FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Media Asset Tracking Tool

Preparação de Dissertação - Relatório Final

Tomás Tavares

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Orientador: Prof. Maria Teresa Andrade

Orientador Externo: Ricardo Serra

1 de Fevereiro de 2017

Conteúdo

Capítulo 1

Introdução

A evolução tecnológica sentida nos últimos anos, revolucionou por completo o panorama das tecnologias de informação e por consequência do quotidiano da maioria das pessoas. A informação é produzida, distribuída e consumida a velocidades nunca antes vistas, podendo essa informação ser acedida a qualquer hora e em qualquer local.

Por um lado, a grande proliferação da internet foi em parte responsável por este fenómeno, pois sustentou o aparecimento de novos canais de comunicação, nomeadamente das redes sociais. Por outro lado, o aparecimento de novos equipamentos, cada vez mais acessíveis para o consumidor, como *smartphones* ou *smart TVs*, tiveram também um papel de relevância nesta evolução, pois permitiram que qualquer indivíduo se tornasse capaz de produzir e difundir informação.

Na produção e consumo de televisão existiu também uma revolução tecnológica. Em primeiro lugar deu-se o abandono do sinal analógico em prol do sinal digital, deixando de se utilizar os sistemas baseados em cassetes magnéticas, mas sim em ficheiros digitais. Para além disso, registou-se também uma evolução ao nível das redes de distribuição, que se tornaram cada vez mais eficientes, com maiores velocidades de transmissão e largura de banda, assim como ao nível dos equipamentos, que se tornaram mais desenvolvidos e acessíveis, quer ao nível da captura, da distribuição ou do consumo de televisão. Este factor aliado a progressos nas áreas de codificação e tratamento de sinal, veio permitir o aparecimento de sinais e formatos de maior qualidade e resolução. Por último, o fenómeno da internet previamente descrito influenciou também a forma como a televisão é consumida. Surgem assim novas plataformas de consumo de televisão e de modelos de interacção com o utilizador, através de, por exemplo, as redes sociais, com a introdução da *social TV*.

Todas estas mudanças de paradigma alteraram os hábitos diários de grande parte dos consumidores, que hoje exigem sistemas cada vez mais eficientes e complexos, que sejam capazes de acompanhar o desenvolvimento tecnológico e permitir uma experiência capaz de cativar o utilizador.

1.1 Enquadramento

Esta nova realidade no panorama de produção de televisão veio introduzir novos desafios tecnológicos e de engenharia. É necessário encontrar formas de gerir, tornar eficientes e desenvolver os novos sistemas de produção de televisão. Não efectuar uma gestão eficiente dos recursos de um sistema de produção televisiva, pode significar perdas significativas do ponto de vista de negócio

Nos dias de hoje, as tecnologias para desenvolvimento de *software* estão também cada vez mais evoluídas e capazes de suportar soluções mais complexas. Se a essas tecnologias forem aliadas metodologias de desenvolvimento de *software* eficientes, poder-se-ão produzir ferramentas valiosas e poderosas, que prestem auxílio em ambientes de produção, trazendo resultados práticos ao nível da eficiência do sistema, mas também do ponto de vista de negócio.

Neste contexto, o presente projecto é proposto pela *MediaGaps*, *start up* especializada no desenvolvimento de *software* para a indústria televisiva e de entretenimento. É um projecto de investigação e desenvolvimento integrado no plano estratégico da empresa, que procura desenvolver uma ferramenta inovadora, baseada nas tecnologias e metodologias mais actuais.

1.2 Motivação e Objectivos

Hoje em dia, é ainda comum que não haja mecanismos capazes de mapear de forma eficiente a distribuição de *media assets* num sistema de produção de conteúdos televisivos, resultando em sistemas ineficientes, com desaproveitamento de recursos e perdas de tempo e dinheiro. Desta forma, o objectivo final desta dissertação é desenvolver uma ferramenta capaz de auxiliar na gestão e optimização de sistemas nestes ambientes, fazendo a monitorização de *media assets* numa rede de computadores e fornecendo informação para que o sistema possa ser melhorado.

Para tal, é necessário realizar uma larga pesquisa que deverá incidir não só sobre o ambiente de produção, a sua arquitectura e o tipo de formatos e ficheiros utilizados, mas também sobre toda as tecnologias que sustentarão o desenvolvimento da aplicação, assim como metodologias para o desenvolvimento.

Após a primeira fase de pesquisa e investigação, terá lugar a fase de levantamento de requisitos, desenho da arquitectura de sistema e desenvolvimento e teste de um protótipo funcional da ferramenta a implementar.

