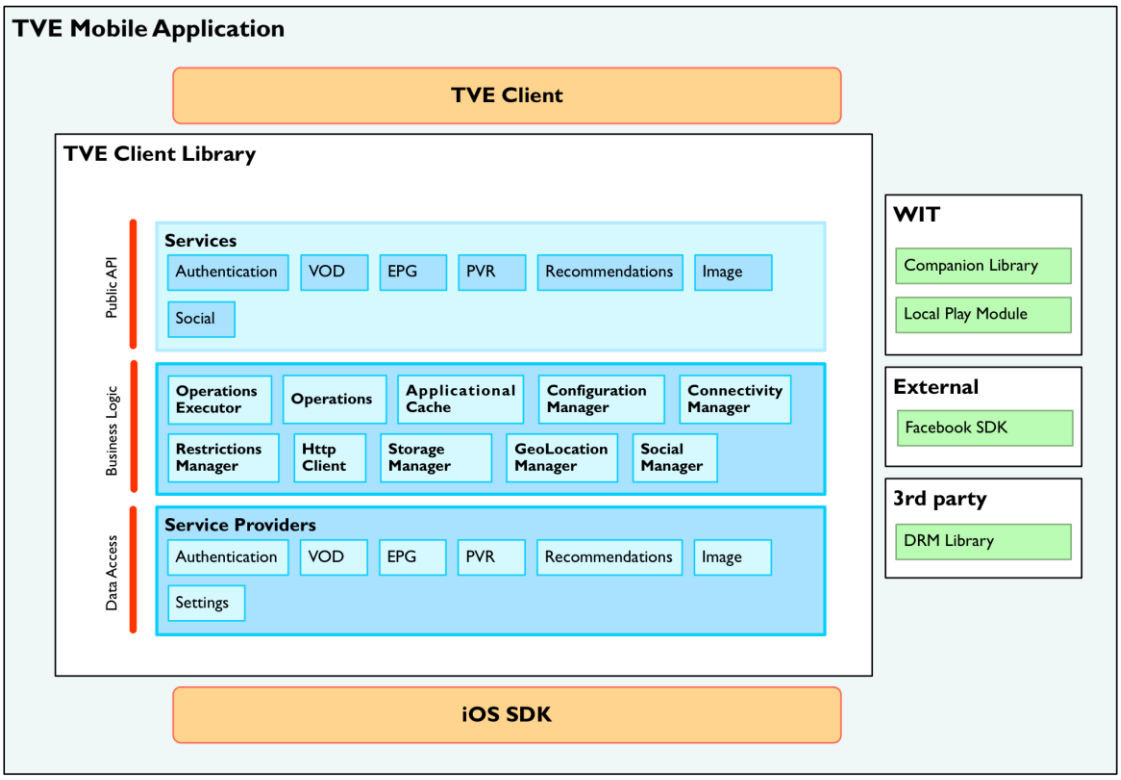

Figura 10 - Arquitetura da biblioteca cliente

De seguida serão introduzidas as várias camadas constituintes da biblioteca cliente.

4.4.1.1. Camada de acesso a dados

A camada de acesso a dados é responsável por realizar toda a comunicação com a plataforma através dos *data services*, realiza toda a obtenção de dados assim como a realização de operações através dos serviços. Esta é a camada de mais baixo nível da biblioteca estando dividida por módulos denominados *ServiceProviders*.

Um *ServiceProvider* procede à integração com a plataforma, através dos vários *web services* disponíveis, disponibilizando o acesso às várias funcionalidades disponíveis e constituindo assim a API interna que fornece o suporte para as camadas superiores. É inteiramente responsável pela componente de comunicação assim como da respetiva conversão dos dados derivados dessa comunicação para o modelo de dados da biblioteca.

Na Figura 11 é possível visualizar os vários *ServiceProviders* que constituem a camada de acesso a dados.

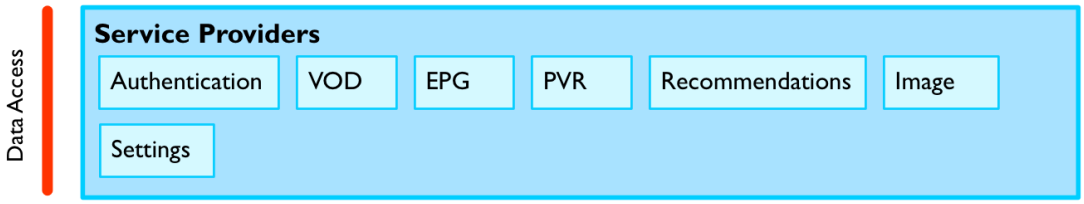

Figura 11 - Componentes da camada de acesso a dados

A Tabela 12 apresenta as várias componentes constituintes da camada de acesso a dados da biblioteca.

Authentication	Fornece interfaces internas para proceder à autenticação do utilizador assim como realizar o registo do dispositivo e obtenção de uma sessão
VOD	Fornece interfaces internas para acesso às várias componentes do módulo de VOD desde a obtenção das várias listas de conteúdo, pesquisa, compra
EPG	Fornece interfaces internas para acesso às várias componentes do módulo de EPG desde a obtenção da lista de canais, obtenção da programação dos canais, gestão dos canais favoritos, pesquisa
PVR	Fornece interfaces internas para acesso às várias componentes do módulo de PVR desde a obtenção dos vários programas gravados e agendados, agendamento de programas, cancelamento de programas, pesquisa
Recommendations	Fornece interfaces internas para acesso às recomendações existentes na plataforma
Image	Fornece interfaces internas dedicadas ao carregamento de imagens da plataforma
Settings	Fornece interfaces internas para o download das regras que alimentam o gestor de restrições para aplicação de restrições ao conteúdo

Tabela 12 - Componentes constituintes da camada de acesso a dados

4.4.1.2. Camada de negócio

A camada de negócio apresenta um papel central na arquitetura da biblioteca ao reunir o conjunto de entidades que procedem à invocação dos *ServiceProviders* e que contêm o completo fluxo de execução de uma determinada operação. Estas entidades, que têm um papel de orquestradores, são denominadas *Operations*. É nestas unidades que se encontra todo o negócio da biblioteca e consequentemente da aplicação, visto todos os

acessos a dados serem realizados aqui assim como o respetivo, *parsing*, *caching* de dados, validações de erros.

Para além das *Operations*, é nesta camada que encontramos toda a lógica de suporte como a cache aplicacional, o gestor de conectividade, o gestor de restrições entre outros, sendo depois integrados no fluxo de execução das *operations*. Exemplo disso é a cache aplicacional que é usada nas várias *operations* de modo a tirar partido de mecanismos de *caching*.

Na Figura 12 podemos visualizar as várias componentes constituintes da camada de negócio da biblioteca.

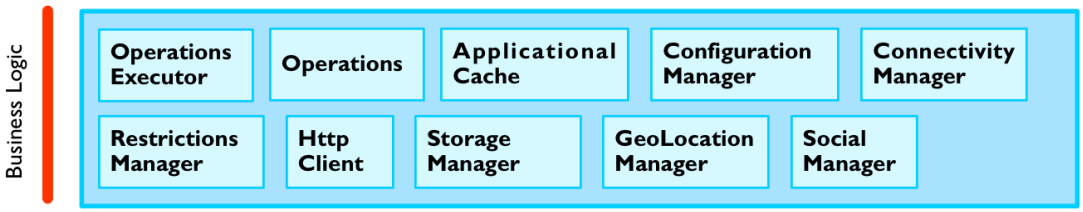

Figura 12 – Camada de negócio da biblioteca

A Tabela 13 apresenta as várias componentes constituintes da camada de negócio da biblioteca.

Operations	Entidades responsáveis por definir o fluxo de execução das várias operações suportadas pela biblioteca, funcionando como orquestradores
Operations Executor	Entidade responsável pelo agendamento e execução das várias *operations* da biblioteca sendo esta execução feita em *threads* secundárias e não na *main-thread*
Applicational Cache	Cache aplicacional da biblioteca que permite o armazenamento de objetos já no modelo de dados da aplicação evitando assim o processo de *parsing* de dados, suportando uma validade associada; possui igualmente suporte para especificar o espaço máximo a ser usado pela cache
Configuration Manager	Entidade responsável por toda a gestão das configurações da aplicação, desde os *endpoints* dos serviços da plataforma, o tamanho máximo da cache aplicacional, entre outras configurações; estas configurações atualmente são embutidas na aplicação mas no futuro serão fornecidas pelo *back-end*
Connectivity Manager	Entidade responsável por determinar a rede atual à qual o dispositivo se encontra ligado assim como observar mudanças e detetar a perda de conectividade internet
Restrictions Manager	Entidade responsável por validar o conjunto de regras de restrição ao conteúdo definidas pelo operador tendo em conta a localização atual do dispositivo e o tipo de utilizador autenticado na aplicação definindo assim quais as funcionalidades e módulos da aplicação ativos

HTTP Client	Entidade responsável por realizar todos os pedidos HTTP da biblioteca
Storage Manager	Entidade responsável pelo armazenamento de dados no dispositivo, *filesystem*, assim como armazenamento de credenciais de forma segura através do *keychain* da plataforma
GeoLocation Manager	Entidade responsável por detetar as coordenadas atuais do dispositivo e mapear essas coordenadas para um país de modo a fornecer suporte ao *Restrictions Manager* Nota: devido a alterações de requisitos a plataforma passou a fornecer a localização atual do dispositivo
Social Manager	Entidade responsável por gerir as contas sociais associadas na aplicação e fornecer uma API para a partilha de conteúdo de uma forma mais simplificada; atualmente suporta a rede social Facebook

Tabela 13 - Componentes constituintes da camada de negócio

4.4.1.3. API pública

A API pública representa o ponto de integração entre a aplicação e a biblioteca disponibilizando o conjunto de funcionalidades disponíveis para as aplicações através dos *Services*, abstraindo-as de todos os detalhes da implementação fornecendo apenas o resultado da execução da operação. Do ponto de vista técnico esta camada apresenta uma lógica muito reduzida contendo apenas a inicialização da respectiva *operation* responsável por toda a execução da operação, sendo posteriormente submetida para execução no *Operations Executor*.

Na Figura 13 podemos observar os vários *Services* disponibilizados por esta camada.

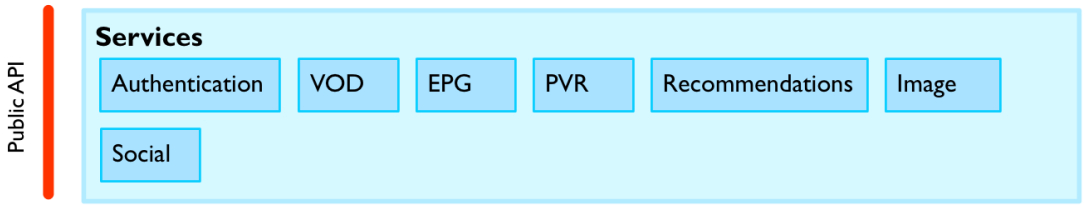

Figura 13 – API pública da biblioteca

A Tabela 14 apresenta as várias componentes constituintes da API pública da biblioteca.

Authentication	Fornece interfaces públicas para proceder à autenticação do utilizador assim como realizar o registo do dispositivo e obtenção de uma sessão
VOD	Fornece interfaces públicas para acesso às várias componentes do módulo de VOD desde a obtenção das várias listas de conteúdo, pesquisa, compra

EPG	Fornece interfaces públicas para acesso às várias componentes do módulo de EPG desde a obtenção da lista de canais, obtenção da programação dos canais, gestão dos canais favoritos, pesquisa
PVR	Fornece interfaces públicas para acesso às várias componentes do módulo de PVR desde a obtenção dos vários programas gravados e agendados, agendamento de programas, cancelamento de programas, pesquisa
Recommendations	Fornece interfaces públicas para acesso às recomendações existentes na plataforma
Image	Fornece interfaces públicas dedicadas ao carregamento de imagens da plataforma
Social	Fornece interfaces para partilha de conteúdo nas redes sociais de uma forma abstrata; atualmente usa o SDK do Facebook para fazer a partilha de conteúdo nesta rede social

Tabela 14 - Componentes constituintes da API pública

Na Figura 14 podemos observar o fluxo e as várias entidades envolvidas numa chamada à API passando desde a *Operation* respetiva, ao *Operations Executor* até as entidades que a *Operation* utiliza de modo a executar a respetiva operação.