Capítulo 2

Revisão Bibliográfica

Neste capítulo são apresentados os resultados de uma pesquisa acerca dos diferentes temas, tecnologias e metodologias que suportam o projecto, fazendo uma descrição do panorama actual para cada um deles.

2.1 Introdução

Se por um lado, o panorama da televisão mudou, fazendo com que o utilizador tenha novas formas de consumo em diferentes plataformas e locais, por outro, as tecnologias de desenvolvimento web estão também cada vez mais poderosas, permitindo aplicações cada vez mais complexas e experiências cada vez melhores para o utilizador. Aliadas a estes avanços tecnológicos estão também novas metodologias de desenvolvimento de *software*, que tornam o trabalho de um *devoloper* cada vez mais metódico, trazendo também vantagens práticas nas aplicações desenvolvidas.

2.2 Produção de conteúdo televisivo em ambiente profissional

Durante alguns anos, a cadeia de valor e de produção de conteúdos televisivos manteve-se, de forma geral, inalterada. No entanto, nos dias que correm a informação é gerada e consumida a velocidades cada vez maiores, que chegam aos consumidores através de variadas plataformas e meios. Para acompanhar esta evolução, a televisão teve também de se adaptar e procurar evoluir no sentido de preencher os requisitos impostos pelos consumidores.

Esta evolução fez surgir novas formas de consumir conteúdos televisões. A televisão deixou de ser um simples electrodoméstico presente na casa de milhões de lares, e passou a ser possível ver televisão em diferentes ecrãs, em qualquer sítio e a qualquer hora. O utilizador passou a ter controlo sobre aquilo que quer ver, a qualquer hora, com o aparecimento de soluções *on demand* (televisão não linear), e a imersividade e interactividade tornaram-se cada vez mais presentes.

Apesar de toda esta evolução na produção de televisão, é possível definir uma cadeia de valor assente em 4 fases [1]:

- Produção de conteúdos;

- Programação;
- Distribuição;
- Consumo.

De uma forma resumida, os conteúdos são produzidos e de seguida organizados numa programação. Nesta fase são organizados sequencialmente e intercalados com outros conteúdos, nomeadamente publicitários. O passo seguinte é distribuí-los pelas diferentes plataformas, fazendo-os chegar até aos consumidores.

2.2.1 Workflows de Produção de Televisão

A produção de conteúdos televisivos pode ser subdivida em 3 fases. A fase de pré-produção consiste na conceptualização e idealização do conteúdo a ser produzido. A fase produção consiste na captura dos sinais vídeo e áudio propriamente ditos. Posteriormente dá-se a fase de pós-produção na qual os conteúdos são processados e editados.

No passado, a tecnologia mais utilizada na produção e gravação de conteúdos televisivos eram as cassetes[2]. Nestes sistemas, conhecidos por *tape-based*, o sinal analógico de vídeo e áudio era adquirido e gravado em fitas magnéticas. Todo o processo posterior à aquisição e gravação do sinal era extremamente trabalhoso e dependente de intervenção humana. Toda a catalogação, registo de informação e arquivo tinha de ser feito manualmente, assim como a transição entre as diferentes fases do *workflow* de produção.

Na pós-produção, a edição dos conteúdos era também mais difícil, uma vez que a fita se comporta de forma sequencial e linear, e portanto os mecanismos para percorrer e juntar diferentes fitas eram bastante mais complexos do que nos sistemas utilizados nos dias de hoje.

Para além destas características, pelo facto do material de arquivo ser físico e analógico, a qualidade dos conteúdos tende a degradar-se com o tempo. Era também muito comum existirem dificuldades de compatibilidade entre diferentes formatos e equipamentos necessários para processar os conteúdos.

Com a era digital, novos sistemas de produção surgiram, utilizando como base ficheiros digitais em alternativa às cassetes. Por esta razão, são conhecidos como sistemas *file-based*.

À semelhança dos sistemas *tape-based*, no paradigma digital, o *workflow* tem como ponto de partida a captura e aquisição de um sinal vídeo e áudio através de equipamentos (*hardware*) como câmeras ou microfones. Tipicamente não é aplicada compressão ao vídeo e ao áudio nesta etapa. Após a fase de captura, realiza-se o *ingest*. Nesta fase, os conteúdos multimédia (vídeo e áudio) são encapsulados juntamente com metadados, que é informação sobre os conteúdos em causa. Estes dados serão muito úteis nas fases seguintes, uma vez que contêm informação sobre sincronização, legendas, descrições dos ficheiros de áudio e vídeo, entre outros. Todo este processo tem o nome de *wrapping*. O objectivo desta fase é agrupar os diferentes ficheiros num único ficheiro.