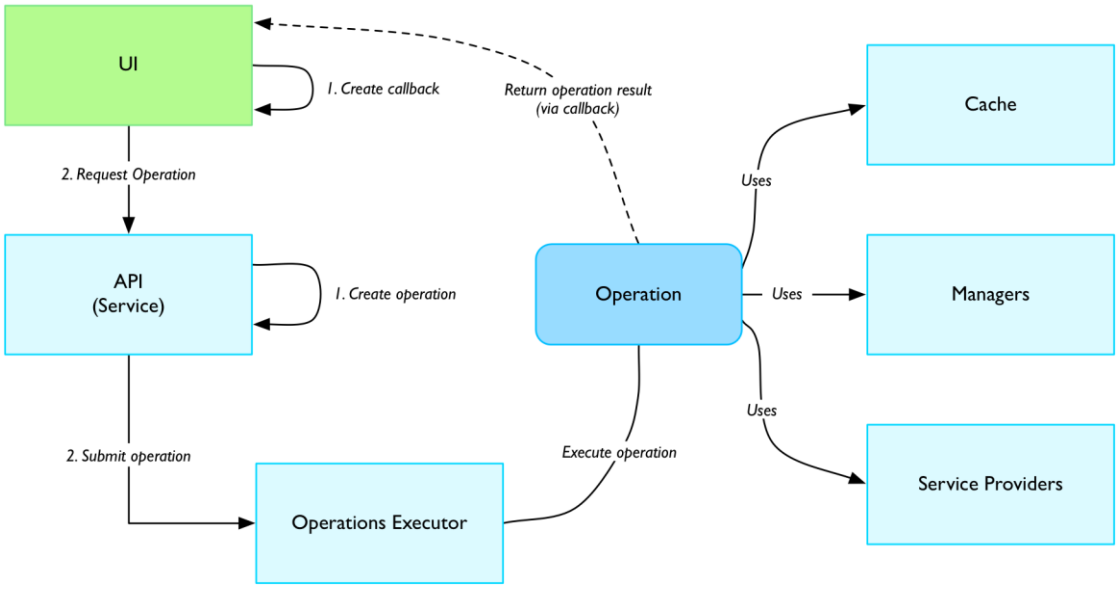

Figura 14 - Entidades envolvidas numa chamada à API

Na Figura 15 é possível visualizar a sequência de execução base resultante de uma chamada à API, sendo possível visualizar os vários intervenientes que contribuem para a execução da operação.

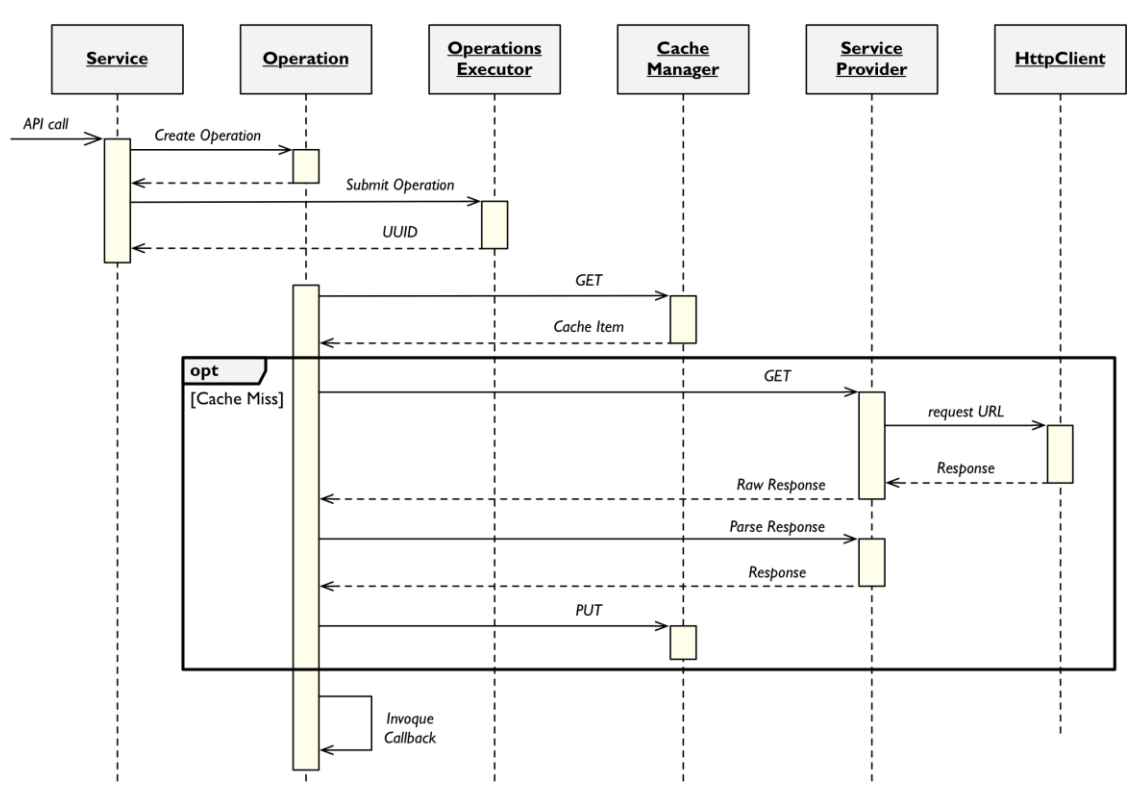

Figura 15 - Sequência de execução numa chamada à API

4.4.1.4. Módulos externos

A biblioteca faz ainda recurso a um conjunto de módulos externos de modo a auxiliar e completar o conjunto de funcionalidades que disponibiliza. Na Tabela 15 são apresentados os vários módulos externos constituintes da biblioteca.

Companion Library	Este módulo é responsável pelo emparelhamento do dispositivo com a STB lidando com toda a lógica de comunicação, envio de comandos que permitem realizar remotamente operações como a sintonização de um canal, simular o clique num botão do telecomando, entre outras funcionalidades. Este componentes encontra-se isolado tendo em conta que suporte atualmente apenas a plataforma Mediaroom podendo ser ou não incluído na aplicação dependendo das pretensões dos operadores
Local Play Module	Este módulo reúne toda a componente de partilha de conteúdo do dispositivo para a TV, imagem e vídeo, e está dependente do módulo Companion Library por requerer um emparelhamento com a STB, tendo sido seguida a mesma abordagem de isolamento para inclusão na aplicação consoante as pretensões dos operadores

Facebook SDK	Este SDK dá suporte à partilha de informação na rede social Facebook tendo sido integrado na biblioteca a fim de fornecer uma interface abstrata para os clientes usarem na partilha de conteúdo para a rede social
DRM Library	Este módulo atualmente não está integrado na biblioteca tendo em conta que existem várias soluções no mercado e não é possível suportar todas, assim este módulo será devidamente integrado consoante a solução usada ou escolhida pelo operador para reprodução de conteúdo protegido

Tabela 15 - Módulos externos usados pela biblioteca

4.4.2. Aplicação TVE

A aplicação TVE representa uma aplicação nativa iOS escrita em Objetive-C, sendo inteiramente responsável pela camada de apresentação e delegando toda a lógica de execução de operações como de acesso a dados para a biblioteca cliente. Este isolamento possibilita o suporte de múltiplas versões do cliente com diferentes interfaces gráficas ou mesmo clientes entre diferentes plataformas, iOS e Mac OS X.

A estruturação do cliente tem como base a *pattern* Model-View-Controller (MVC). Esta *pattern* possui um papel fundamental em vários componentes e mecanismos das frameworks Cocoa e Cocoa Touch que constituem a respetiva base das aplicações para Mac OS X e iOS. De salientar no entanto que os modelos, uma das componentes da *pattern*, não estarão localizados maioritariamente na aplicação visto ser responsabilidade da biblioteca definir os modelos de dados usados.

A Apple usa uma versão modificada da *pattern* para as suas frameworks que se encontra representada na Figura 16.

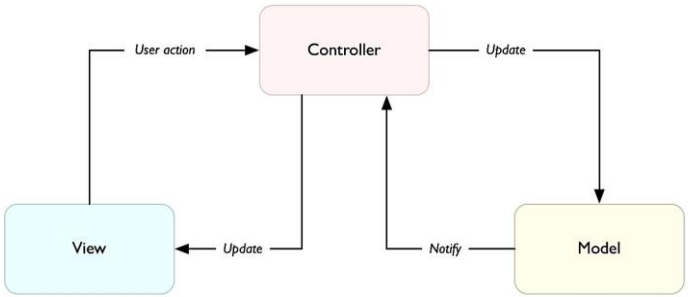

Figura 16 - Cocoa MVC

Tendo em conta o suporte de dois dispositivos, iPhone e iPad, torna-se necessário definir estratégias que possibilitem e promovam a reutilização de código assim como de componentes entre ambos os dispositivos. O isolamento e centralização da camada de negócio e acesso a dados na biblioteca já garante uma total reutilização destas camadas. Para promover a reutilização de componentes entre dispositivos criou-se uma *Universal App* em detrimento de aplicações dedicadas para cada dispositivo, assim existe um só projeto para ambos os dispositivos promovendo a partilha de componentes comuns entre ambos.

Outro aspeto a ter em atenção passa pelo conjunto de interfaces da aplicação que em alguns módulos são muito semelhantes entre dispositivos excetuando as respetivas dimensões assim como comportamentos específicos de cada dispositivo. Assim de modo a promover a reutilização de componentes comuns entre dispositivos é feito exaustivamente recurso a mecanismos de herança para reutilizar lógica entre *controllers* e uso da *creational pattern Abstract Factory* de modo a abstrair a inicialização dos componentes das duas plataformas. Com o uso desta *pattern* o respetivo *controller* que deve ser inicializado é determinado em *runtime* evitando assim que cada *controller* tenha conhecimento de qual o componente que deve inicializar na respetiva versão iPhone e iPad.

Tendo em consideração as convenções[23] da Apple todas as classes pertencentes ao cliente serão precedidas do prefixo TVE.

Versão do sistema operativo

Um dos aspetos relevantes no desenvolvimento do cliente passa pela versão mínima do sistema operativo a suportar que têm impacto nas APIs disponíveis assim como nos componentes disponibilizados pelo SDK.

O ecossistema iOS é caracterizado pela forte adopção da última versão do SO por parte dos seus utilizadores registando-se uma fragmentação muito reduzida, o que proporciona aos programadores a possibilidade de suportar apenas as versões mais recentes do sistema operativo.

Atualmente o sistema operativo iOS está na versão 6 tendo sido lançado em Setembro de 2012, sendo que passado apenas 1 mês após o seu lançamento cerca de 60% dos utilizadores já tinham feito a atualização da versão 5 para a versão 6 e apenas cerca de 10% dos seus utilizadores ainda usavam uma versão inferir à 5. Tendo em conta este comportamento e estes números a aplicação suportará a versão mínima 5 do sistema operativo.

Recentemente na WWDC'13 foi anunciado que de 93% dos utilizadores usam a versão 6 do iOS e apenas 6% ainda se encontram a usar a versão 5[4]. Tendo em conta igualmente o lançamento do iOS 7 no Outono será natural no futuro aumentar a versão mínima suportada para a versão 6. Na Figura 17 é possível visualizar os números anunciados e disponíveis no site de programadores da plataforma[24].

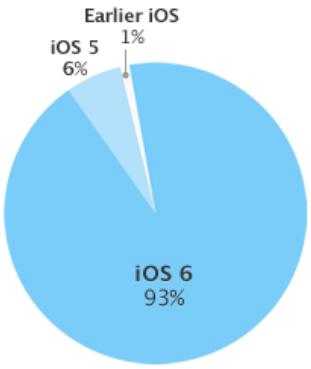

Figura 17 - Percentagem de utilização das diferentes versões do iOS

[4] Estes números foram calculados com base no tráfego da App Store num período de 14 dias terminando a 3 de Junho de 2013.

Capítulo 5
Implementação

Neste capítulo são apresentados os detalhes de implementação e decisões tomadas no desenvolvimento da aplicação iOS de modo a cumprir os objetivos traçados para o projeto.

Tendo em conta o desenvolvimento ter sido conduzido em equipa são apresentadas de seguida as componentes da biblioteca desenvolvidas pelo estagiário.

- *Services*
- *Operations*
- *Operations Executor*
- *Service Providers*
- *Http Client*
- *Connectivity Manager*
- *Storage Manager*
- *GeoLocation Manager*
- *LocalPlay* – módulo de vídeo

Na aplicação existiu igualmente uma divisão sendo de seguida apresentadas as componentes desenvolvidas pelo estagiário.

VOD

- Ecrã de *homepage* – desenvolvida a versão iPhone e iPad
- Ecrã de pesquisa – desenvolvida a versão iPhone e iPad
- Ecrã de detalhes – desenvolvida a versão iPad
- Ecrã de categorias – desenvolvida a versão iPhone e iPad

EPG

- Grelha de programação – desenvolvida a versão iPad
- Ecrã de detalhes – desenvolvida a versão iPad
- Pesquisa – desenvolvida a versão iPhone e iPad

PVR

- Listagem de programas gravados e agendados – desenvolvida a versão iPad
- Ecrã de detalhes – desenvolvida a versão iPad
- Pesquisa – desenvolvida a versão iPhone e iPad

Autenticação

- Ecrã de login – desenvolvida a versão iPhone e iPad

Menu

- Menu lateral com suporte ao utilizador autenticado – desenvolvida a versão iPhone e iPad

5.1. Biblioteca cliente

O desenho e concepção da biblioteca cliente visou um objectivo específico e concreto, desacoplar toda a lógica de negócio da camada de apresentação de modo a minimizar o impacto de modificações e desenvolvimentos específicos para diferentes clientes. Assim, existiu sempre a preocupação de isolar toda a lógica interna na biblioteca delegando apenas a responsabilidade da apresentação às aplicações. De seguida são apresentados alguns aspectos técnicos com a apresentação das opções tomadas no desenvolvimento da biblioteca.

5.1.1. Integração com a plataforma

Na arquitetura da biblioteca, os *ServiceProviders* representam a instância que realiza a devida integração com a plataforma, necessitando assim lidar com os vários requisitos e pormenores associados a este processo, nomeadamente a gestão de sessão assim como a autenticação de todos os pedidos realizados aos web services. Tornou-se assim necessário um *ServiceProvider* ter um conjunto de dados para lhe permitissem realizar e manter o correto estabelecimento da comunicação, um contexto, tendo assim sido criado o objecto *ContextInfo*.

O *ContextInfo* é responsável por reunir toda a informação necessária para permitir a comunicação com a plataforma, contendo a sessão atual que é requerida pela plataforma, o respetivo DUID atribuído ao dispositivo, a PSK[5] necessária no processo de encriptação e autenticação do pedido e finalmente a informação relativa à assinatura da aplicação. Este contexto é uma dependência do *ServiceProvider* sendo injetado a quando da sua invocação.

Recorrendo ao *ContextInfo*, o *ServiceProvider* lida com o conjunto de requisitos que a comunicação com a plataforma pressupõe, como a geração e assinatura de todos os pedidos com o *token* de autenticação, a especificação em cada pedido da língua atual do dispositivo de modo a suportar internacionalização por parte dos dados recebidos, entre outros requisitos.

O *ServiceProvider* é ainda responsável por realizar um primeiro processamento dos dados recebidos nomeadamente uma atualização da localização atual do dispositivo, que é igualmente guardada no *ContextInfo*, assim como verificar a existência de uma validade associada aos dados recebidos de modo a fornecer informação para o mecanismo de *caching* da biblioteca. Perante o retorno da informação recebida pela plataforma assim como a respetiva validade ou um erro no caso de falha, foi criado igualmente um objecto *DataResponse*.

Este objeto é responsável por conter o erro ocorrido caso tenha existido uma falha ou, em caso de sucesso, os dados recebidos no seu formato *raw* assim como a respetiva validade, caso esteja presente através do *header* de cache HTTP. O formato *raw* dos dados visa o desacoplamento da comunicação e transferência dos dados do processo de *parsing*, de modo a fornecer uma API de mais baixo nível e que permite conferir um maior nível de controlo às camadas superiores.

[5] Pre-shared key

Este processo de comunicação é comum a todos os *ServiceProviders* tendo-se realizado a centralização de toda esta lógica num *ServiceProvider* base ficando depois toda a lógica disponível nas respetivas especializações para os vários módulos através do mecanismo de herança.

5.1.2. Comunicação

A comunicação HTTP é feita recorrendo ao *HttpClient*, responsável por realizar todos os pedidos feitos ao nível da biblioteca. O *HttpClient* representa um simples executador de pedidos HTTP, recebendo um *request* composto por um URL e um conjunto de *headers* realiza a sua execução e retorna os dados recebidos em caso de sucesso ou o erro ocorrido em caso de falha. A comunicação é realizada usando um componente nativo do SDK, o NSURLConnection[25]. A utilização deste componente prende-se por oferecer os requisitos necessários para lidar com o mecanismo de comunicação e vir diretamente integrado na plataforma com uma API de fácil integração, evitando a adição de uma dependência externa. A sua utilização estar isolada no *HttpClient* visa igualmente garantir um determinado nível de isolamento que permita uma futura alteração do cliente HTTP responsável por executar os pedidos sem com isso afetar todos os *ServiceProviders*.

A criação do *HttpClient* visou igualmente cumprir um requisito das *guidelines*[26] relativas à interface para aplicações iOS, onde é especificado um componente denominado *Network Activity Indicator*, situado na barra de estado do dispositivo que é responsável por demostrar quando está a ocorrer comunicação pela rede, devendo as aplicações ativar este componente sempre que executem um pedido à rede e desativar quando este termina. Com esta centralização é possível lidar com este mecanismo internamente de uma forma transparente para os *ServiceProviders* que realizam assim simples pedidos HTTP.

Tendo em conta o *Network Activity Indicator* ser uma componente exclusiva de UI do iOS, foi definido um *callback* que deverá ser fornecido pelo cliente de modo a conceber a ativação e desativação deste mesmo componente tendo em conta que este é um requisito da aplicação e não da biblioteca, exemplo disso é a sua inexistência na plataforma Mac OS X não sendo necessário implementar este *callback* nesta plataforma.

5.1.3. Assincronismo

Tendo em conta o modelo por camadas da biblioteca podemos constatar que a camada de negócio apresenta um papel fundamental ao fornecer a implementação para a API pública recorrendo à camada de acesso a dados, quando necessário, representando assim um elo de ligação entre as duas camadas.

A implementação da API pública é feita sob a forma de macro-tarefas que procedem à invocação de toda a lógica necessária à execução daquela operação. Tendo em conta que a grande maioria das operações envolvem acessos à rede e tratamento de dados, torna-se fundamental garantir que estas operações são executadas em *threads* secundárias de modo a não sobrecarregar a *main-thread*, responsável por gerir toda a UI da aplicação. Desta forma, a API pública torna-se de igual forma assíncrona e elimina a preocupação dos clientes em proceder à sua invocação de forma assíncrona sendo este mecanismo gerido pela própria biblioteca.

Este assincronismo foi alcançado recorrendo às *Operations*.

5.1.4. Operations

Uma *operation* do ponto de vista conceptual representa a execução de uma macro-tarefa específica. Do ponto de vista técnico uma *operation* representa um objeto que contém um encapsulamento de código e de dados relativos à execução de uma única tarefa. Este objeto é o chamado *single-shot object*, visto executar a tarefa uma única vez não podendo ser usado para executá-la novamente. A execução destes objetos é realizada numa *queue* de *operations*, responsável por receber *operations* e proceder à sua execução num das *threads* secundárias pertencentes à *thread-pool* interna da queue.

Uma *operation* é criada e submetida para uma *queue* que realiza posteriormente a sua execução, não existindo no entanto a garantia que esta se inicie de imediato podendo ficar em espera até o sistema conseguir executá-la. Uma das caraterísticas deste componente reside na possibilidade de efetuar o seu cancelamento, independentemente de estar em espera ou execução. É, no entanto, responsabilidade da implementação da *operation* verificar periodicamente se esta foi cancelada e nesse caso terminar a sua execução. Tendo em conta estas caraterísticas está inerente a existência de um estado interno associado podendo ser visualizado na Figura 18 os diferentes estados possíveis.

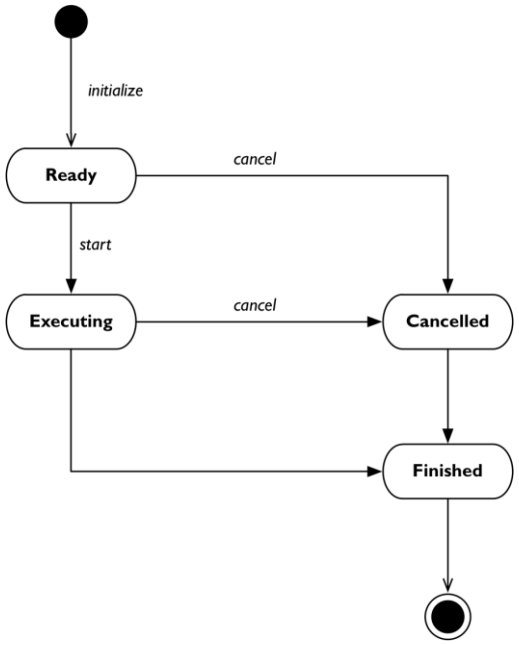

Figura 18 - Diagrama de estado de uma *operation*

A adopção das *operations* deveu-se a um conjunto de fatores:

- a sua execução poder ser realizada em *threads* secundárias o que permite diminuir de imediato a carga sobre a *main-thread*;
- fornece uma abstração relativamente à sua forma de execução, focalizando-se apenas no encapsulamento do código e dos dados para executar a sua tarefa;
- o controlo que fornece ao possibilitar o seu cancelamento independentemente de estar em espera ou já em execução;

O controlo que fornece é o principal fator que pesa na opção de utilização das *operations* por permitir economizar ciclos de processamento quando este já não é efetivamente necessário. Tendo em consideração que estamos a lidar com dispositivos móveis este fator ganha particular relevo pelo limitado poder de computação assim como pelo fator energia pela utilização de baterias. Torna-se crucial possuir este controlo de modo a efetivamente melhorar não só a performance da aplicação assim como diminuir o seu consumo de energia.

Outro dos fatores preponderantes na sua adopção passa pela sua execução em *threads* secundárias possibilitando assim um mecanismo assíncrono, tendo sendo definidos dois *callbacks* de resposta, um de sucesso e um de falha, que serão invocados no final da *operation*. Esta especificação visa proceder à notificação do resultado resultante da sua execução, fornecendo os dados computados através do *callback* de sucesso ou um *NSError*[6] através do *callback* de falha no caso da execução ter falhado, permitindo assim disponibilizar informação sobre o tipo de erro ocorrido.

As *operations*, no contexto da biblioteca, são assim responsáveis por definir o fluxo de execução das várias funcionalidades suportadas pela biblioteca. Este fluxo de execução é comum à grande maioria das *operations* por constituir, um acesso à cache para obtenção dos dados ou um acesso aos serviços, através dos *providers*, e *parse* dos dados quando os mesmos não se encontram armazenados na cache. Tendo em conta a partilha do fluxo de execução foi usada a *pattern Template Method* para definir o fluxo de execução composto por um conjunto de passos que serão posteriormente implementados pelas *operations*, lógica especifica da operação. Esta opção permitiu não só reduzir o esforço de desenvolvimento como garantiu igualmente o correto funcionamento de forma uniforme por todas as *operations*.

[6] *NSError* é um objecto que possuí um domínio, um código e uma mensagem (opcional), permitindo assim fornecer informação relativa ao erro ocorrido.

Na Figura 19 encontra-se representada a implementação da *pattern* estando evidenciada a sua utilização por uma *operation* da biblioteca.

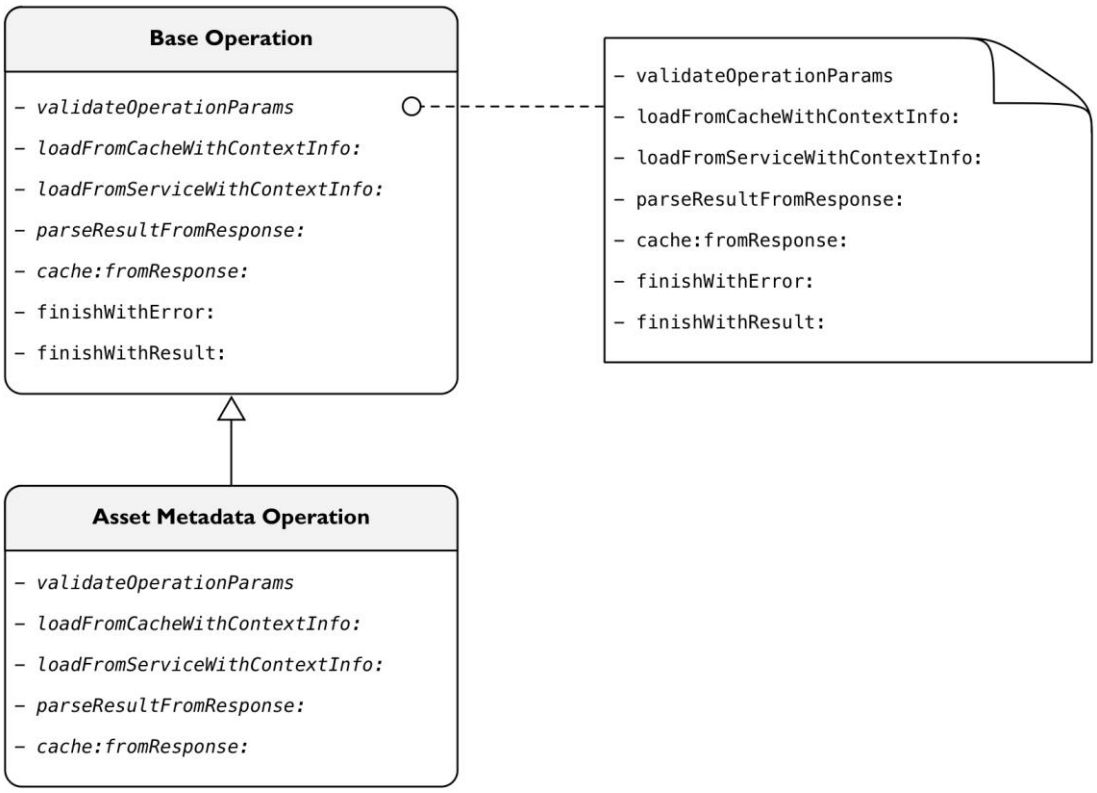

Figura 19 - Representação da utilização da *pattern Template Method*

Foi ainda criada uma *operation* base para operações que não tenham associados mecanismos de *caching* de modo a remover a necessidade destas operações implementarem estes métodos.

5.1.5. Operations Executor

O *operations executor* representa o *engine* responsável por receber todas as *operations* da biblioteca, efetuar o seu agendamento e consequente execução de forma assíncrona.