Existem inúmeras possibilidades de parâmetros e configurações para os diferentes ficheiros a serem encapsulados. Estas possibilidades variam tendo em conta os formatos de vídeo e áudio,

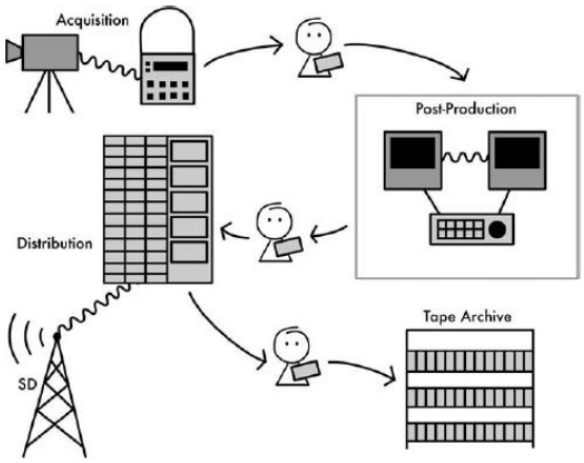

Figura 2.1: Workflow de um Sistema *tape based* [3]

a aplicação a que os conteúdos se destinam, os equipamentos e *software* utilizados e até mesmo as preferências pessoais dos realizadores e produtores. Todas estas variáveis fazem com que seja difícil existir um padrão conhecido e instituído à saída da fase de *ingest*. No entanto, o formato mais utilizado no *wrapping* é o MXF, que foi criado precisamente com o objectivo de uniformizar os conteúdos, por um lado suportando diferentes configurações mas por outro garantindo interoperabilidade entre os diferentes intervenientes na produção.

Por fim, os conteúdos são enviados para uma fase de pós produção, onde serão editados, tratados e processados, antes de serem enviados para transmissão ou arquivo. Uma vez que estamos a falar de formatos digitais, todas estas operações são feitas recorrendo a *software*, o que as torna consideravelmente mais simples, do que no caso de formatos analógicos.

Neste paradigma digital, a intervenção humana é menor do que nos sistemas *tape-based*. Os ficheiros são processados em poderosos *softwares* que tornam as actividades de catalogação, edição, transporte e armazenamento muito mais simples, podendo até chegar a ser automatizados.

2.2.2 MXF Material eXchange Format

O MXF surge nos anos 90, quando o rumo da televisão começou cada vez mais a passar pelas novas tecnologias de informação [2]. As cassetes caíam em desuso e no seu lugar surgiam ficheiros digitais, servidores de vídeo e *softwares* de compressão e edição de vídeo. Com todas estas novidades tecnológicas, surgiram diferentes formatos e terminologias, que por sua vez criavam dificuldades de interoperabilidade entre diferentes fabricantes e tecnologias. Deste cenário surgiu um esforço conjunto entre a *European Broadcasting Union (EBU)* e a *Society of Motion Picture Television Engineers (SMPTE)*, para a criação de um formato padrão e aberto. Em primeiro lugar foram identificados os pré-requisitos que o novo formato deveria preencher. Após alguns primeiros esboços, em 2004 foi lançada a primeira versão do Material eXchange Format (MXF), com o nome SMPTE S377M.

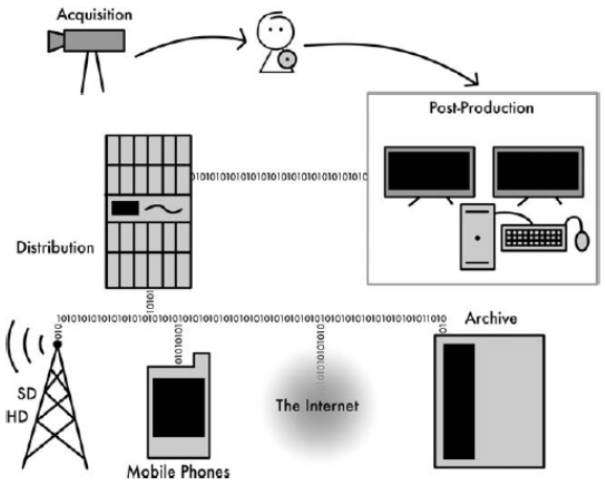

Figura 2.2: Workflow de um Sistema *file base* [3]

O MXF é um *wrapper* que encapsula conteúdo audiovisual e dados associados a esse conteúdo [4]. Desta forma, não actua como um codificador, e transporta conteúdo em bruto ou codificado em qualquer formato, sem alterar a qualidade do conteúdo que está a encapsular.Diz-se por isso *format-agnostic*. O MXF pode ser utilizado em diferentes aplicações, desde armazenamento a edição em tempo real, permitindo uma grande flexibilidade, uma vez que permite adaptações para optimizar a aplicação em causa, sem perder eficiência.