Internamente contém duas *operation queue*, uma para a obtenção de dados e uma outra para obtenção de imagens. Esta divisão visa minimizar o impacto do acesso a imagens, realizados em quantidades massivas pelos clientes, do acesso a dados que possuem comparativamente uma maior prioridade. As *operations* possuem uma propriedade, *queuePriority*, que permite especificar a respetiva prioridade de execução na *queue*, o que poderia justificar a utilização de uma única *queue* com a devida especificação de prioridades, no entanto, tendo em conta a quantidade massiva de acesso a imagens num determinado instante os acessos a dados poderiam ficar em espera caso estivessem a ser processados vários pedidos de imagens naquele mesmo instante.

As *operations* representam lógica interna da biblioteca não ficando por isso disponíveis para os clientes, assim foi necessário criar um mecanismo de cancelamento disponível para os clientes poderem proceder ao cancelamento de *operations*, não perdendo desta

forma o controlo que estes componentes oferecem. Este mecanismo é fornecido pelo *Operations Executor* que para cada *operation* submetida gera um UUID[7] que identifica univocamente a *operation* ficando este identificador disponível para os clientes poderem proceder ao seu cancelamento. Internamente é mantido o mapeamento entre o identificador e a *operation* enquanto esta se encontra em espera ou execução, sendo este mapeamento removido e os respetivos recursos desalocados quando esta termina. Este mapeamento é mantido recorrendo a um dicionário, tabela *key-value*, existindo mecanismos de execução exclusiva pela sua invocação num ambiente *multi-thread*.

5.2. Aplicação TVE

O cliente móvel TVE é responsável por fornecer toda a *user experience* da produto, revelando uma importância crucial não só do ponto de vista técnico mas igualmente do ponto de vista visual.

5.2.1. Estrutura de navegação

A definição da estrutura foi o primeiro e um dos passos fundamentais no desenvolvimento do cliente móvel TVE por constituir toda a base para os desenvolvimentos futuros.

O sistema de navegação base da aplicação reside num menu lateral, componente que assumiu uma popularidade muito notória no ecossistema das aplicações móveis e permite uma navegação entre módulos de forma simples, eficiente e ao mesmo tempo de forma muito agradável do ponto de vista visual. Do ponto de vista funcional apresenta-se como uma vista que é revelada pelo deslocamento da vista atual no sentido positivo do eixo das positivas. Do ponto de vista técnico este menu é constituído por dois controlos, um controlo frontal onde é apresentado o conteúdo do módulo, *FrontViewController*, e um controlo traseiro que apresenta o menu de navegação, *RearViewController*. Estes dois componentes estão contidos pelo *RootViewController* responsável por realizar a sua gestão assim como realizar os movimentos para revelar e esconder o menu lateral.

Tendo em conta a proliferação deste componente com um elevado número de implementações *open-source* disponíveis foi usada uma destas soluções, o ZUUIRevealController[27].

[7] *Universally unique identifier*

Na Figura 20 podemos verificar a estrutura de navegação da aplicação evidenciando o menu no *RearViewController* que controla com o módulo a ser apresentado no *FrontViewController*.

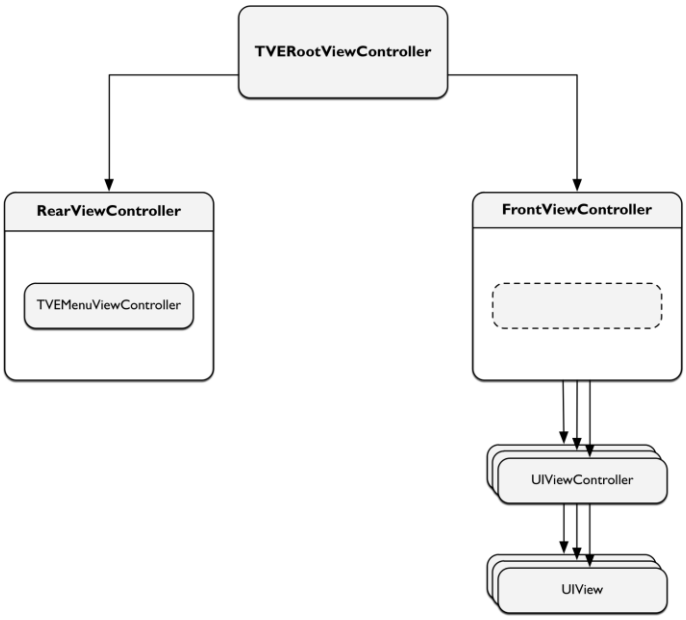

Figura 20 – Estrutura de navegação da aplicação

Na Figura 21 podemos visualizar o resultado final da navegação da aplicação evidenciando as componentes pertencentes ao *RootViewController*.

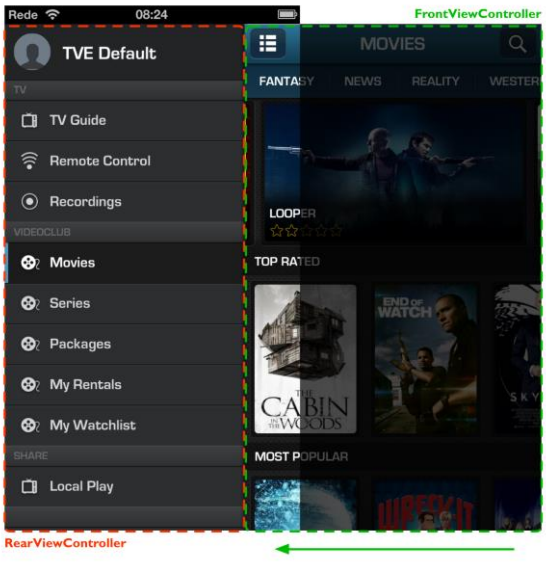

Figura 21 – Implementação da navegação na aplicação

5.2.2. Listas e Coleções

Os dispositivos móveis têm vindo a evoluir consideravelmente no seu poder de computação mas continuam ainda a apresentar várias limitações a nível de computacional assim como a nível de recursos. O fator mais crítico nestes dispositivos é a memória, sendo necessário constantemente adoptar mecanismos que permitam a redução da ocupação de memória.

A aplicação TVE, de modo a alcançar uma UI fortemente apelativa e agradável, foi desenhada para fazer um recurso expressivo a imagens que fornecem cor à aplicação. No entanto, esta utilização expressiva de imagens significa um aumento brutal de ocupação de memória.

Assim foi necessário adoptar mecanismos que permitam a reutilização de componentes permitindo assim reduzir a memória requerida pela aplicação. As listas e as coleções representam esses mecanismos.

Uma lista é um componente responsável por listar conteúdo de um determinado tipo sendo essa listagem realizada através de células responsáveis por apresentar o respetivo conteúdo. A lista apresenta no entanto, mecanismos para realizar uma reutilização de células evitando assim uma elevada alocação de componentes para um conjunto elevado de dados. O processo de reutilização tem em conta o número de células possíveis de serem visíveis no ecrã contemplando ainda uma margem de segurança, sendo depois feitas sucessivas deslocações à medida que as células deixam de estar visíveis para apresentarem os próximos elementos da lista. Estas listas são suportadas nativamente pelo SDK do iOS para uma disposição vertical tendo sido realizada uma extensão deste componente de modo a permitir igualmente listas horizontais que são usadas de forma muito frequente na aplicação.

Na Figura 22 é possível visualizar a utilização das listas na aplicação assim como a mecânica associada na reutilização das células.

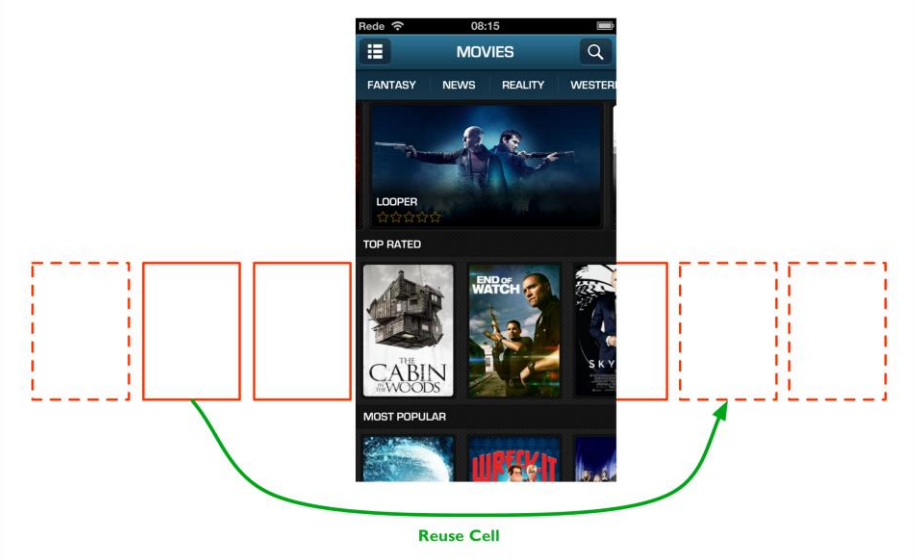

Figura 22 - Utilização de uma lista na aplicação

Uma coleção partilha muitas semelhanças com uma lista, contém igualmente mecanismos de reutilização de células, mas apresenta um método de disposição dos

seus componentes bastante diferente através de um preenchimento horizontal e vertical formando assim uma tradicional grelha com linhas e colunas. Este componente existe nativamente a partir do iOS6, tendo sido usado um projeto *open-source*, PSTCollectionView[28], que fornece uma implementação para o iOS5 e que permite detetar qual a versão do sistema operativo de modo a utilizar o componente nativo, se possível.

Estes componentes apresentam uma importância fundamental na aplicação por permitirem a disposição de conteúdo sem com isso esgotar por inteiro a memória do dispositivo sendo usados exaustivamente em toda a aplicação.

5.2.3. Reutilização de componentes e TVEUIFactory

Os dispositivos iPhone e iPad partilham de muitas semelhanças no seu modo de funcionamento assim como a nível de componentes, podendo ser atingido um elevado grau de partilha de desenvolvimentos para ambas as plataformas.

Assim, durante os desenvolvimentos existiu sempre o cuidado de identificar o conjunto de código comum a ambas as plataformas tratando depois das especificidades de cada dispositivo nomeadamente a customização dos tamanhos dos componentes. Foi adoptada a *creational pattern Abstract Factory* tendo em vista alcançar uma determinada abstração na instanciação dos componentes independentemente do dispositivo sendo determinada apenas *runtime* consoante a *factory* inicializada. Através desta *pattern* eliminou-se a necessidade de possuir código específico para a inicialização de componentes do respetivo dispositivo.

Na Figura 23 podemos visualizar um exemplo do cuidado tido na concepção não só da interface gráfica mas principalmente dos componentes que são reutilizados entre os dois dispositivos, promovendo a reutilização de código tendo como consequência a redução do tempo de desenvolvimento da aplicação assim como uma maior qualidade do software, tendo sido o principal objetivo que justificou esta opção que acabou por oferecer óptimos resultados.

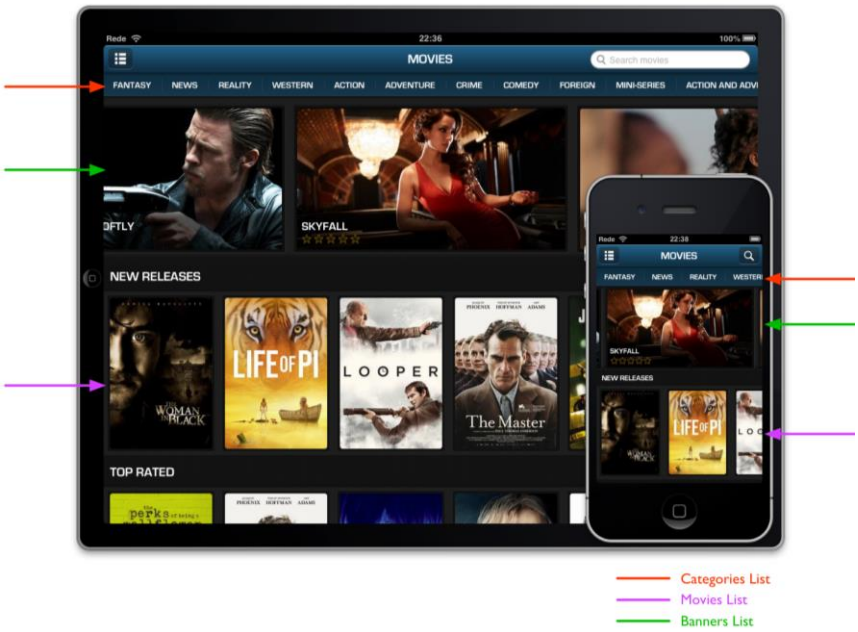

Figura 23 - Reutilização de componentes entre iPhone e iPad

5.2.4. Assincronismo

O assincronismo fornecido pela biblioteca através da API pública remove totalmente a preocupação e a necessidade da aplicação recorrer a *threads* secundárias para a execução de determinado pedido de modo a manter a *main-thread* disponível para todas as operações de UI, estando este mecanismo embutido na própria biblioteca.

No entanto, quando é necessária a execução de um conjunto de operações em paralelo este assincronismo não nos permite perceber quando todas as operações terminaram para atualizar o cliente com os todos os resultados obtidos. Esta limitação foi resolvida recorrendo à *pattern Observer* e um contador de operações ainda em execução, este contador era inicializado com o número de operações a executar e era sucessivamente decrementado quando uma terminava, até atingido o valor zero significando que todas as tarefas tinham terminado. Esta abordagem resolveu a limitação do assincronismo da API pública mas apresentou um outro inconveniente: cada controlador que realize múltiplas operações em paralelo precisa proceder à inicialização do *observer* assim como gerir o mecanismo de decremento do contador.

Tendo em conta que a aplicação requer frequentemente a execução de múltiplas operações em paralelo foi desenhado um novo mecanismo, um *Batch Job*. Este componente recebe um conjunto de operações a executar sob a forma de *callbacks* procedendo à sua invocação quando é iniciado, guardando internamente o UUID de cada invocação. Recorrendo à *pattern Publish-Subscribe*, o *Operations Executor* procede ao envio de uma notificação quando uma operação termina, enviando o respetivo UUID, sendo essa notificação recebida pelo *Batch Job* que irá verificar se coincide com uma das suas operações e assim consegue gerir quais as operações que terminaram e as que ainda se encontram em execução. Após todas terem terminado procede à invocação de um *callback* registado pelo controlador que inicializou o *Batch Job*.

Através deste componente foi resolvida a limitação do assincronismo da API pública de uma forma simples e poderosa visto ser possível com base no *Batch Job* criado proceder a especializações para execução de operações de forma sequencial ou mesmo o estabelecimento de dependências entre operações.

Na Figura 24 é possível visualizar o fluxo de execução de um *Batch Job*.

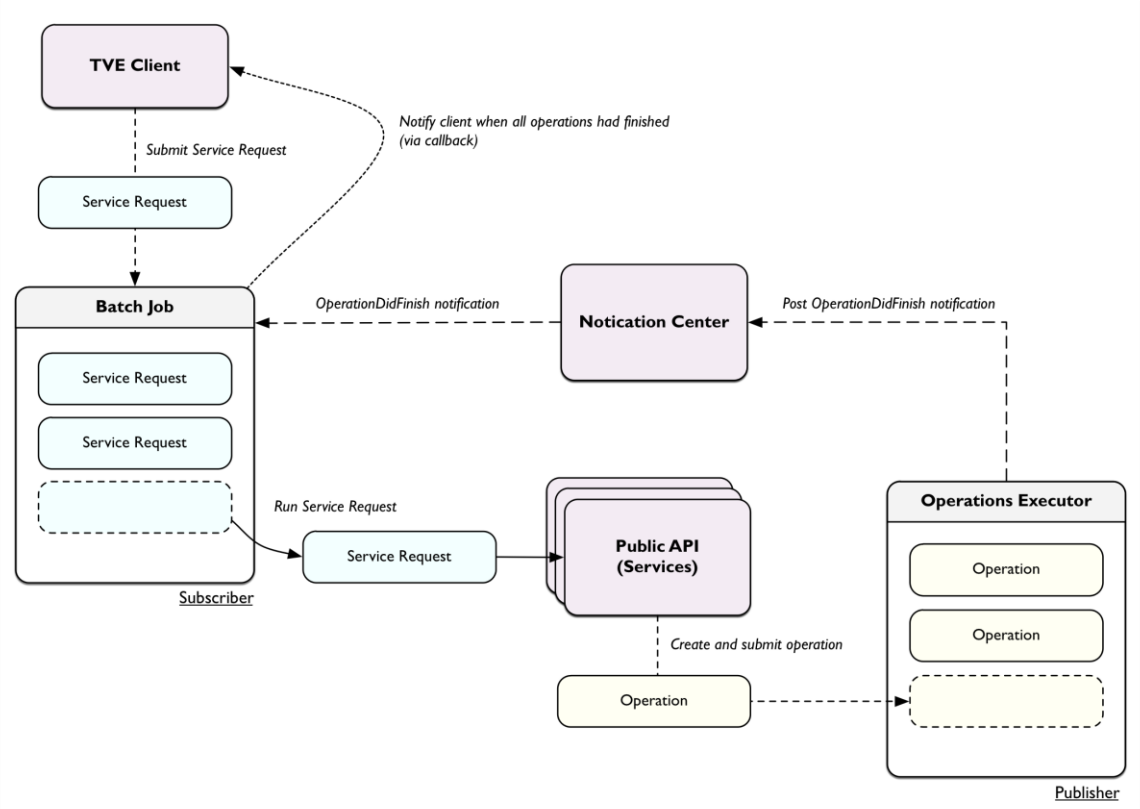

Figura 24 - Fluxo de execução de um Batch Job

5.2.5. TV Guide iPad

O desenvolvimento de uma grelha de programação dinâmica e contínua envolve uma séries de questões do ponto de vista de performance tendo em conta o elevado número de canais por vezes disponíveis que por sua vez têm associados um elevado número de programas associados. Tendo em conta o dispositivo alvo ser um dispositivo móvel com um poder de computação limitado, a questão de performance torna-se ainda mais relevante.

A abordagem para fornecer uma navegação dinâmica e contínua passou pela criação de uma vista com o tamanho equivalente ao nº de canais disponíveis e o nº total de dias a apresentar. No entanto, tornou-se evidente a impossibilidade de desenhar toda a grelha de programação e mantê-la em memória muito por conta da apresentação da imagem de cada programa na grelha. Assim foi criado um mecanismo de desenho dinâmico que tem como base o toque do utilizador no ecrã, sendo que quando este termina calcula o conjunto de canais visíveis assim como o conjunto de dias a apresentar procedendo depois ao desenho dos programas respetivos. Tendo em conta a limitação de memória destes dispositivos são ainda tomadas medidas pró-ativas de libertação de memória procedendo à libertação dos programas desenhados que estejam a cima de um determinado limiar. Este componente tem igualmente em conta as notificações do sistema operativo relativas à falta de memória do dispositivo procedendo à libertação de programas que não estejam visíveis.

Na Figura 25 podemos visualizar as duas listas que fornecem a base deste componente, a lista de canais e a grelha com os vários programas a apresentar.

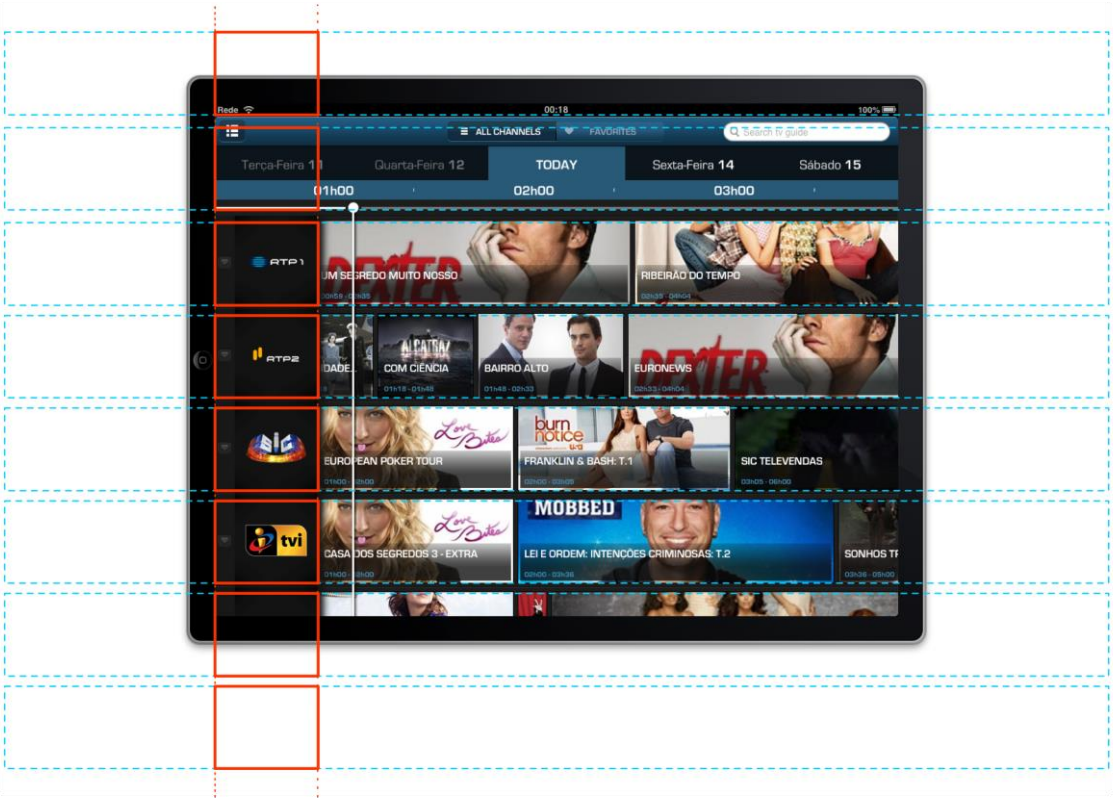

Figura 25 – Listas de suporte ao TV Guide no iPad

5.3. Módulo Local Play

O módulo Local Play visa disponibilizar a reprodução de conteúdo armazenado no dispositivo diretamente na televisão recorrendo às STB instaladas nas casas dos clientes, funcionando como um complemento à experiência *TV Everywhere* e que oferece um acréscimo de valor no produto final.

Um vídeo representa um conjunto de dados comprimidos por um *codec* sob a forma de *frames*, P, I, B, armazenadas posteriormente por um *container* sob a forma de pacotes. Associado a um conteúdo de vídeo está igualmente um protocolo de transporte quando é visada a sua transmissão.

As STB do Mediaroom apresentam um suporte deficitário a nível de *codecs* de vídeo (H.264, VC1, MPEG2), *containers* de vídeo (MP4, MPEG2-TS) e protocolos de transporte (HTTP). O iOS por sua vez usa o *codec* H.264 e o *container* MOV para armazenamento dos vídeos gravados pelo dispositivo, sendo assim necessário efetuar processamento para resolver a incompatibilidade do *container*.

Para o efeito foi utilizado o FFmpeg[29], biblioteca *standard* na indústria do vídeo, escrita em C, possui suporte a um vasto leque de *codecs, containers* entre outras funcionalidades. Esta foi aliás a principal razão da sua opção, visto possuir um imenso suporte que permite alcançar uma determinada flexibilidade relativamente ao *codec*

assim como ao *container* usado, permitindo no futuro suportar outras STB com um conjunto de requisitos de vídeo diferentes. Os outros fatores que pesaram na sua adopção consistem na maturidade já alcançada, encontrar-se em ciclos de desenvolvimento ativos com uma larga comunidade e finalmente pela sua forte utilização na área de vídeo. De modo a permitir a sua utilização pela aplicação foi necessário conduzir à sua compilação para a arquitetura ARM.

O processamento realizado pela aplicação passa pela obtenção das respetivas *frames* de vídeo armazenadas pelo *container* MOV e proceder ao seu armazenamento sob o *container* MPEG-TS suportado pela STB, não sendo necessário conduzir um processo de *transcoding*, fortemente exigente do ponto de vista computacional, por existir suporte ao *codec* H.264.

A aplicação possui igualmente um servidor HTTP embutido de modo a disponibilizar o conteúdo de vídeo à STB, tirando partido da comunicação pré-estabelecida com a STB é possível enviar-lhe um comando para proceder à abertura de um URL específico, neste caso no próprio dispositivo a fim de obter o respetivo contéudo.

O iOS com a política de *sandboxing* não permite o acesso por parte das aplicações a conteúdo fora da *sandbox* da própria aplicação, o que impossibilita o acesso direto ao conteúdo do dispositivo sendo necessário recorrer a APIs fornecidas pelo SDK para esse mesmo efeito. Assim, foi necessário operar ao nível da memória efetuando sucessivas leituras de dados (bytes) usando as APIs do SDK que depois recorrendo ao FFmpeg são convertidos para o *container* final sendo posteriormente enviados através do *socket* HTTP estabelecido com a STB.

Na Figura 26 é possível visualizar o fluxo de dados relativo ao módulo Local Play.

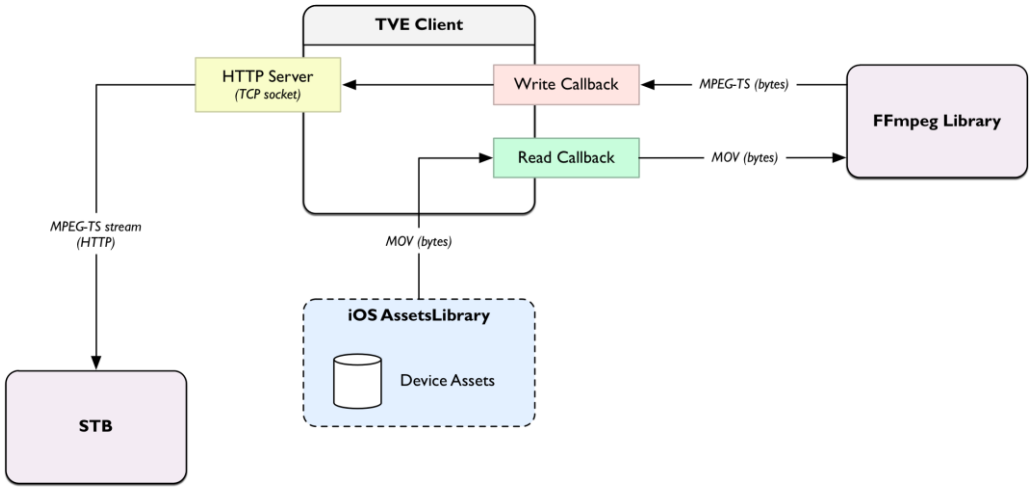

Figura 26 – Fluxo de dados relativo ao módulo Local Play

5.4. Benchmark de parsers XML e JSON

A atual plataforma *TV Everywhere* usa o formato XML para codificação dos dados transmitidos para os seus clientes não existindo suporte para outros formatos de dados. De modo a suportar este tipo de dados foi conduzido um processo de *benchmarking* para proceder à escolha do *parser* a usar pela biblioteca, tendo ainda sido conduzido paralelemente um comparativo entre formatos de dados, XML e JSON, de modo a identificar as diferenças de performance entre os dois formatos e qual dos dois se apresenta como mais apropriado para aplicações iOS. Esta análise pode ser consultada com maior detalhe no Anexo E.

Este processo de *benchmark* teve em conta as bibliotecas mais populares para cada um dos formatos de dados, tendo sido criados dois casos de teste distintos de modo a obter uma maior relevância dos dados obtidos. Estes casos de teste têm em conta o suporte a atributos e elementos por parte do formato XML que leva a diferenças significativas no espaço ocupado pelos dados como é possível visualizar na Figura 27.

```
// atribute-based
<book isbn="1449396097" title="Redis: The Definitive Guide"/>

// elemento-based
<book>
  <isbn>1449396097</isbn>
  <title>Redis: The Definitive Guide</title>
</book>
```

Figura 27 - Comparativo entre as variantes do formato XML

Assim foram definidos dois casos de teste:

- Dataset A – representa o top400 da iTunes Store que disponibiliza estes resultados em ambos os formatos, JSON e XML. No formato XML são usados atributos assim como elementos para representação dos dados.
- Dataset B – *dataset* gerado por um script Ruby escrito para este propósito, gerando um inventário de livros de uma biblioteca usando permutações sobre um conjunto de 5 livros definidos. O script gera dados para o formato JSON e XML, usando para este última apenas atributos para representação dos dados.

O processo de *benchmarking* foi conduzido em 4 dispositivos com diferente hardware e software de modo a obter um espectro de resultados mais abrangentes e relevantes. De seguida são apresentados os resultados obtidos num comparativo entre o formato XML e JSON.

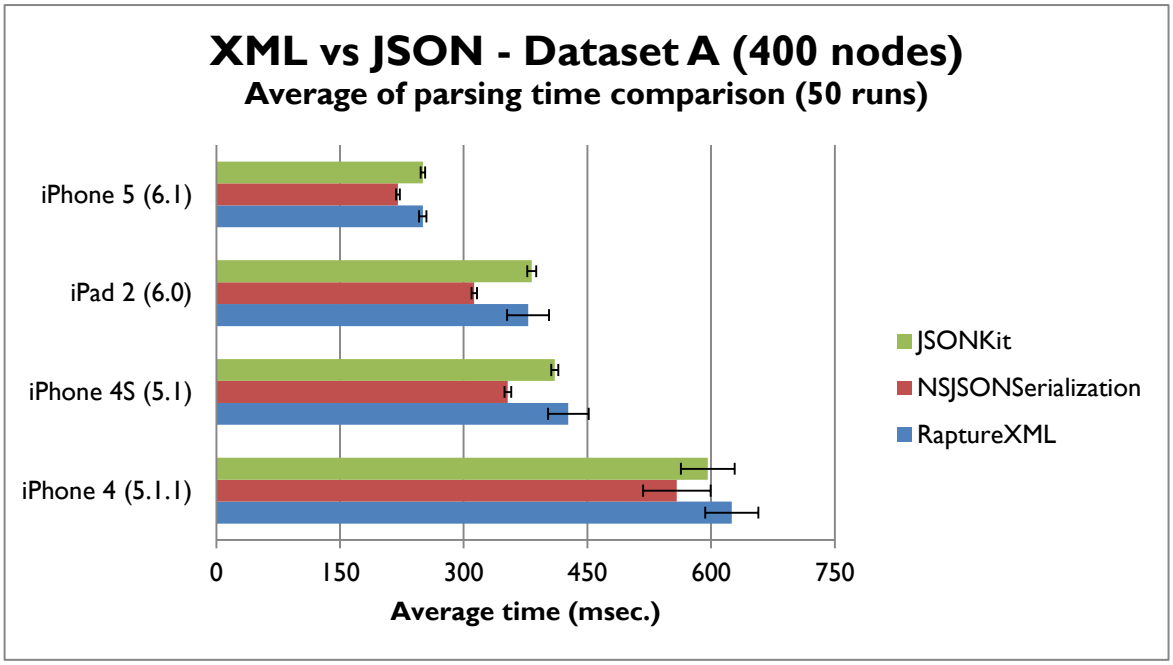

Figura 28 - Resultados do benchmark de parsers para o *dataset* A

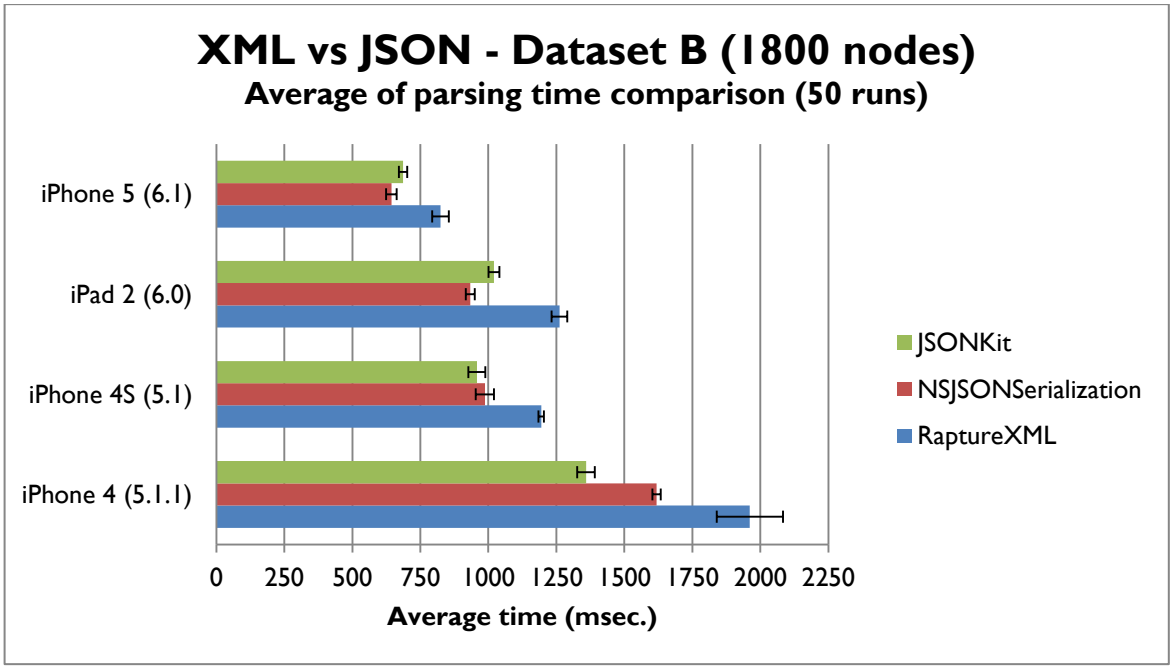

Figura 29 - Resultados do benchmark de parsers para o *dataset* B

Os resultados revelam uma vantagem para o formato JSON em relação ao formato XML, sendo possível constatar que o *parser* RaptureXML apresenta resultados muito próximos dos resultados dos *parsers* JSON. Estes resultados mostraram que existe um ganho significativo no suporte ao formato JSON pela plataforma tendo sido incluído no *roadmap* da plataforma o suporte para este formato. Posteriormente será necessário implementar o suporte ao formato JSON por parte da aplicação sendo o impacto desta alteração significativamente baixo graças ao modelo em camadas adoptado onde toda a esta lógica se encontra reunida na camada de acesso a dados. Atualmente a biblioteca

usa o *parser* RaptureXML por apresentar dos melhores resultados para o formato XML e por apresentar uma API muito simples que que simplifica consideravelmente o tempo de desenvolvimento.

5.5. Reprodução de conteúdo protegido

A aplicação TVE pretende permitir um fácil e rápido acesso ao conteúdo de vídeo do operador por parte dos seus clientes promovendo a sua aquisição assim como a respetiva visualização. Esta visualização do conteúdo apresenta no entanto várias restrições de segurança de modo a proteger a propriedade intelectual dos seus autores assim como o acesso indevido ao conteúdo do operador. Estas restrições são resultantes do impacto que a pirataria tendo vindo a adquirir na disponibilização ilegal de conteúdo de vídeo.

Torna-se assim essencial que a aplicação venha a suportar um sistema DRM, *Digital rights management*, que forneça suporte e acesso a conteúdo protegido. Este conteúdo é encriptado previamente pelo emissor, transferido de forma segura sendo a desencriptação conduzida depois por parte do cliente a quando da sua reprodução. O conteúdo armazenado no dispositivo é guardado de forma encriptada sendo realizada a sua desencriptação no momento de reprodução, ocorrendo previamente validações de modo a garantir que o acesso é permitido.

Tendo em conta que existem várias soluções deste tipo no mercado não é viável a integração das várias soluções na aplicação TVE, realizando-se esta integração posteriormente tendo em conta a solução usada pelo operador. No entanto, tendo em conta a importância desta componente para o sucesso do cliente, foi conduzida uma integração de teste com uma solução atualmente no mercado, o cliente DRM da empresa Discretix[30]. Esta integração ocorreu com sucesso tendo sido alcançada a reprodução de conteúdo vídeo protegido disponível na plataforma da Discretix. Esta integração não visou um incremento no produto mas sim demonstrar a aptidão e a capacidade de proceder à integração futura destes sistemas na aplicação TVE.

5.6. Ferramentas de suporte ao negócio

Tendo em conta o público alvo da aplicação TVE serem os clientes do operador que recorrerão à aplicação para consulta de conteúdo assim como aquisição do mesmo, torna-se importante recolher métricas sobre qual o conteúdo mais relevante da aplicação, quais as funcionalidades mais utilizadas pelos utilizadores, número de clientes ativos assim como a sua distribuição geográfica, entre um conjunto vasto de métricas.

Assim, de modo a fornecer um incremento de valor ao produto foi introduzido na aplicação o Google Analytics[31] para dispositivos móveis. A opção pelo Google Analytics derivou do vasto conjunto de funcionalidades oferecidas, fornecer o *back-office* para consulta das métricas não sendo necessário existir desenvolvimentos deste tipo e por ser uma solução robusta e gratuita com a selo de qualidade Google. Esta integração permite-nos recolher métricas relativas a:

- Ecrãs visitados pelos utilizadores;
- Informação relativa ao conteúdo partilhado pelos utilizadores;

- Eventos da aplicação como o pressionar de um botão;
- Informação relativa a sessões;
- Informação sobre as transações realizadas pelos utilizadores;
- Informação de *crashes* e exceções.

Estas métricas revelam não só uma importância do ponto de vista comercial mas igualmente do ponto de vista de experiência de utilização da aplicação possibilitando realizar inferências sobre quais os componentes menos populares e conduzir esforços para melhorar a usabilidade da aplicação.

No anexo G é possível consultar com maior detalhe a análise realizada às várias ferramentas.

5.7. Ferramentas de controlo de qualidade

O aplicação TVE irá correr num número elevado de dispositivos diferentes e apesar do iOS ser o sistema operativo com a maior taxa de atualização do sistema operativo, é inevitável que acabará por correr em múltiplas versões. O hardware e principalmente o software, apresentam diferenças entre versões sendo necessário conduzir testes nas suas várias versões para detetar as particularidades existentes e tratá-las a fim de evitar a ocorrência de exceções na aplicação.

É importante possuir um mecanismo de alertas e recolha de métricas que permita auxiliar o controlo de qualidade da aplicação, tendo sido integrado na aplicação a solução Crashlytics[32]. Esta solução visa recolher os *traces* associados a um *crash* ou exceção para posterior consulta, permitindo assim realizar um acompanhamento constante da aplicação de modo a detetar anomalias e realizar a sua respetiva correção. O Google Analytics também é capaz de recolher esta informação tendo, no entanto, sido optado pela integração do Crashlytics de modo a possuir um mecanismo dedicado e consequentemente mais focado e com maior número de funcionalidades neste aspeto de elevada importância para o produto.

No anexo H é possível consultar com maior detalhe a análise realizada às várias ferramentas.

5.8. MediaProxy

Na fase final dos desenvolvimentos houve a necessidade de desligar o MediaProxy usado pelas aplicações devido à carga que estava a introduzir na máquina onde se encontra alojado e que é partilhada por outros projetos. Como consequência veio-se a constatar um aumento radical da memória utilizada pela aplicação para níveis demasiado altos que não permitem garantir a sua estabilidade e fluidez.

O MediaProxy foi implementado usando a linguagem C# e possui a dependência do servidor web da Microsoft, Internet Information Services (IIS), levando à dependência da plataforma Microsoft Windows para conduzir ao seu alojamento. Assim houve uma tomada de iniciativa pelo estagiário no desenvolvimento de uma nova versão do MediaProxy com dois objetivos em mente: garantir um suporte multiplataforma assim

como manter toda a interface e funcionalidades existentes. Tendo em conta este ser um componente central da plataforma estando responsável por uma grande percentagem de tráfego foi igualmente tido em mente garantir a escalabilidade da solução.

5.8.1. Motivação

A motivação para o desenvolvimento do módulo MediaProxy derivou do aumento de performance possível de alcançar pelas aplicações através da sua utilização para os acessos a imagens.

Na figura seguinte podemos observar o consumo de memória de dois ecrãs relativos ao VOD onde o conteúdo é apresentado recorrendo exaustivamente a imagens, onde num primeiro cenário não é feita utilização do MediaProxy sendo as imagens obtidas e desenhadas no seu tamanho original[8] e um outro cenário onde as imagens são obtidas com o respetivo tamanho com que serão apresentados no componente.

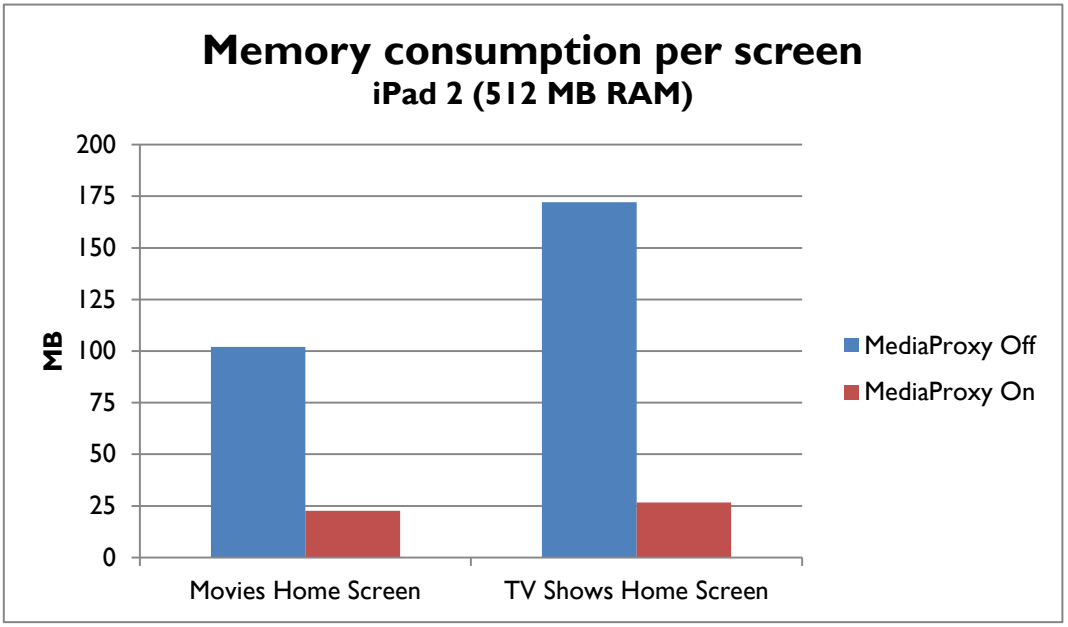

Figura 30 - Consumo de memória por ecrã com e sem utilização do MediaProxy

Na Figura 30 é visível que não existindo um controlo nas imagens apresentadas o aumento dos recursos é muito significativo o que acabará por comprometer a performance e a fluidez da aplicação. Para além da redução da memória consumida é ainda possível de reduzir a largura de banda consumida e consequentemente reduzir o tempo de transferência o que acaba por permitir garantir uma experiência de utilização mais fluída.

5.8.2. Arquitetura

Com o objetivo de garantir a escalabilidade da solução foi definida a utilização da tecnologia Node.js que segue uma abordagem *event-driven* onde as operações de I/O são conduzidas de uma forma assíncrona libertando o *event-loop* responsável por receber e processar todos os pedidos dos clientes, o que se apropria para o desenvolvimento de

[8] Resolução 1000 x 1500

um *proxy* onde cada pedido envolve o estabelecimento de uma nova ligação pretendida pelo cliente. Esta abordagem apresenta benefícios do ponto de vista de escalabilidade pela redução do *overhead* de criação de *threads* ou processos existindo apenas uma única *thread* com um *event-loop* que processa todos os clientes e executa as operações de longa duração, como o são as operações de I/O, numa *thread-pool* interna usando callbacks de resposta quando estas terminam a sua execução.

Para fornecer operações de redimensionamento de imagens, um dos objetivos do MediaProxy, foi analisado um conjunto de soluções tendo a escolha recaído na utilização de uma das bibliotecas *standard* no processamento e manipulação de imagens, a biblioteca GraphicsMagick[33]. Esta biblioteca em conjunto com a ImageMagick[34], sua originária, representam a referência no processamento de imagens tendo sido optado pela utilização da GraphicsMagick pelo suporte à biblioteca OpenMP[35] que permite realizar uma paralelização das operações reduzindo o custo temporal das operações e tirando partido de todo o hardware da máquina.

Tendo em conta que as operações de redimensionamento de imagens são operações com elevado custo computacional foi igualmente integrada uma cache LRU de modo a permitir uma reutilização de imagens e assim aumentar a capacidade do sistema.

Na Figura 31 é possível analisar a arquitetura do sistema.

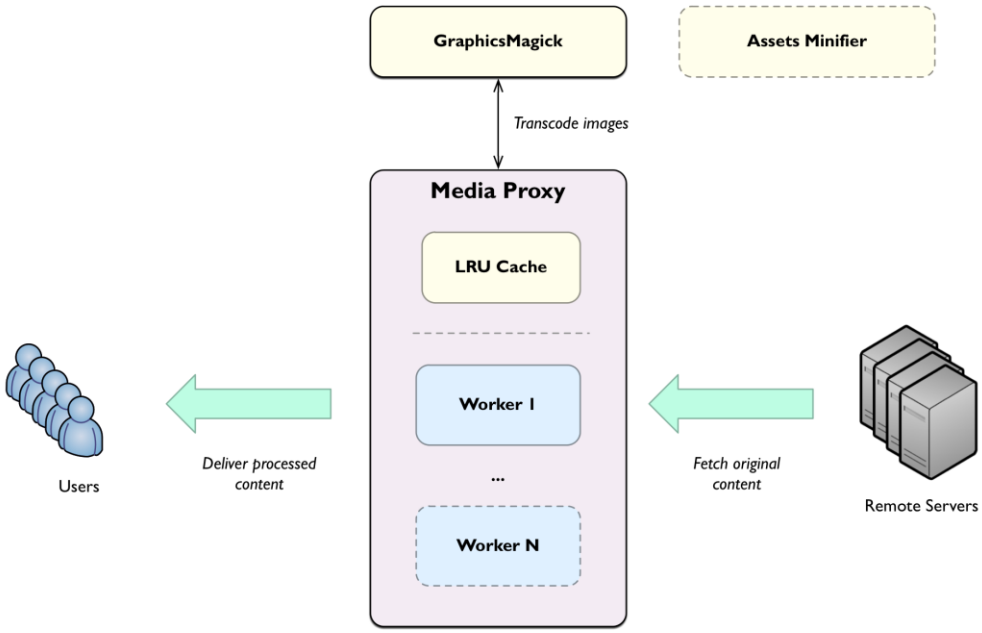

Figura 31 - Arquitetura do MediaProxy

Tendo em conta a arquitetura do Node.js ser *single-thread* são usadas múltiplos *workers* de modo a tirar partido de todos núcleos do processador comunicando entre si através da técnica *Inter-Process Communication* (IPC), sendo esta funcionalidade disponibilizada através de APIs nativas.

Outra das componentes que será incluída com o objetivo de estender e completar o MediaProxy passa pela adição de um módulo de minificação que poderá executar este processo para ficheiros HTML, Javascript e CSS de modo a diminuir o tamanho destes

recursos de uma forma transparente e garantindo uma redução da largura de banda consumida, podendo ser usado pela futura aplicação

5.8.3. Resultados de performance

De modo a proceder a uma primeira avaliação da performance do MediaProxy desenvolvido foram conduzidos um conjunto de testes para avaliar a sua performance para um diferente número de ligações em simultâneo, tendo sido posteriormente conduzido um teste de carga mais extenso de modo a analisar se o sistema consegue manter o número de pedidos por segundo e o respetivo tempo de resposta ao longo do tempo ou existe uma degradação destes números. Os resultados destes testes podem ser consultados de forma mais pormenorizada no Anexo I.

De modo a identificar a qualidade da ligação de rede, rede gigabit, foi conduzido um teste de carga cujos resultados estão disponíveis na Tabela 16 e onde é possível visualizar que a largura de banda está dentro dos valores esperados para uma rede gigabit.

Cliente	Servidor	Throughput (MB/s)
Servidor de Carga	Proxy	115.64
Proxy	Servidor de imagens	118.35

Tabela 16 - Largura de banda da ligação entre as máquinas do ambiente de testes

Na Tabela 17 podemos visualizar a performance do sistema para a operação de redimensionamento de uma imagem, com uma resolução elevada, para os tamanhos mais comuns da aplicação nas versões retina (HD) e não-retina (SD). Estes resultados foram obtidos através da variação do número de ligações em simultâneo de modo a identificar a capacidade de resposta do sistema.

Redimensionamento	Req/Sec	Throughput (MB/s)
1200 x 1600 ⇒ 160 x 242	6	0.085
1200 x 1600 ⇒ 360 x 484	6	0.3

Tabela 17 - Resultados das operações de redimensionamento sem cache

Na Tabela 18 são apresentados os resultados para o teste de carga mais extenso com uma duração de 10 minutos, 75 ligações em simultâneo usando o mecanismo *keep-alive* e com o conteúdo já disponível em *cache*.

Redimensionamento	Req/Sec	Throughput (MB/s)
1200 x 1600 ⇒ 160 x 242	6787	98.15
1200 x 1600 ⇒ 360 x 484	2280	110.5

Tabela 18 – Resultados das operações de redimensionamento com cache

Analisando os resultados podemos analisar que o módulo desenvolvido para um conjunto de 75 ligações tirando partido da *cache* consegue praticamente saturar a ligação de rede conseguindo assim retirar o máximo do *hardware* disponível, estando o número de pedidos por segundo diretamente relacionado com a quantidade de dados a enviar e a capacidade da ligação de rede.

A utilização de uma *cache* interna apresenta-se como fundamental por permitir a reutilização das operações de redimensionamento realizadas previamente e que como podemos ver na Tabela 17 apresentam um custo computacional muito elevado, permitindo assim conferir ao sistema a capacidade de lidar com o conteúdo mais popular sem com isso comprometer drasticamente a performance do sistema.

O MediaProxy apresenta-se como uma componente importante na performance das aplicações por possibilitar diminuir a largura de banda consumida assim como reduzir drasticamente os recursos consumidos, nomeadamente a memória, por possibilitar a disposição das imagens no tamanho esperado pelo componentes. É uma solução mais poderosa comparativamente ao processo de redimensionamento prévio de todas as imagens para tamanhos definidos, grande, médio e pequeno, por existirem diferentes tamanhos de ecrã disponíveis o que não possibilita uma adaptação em pleno ao tamanho esperado, apesar de permitir a redução da largura de banda e os recursos consumidos. A utilização do MediaProxy apresenta-se como a solução mais poderosa mas que apresenta igualmente um conjunto de desafios do ponto de vista computacional e de escalabilidade, sendo possível minimizá-los com a utilização de uma *cache* e assim garantir a viabilidade do sistema. A utilização da tecnologia Node.js permitiu igualmente conferir à solução desenvolvida um determinado nível de escalabilidade tendo em conta o volume de tráfego que permitiu suportar utilizando exaustivamente o mecanismo de *caching* para disponibilização das respostas aos pedidos.

Capítulo 6
Controlo de qualidade

O desenvolvimento e implementação do projeto não garante por si só o sucesso e o cumprimento do projeto devendo existir um constante controlo de qualidade a fim de validar e garantir a qualidade do software a ser desenvolvido. Neste capítulo são apresentadas os vários processos que visaram garantir a qualidade do software desenvolvido.

6.1. Static Analyzer

As aplicações iOS são aplicações compiladas recorrendo a um compilador, Apple LLVM Compiler, que garante desde logo uma primeira validação de eventuais erros existentes no código através de uma análise sintática e semântica. No entanto, tendo em conta a função dos compiladores ser a geração de um binário num curto espaço de tempo esta componente de análise do código não se apresenta como profunda e exaustiva, existindo ferramentas especializadas apenas nessa análise, o denominado *static analyzer*.

Durante os desenvolvimentos foi feito recurso ao *static analyzer* fornecido nas ferramentas de desenvolvimento, Clang Static Analyzer, que procede a uma análise detalhada de código tendo em consideração aspetos como desalocações de objetos em falta que conduzem a *memory-leaks*, o cumprimento das *guidelines* de código especificadas pela Apple, variáveis inicializadas e não utilizadas entre outros aspectos. Esta análise é importante sendo feita periodicamente de modo a identificar possíveis problemas nos desenvolvimentos podendo ser prontamente resolvidos contribuindo ativamente para a garantia de qualidade do software.

6.2. Continuous Integration

O sucesso de um projeto de software desenvolvido em equipa está intrinsecamente dependente não só dos processos de desenvolvimento dos vários componentes mas sobretudo pelo processo de integração e interligação desses mesmos componentes.

Tendo em conta a arquitetura da aplicação TVE, composta pela aplicação e a biblioteca, torna-se evidente a necessidade de garantir e validar o processo de integração da aplicação com a biblioteca e que por sua vez implica uma correta integração das várias subcomponentes constituintes, estando o sucesso do projeto diretamente dependente destas integrações tendo em conta a aplicação requerer a biblioteca para a execução de todas as operações.

Tendo isto em conta, a utilização de um sistema de controlo de versões, SVN, garantiu desde logo um mecanismo que permite uma integração constante dos desenvolvimentos dos vários componentes do projeto. No entanto, tendo em conta a importância do projeto, foi adoptado um mecanismo de integração contínua através de um *build server* responsável pelo processo de compilação do projeto assim como geração dos respetivos binários. A escolha do *build server* recaiu sobre o *Jenkins*[36] pela

popularidade que adquiriu após o *fork* do projeto *Hudson*[37] tendo também tido peso na decisão ser usado pela empresa noutros projetos possibilitando um suporte interno.

Assim foi criado um *script* que procede ao processo de *checkout* da versão mais recente do repositório SVN procedendo-se ao processo de compilação da biblioteca seguido da aplicação, sendo no final gerado o binário correspondente em caso de sucesso, usado posteriormente para as várias demonstrações do projeto. Em caso de falha, os responsáveis são notificados via email com informação sobre a falha ocorrida.

Este processo apresenta uma elevada importância por permitir garantir a estabilidade do projeto de uma forma contínua tendo em conta os sucessivos desenvolvimentos realizados assim como simplifica o processo de geração de um binário podendo ser conduzido por qualquer pessoa sem com isso sem requerer todo o ambiente de desenvolvimento.

6.3. Testes de software

Em conjugação com os desenvolvimentos foram conduzidos sucessivos testes ao software pela equipa de Controlo de Qualidade da empresa. Este controlo foi realizado pelo elemento André Quaresma que se encontra alocado à unidade de TV com a responsabilidade de efetuar todas os testes de aceitação do software no âmbito da unidade, incluindo assim a aplicação do presente estágio.

Estes testes de software foram sucessivamente conduzidos ao longo dos *sprints* de desenvolvimento de modo a avaliar e validar as respetivas componentes a serem desenvolvidas, tendo-se revelado de elevada importância pelo estabelecimento de um processo que permitiu continuamente garantir o cumprimento dos objetivos traçados com a qualidade exigida ao produto. O presente estágio alcançou assim o esperado enquadramento com os processos seguidos na empresa.

Durante os desenvolvimentos foi igualmente estabelecido um controlo de qualidade do software recorrendo a testes unitários que validam um determinado conjunto de componentes. Estes testes tornam-se de particular relevância pelo desenvolvimento de uma biblioteca que suporta todas as aplicações, responsáveis apenas pela componente de UI, sendo necessário garantir a sua estabilização assim como o seu correto funcionamento.

Apesar da importância destes testes, a elevada volatilidade dos requisitos do projeto conduziu a sucessivas alterações da biblioteca num curto espaço de tempo, o que provocou um défice no suporte desta componente existindo atualmente testes para as componentes mais críticas da biblioteca. No entanto, tendo em conta a importância da biblioteca para o projeto pretende-se futuramente estender o suporte a este tipo de testes às restantes componentes.

6.4. Demonstrações

Ao longo dos desenvolvimentos o projeto foi sucessivamente avaliado no seio da empresa com demos frequentes, tendo permitido uma recolha de feedback que possibilitou um contínuo melhoramento do produto. Para além destas demos internas, o produto foi sucessivamente apresentado em feiras internacionais, como a feira TV

Connect[38] em Londres, assim como a clientes internacionais, o que se revelou muito importante para validação e realizar um constante refinamento ao produto.

6.5. Testes de internacionalização

Tendo em conta os clientes alvo do produto serem operadores internacionais houve a necessidade e a preocupação de garantir a internacionalização do conteúdo textual da aplicação assim como do conjunto de dados fornecidos pela plataforma, visto existir suporte para dados internacionalizáveis. Assim foi usado o mecanismo nativo do iOS, o qual usa ficheiros com um conjunto de itens chave-valor para cada uma das línguas suportadas, sendo utilizado o ficheiro correspondente à língua do dispositivo ou, em caso de falta de suporte, a língua padrão definida.

Ao longo dos desenvolvimentos foram feitos sucessivos testes para garantir a internacionalização dos vários componentes, tornado assim o processo de suporte a uma nova língua simples por requerer apenas a criação do respetivo ficheiro com o conjunto de chaves definidas e a respetiva tradução para a nova língua.

Capítulo 7
Conclusões e trabalho futuro

O presente estágio possibilitou um inúmero conjunto de oportunidades de aprendizagem e desafios, tendo sido extremamente gratificante do ponto de vista profissional assim como pessoal. É possível assumir que os objetivos traçados foram devidamente alcançados sob a forma de uma aplicação para a plataforma iOS que irá constituir o produto das soluções *TV Everywhere* da empresa.

O desenvolvimento de uma aplicação móvel apresenta constantes desafios técnicos, apresentando um paradigma próprio. Existe a necessidade constante de monitorizar os recursos utilizados de modo tirar partido de todos os recursos disponíveis sem com isso comprometer a performance e fluidez da aplicação. O suporte a dois dispositivos, iPhone e iPad, representou igualmente um desafio do ponto de vista técnico tendo requerido uma procura de mecanismos que permitissem o suporte de ambos através de uma reutilização de componentes diminuindo o esforço de desenvolvimento e sobretudo o esforço de manutenção de ambos.

A aplicação foi desenhada tendo em mente o desenvolvimento de uma biblioteca sólida que vise suportar múltiplas interfaces para múltiplos clientes, tendo esse desenho procurado atingir um determinado nível de abstração que permita alterações internas assim como a incorporação de novos requisitos com um impacto diminuído nas outras componentes da aplicação, nomeadamente a camada de apresentação. O estudo de performance dos formatos de dados permitiu igualmente ter uma visão dos ganhos que cada formato de dados permite alcançar sendo algo que será tido em conta em futuros desenvolvimentos. A integração de lógica de suporte ao negócio assim como lógica para analisar possíveis anomalias constituí igualmente a preocupação em oferecer um produto completo, de valor acrescido, pretendendo-se ser um produto totalmente diferenciador e inovador neste sector.

A integração num ambiente IPTV por ser um contexto bastante específico representou uma oportunidade profissional única de explorar estes sistemas assim como de obter um conhecimento privilegiado nesta área. A possibilidade de expandir as funcionalidades destes sistemas para o sector móvel apresentou-se igualmente uma experiência única por representar uma pequena contribuição do futuro e expansão destes sistemas.

No decurso do estágio houve ainda oportunidade de trabalhar com um conjunto de componentes como o vídeo e imagem tendo sido adquirida alguma experiência nestas áreas assim como um contato com bibliotecas de referência destes sectores, tendo sido uma experiência extremamente recompensadora do ponto de vista profissional.

Trabalho futuro

O estágio visou o desenvolvimento de uma primeira versão do produto, versão essa alcançada, existindo uma série de novas funcionalidades pretendidas pela empresa assim como os desenvolvimentos para o lançamento da oferta *TV Everywhere* da Vodafone Alemanha assegurando assim a continuidade do desenvolvimento do projeto.

Referências

1. WIT-Software. *WIT-Software Web Site*. 2013; Disponível em: http://www.wit-software.com
2. Apcer. *ISO 9001:2008 - Quality Management Systems.* 2013; Disponível em: http://www.apcer.pt/index.php?option=com_content&view=article&id=96:iso-9001&catid=3&Itemid=10
3. Apcer. *ISO 14001:2004 - Environmental Management Systems.* 2013; Disponível em: http://www.apcer.pt/index.php?option=com_content&view=article&id=117%3Aiso14001&catid=4&Itemid=45
4. Apcer. *NP 4457:2007 - Certificação de Sistemas de Gestão da Investigação, Desenvolvimento e Inovação.* 2013; Disponível em: http://www.apcer.pt/index.php?option=com_content&view=article&id=141%3Anp-4457&catid=10&Itemid=60&lang=pt
5. Deloitte. *Deloitte apresenta ranking das 30 empresas portuguesas com maior crescimento*. 2013; Disponível em: http://www.deloitte.com/view/pt_PT/pt/industrias/technology-media-telecommunications/a7ce1963cc99b310VgnVCM1000003256f70aRCRD.htm
6. Microsoft Corporation. *Microsoft Web Site*. 2013; Disponível em http://www.microsoft.com
7. Microsoft Corporation. *Microsoft Mediaroom*. 2013; Disponível em: http://www.microsoft.com/mediaroom
8. AT&T. *AT&T Web Site*. 2013; Disponível em: http://www.att.com/
9. Deutsche Telekom. *Deutsche Telekom Web Site*. 2013; Disponível em: http://www.telekom.com/startseite
10. Telefonica. Telefonica Web Site. 2013; Disponível em: http://www.telefonica.com/en/home/jsp/home.jsp
11. Telus. *Telus Web Site*. 2013; Disponível em: http://www.telusinternational.com/
12. Gartner. *Worldwide Smartphone Sales to End Users by Operating System in 4Q12*. 2013; Disponível em: http://www.gartner.com/newsroom/id/2335616
13. ATIS. *ATIS IPTV Exploratory Group Report and Recommendation to the TOPS Council* 2005. [citado em 2013-01]; Disponível em: http://www.atis.org/tops/IEG/ATIS_IPTV_EG_RPT_final.pdf
14. MEO. *MEO Web Site*. 2013; Disponível em: http://meo.pt
15. NDS. NDS Web Site. 2013; Disponível em: http://www.nds.com
16. ZON. *ZON Web Site*. 2013; Disponível em: http://www.zon.pt
17. NAGRA KUDELSKI. *Digital TV Web Site*. 2013; Disponível em: http://www.nagra.com/dtv/
18. MEO GO!. *MEO GO! Web Site*. 2013; Disponível em: http://meogo.meo.pt/
19. ZON Remote. *ZON Remote Web Site*. 2013; Disponível em: http://www.zon.pt/tv/iris/novidades/Pages/App-ZON-Remote.aspx
20. Netflix. *Netflix Web Site*. 2013; Disponível em: http://www.netflix.com
21. HULU. *HULU Web Site*. 2013; Disponível em: http://www.hulu.com
22. MobiTV. *MobiTV Web Site*. 2013; Disponível em: http://www.mobitv.com/
23. Apple Developer Program. *Conventions*. 2013; Disponível em: http://developer.apple.com/library/ios/ -documentation/cocoa/conceptual/ProgrammingWithObjectiveC/Conventions/Conventions.html

24. Apple. Apple Developer Program. 2013; Disponível em: https://developer.apple.com/devcenter/ios/checklist/
25. Apple Developer Program. *NSURLConnection Class Reference*. 2013; Disponível em: http://developer.apple.com/library/mac/ - documentation/Cocoa/Reference/Foundation/Classes/NSURLConnection_Class/Reference/Reference.html
26. Apple Developer Program. *iOS Human Interface Guidelines*. 2013; Disponível em: http://developer.apple.com/library/iOS/ - documentation/UserExperience/Conceptual/MobileHIG/UIElementGuidelines/UIElementGuidelines.html
27. ZUUIRevealController. *Repositório do código fonte*. 2013; Disponível em: https://github.com/pkluz/ZUUIRevealController
28. PSTCollectionView. *Repositório do código fonte*. 2013; Disponível em: https://github.com/steipete/PSTCollectionView
29. FFmpeg. *FFmpeg Web Site*. 2013; Disponível em: http://www.ffmpeg.org/
30. Discretix. *Discretix Web Site*. 2013; Disponível em: http://www.discretix.com/
31. Google Analytics. *Google Analytics Web Site*. 2013; Disponível em: http://www.google.com/analytics/
32. Crashlytics. *Crashlytics Web Site*. 2013; Disponível em: http://try.crashlytics.com/
33. GraphicsMagick. *GraphicsMagick Web Site*. 2013; Disponível em: http://www.imagemagick.org/
34. ImageMagick. *ImageMagick Web Site*. 2013; Disponível em: http://www.graphicsmagick.org/
35. OpenMP. *OpenMP Web Site*. 2013; Disponível em: http://openmp.org/
36. Jenkins. *Site oficial*. 2013. Disponível em: http://jenkins-ci.org
37. Hudson. *Site Oficial*. 2013. Disponível em: http://hudson-ci.org
38. TV Connect. *TV Connect Event Web Site*. 2013; Disponível em: http://www.tvconnectevent.com/