Os pacotes de um ficheiro MXF, são identificados por um UMID e podem ser de dois tipos:

- File packages - representam os ficheiros de origem;
- Material packages - representam a *output timeline* desejada.

Como descrito na figura, os ficheiros MXF dividem-se no mínimo em duas partes: *header* e *footer*. O *header* contem a *Structure Metadata*, que transporta informação importante como a *output timeline*, características técnicas, identificação dos pacotes e informação descritiva dos conteúdos. O *footer* indica o final do ficheiro. Pode ainda existir uma terceira parte, localizada entre o *header* e o *footer*, com o nome de *File Body*, que contém a chamada essência do ficheiro.

Analisando a um mais baixo nível, o MXF estrutura-se num trio *KLV (Key - Length - Value)*. A *key* funciona como um identificador de 16 *bytes*, a *length* indica o tamanho do elemento e o value é a informação propriamente dita (vídeo, áudio ou metadados).

A essência é transportada dentro dos *Essence Containers*, que estruturam e organizam a informação, que pode ser uma imagem, áudio ou metadados. Estes *containers* podem ser de três tipos:

- Frame-Wrapped - a frame e o audio correspondente são encapsulados dentro do mesmo KLV;

Figura 2.3: Estrutura de um ficheiro MXF

- Clip-Wrapped - todos os samples são encapsulados dentro do mesmo KLV;
- Custom-Wrapped - o encapsulamento é costumizável de acordo com a aplicação.

Por último, a *Index Table* faz também parte do ficheiro MXF e faz o mapeamento entre uma *frame* um *offset byte*. Permite assim que haja acesso aleatório.

Por todas estas características, o MXF tornou-se o *standard* mais utilizado em ambientes de produção de televisão, quer pelos fabricantes de *hardware*, quer pelos *end-users*.

2.2.3 Media Asset Management

Media Asset Management pode ser definido como o conjunto de acções tomadas com a finalidade de gerir, organizar, catalogar, armazenar ou distribuir *Media Assets*. Estas acções são feitas recorrendo a sistemas de *hardware* e *software* que suportam todo o processo. A *metadata* associada à essência dos assets tem um papel importante e facilitador na gestão de *Media Assets*, uma vez que fornece informação aos intervenientes no processo. Com a evolução das tecnologias web, surgiram soluções de gestão de *Media Asset* baseadas na *cloud*.

É de elevada importância para as empresas de produção e *broadcast* de conteúdos multimédia, terem ferramentas eficientes de gestão dos seus *assets*. A quantidade de conteúdos produzidos é hoje maior do que nunca, devido às evoluções tecnológicas e às novas plataformas de consumo multimédia. Este tipo de ferramentas permite às empresas uma maior organização e rapidez nas suas operações, que resultam numa maior produtividade e eficiência.

2.3 Tecnologias de Desenvolvimento de *software*

2.3.1 Arquitectura Orientada a Serviços - *Microservices*

Num passado recente as aplicações eram maioritariamente desenvolvidas seguindo uma arquitectura monolítica. Este tipo de arquitectura consistia em longos excertos de código, responsáveis pelos requisitos funcionais e não funcionais da aplicação [5]. Embora este conceito funcionasse para aplicações de complexidade reduzida, quando esta aumentava e com as novas tendências de migração para a *cloud* surgiram novas abordagens.

Uma das abordagens que se tornou mais popular são os *microservices*. Esta é uma arquitectura modular, que tem como princípio a desconstrução de uma aplicação de elevada complexidade em tarefas básicas e primitivas, desempenhadas por *microservices*. Estes *microservices* devem ser processos simples, independentes e que comunicam entre si através de uma API agnóstica à linguagem utilizada em cada um deles [6]. Devem ainda ser independentes do local onde estão alojados e do instante no qual são requisitados, não exigindo que a aplicação se encontre em determinado estado, funcionando de forma assíncrona.

Esta abordagem torna o desenvolvimento de aplicações mais eficiente, pois facilita o processo de alteração, *update* e teste, permitindo ainda a reutilização de *microservices* entre diferentes aplicações, dentro e fora da mesma empresa, e até a comercialização destes serviços.

2.3.2 Bases de Dados Relacionais e Não Relacionais

O desenvolvimento da Internet, reflectido no aumento de utilizadores e da complexidade das aplicações fez com que, nos dias de hoje, a quantidade de informação que uma base de dados tem de tratar, seja cada vez maior. O desempenho da aplicação depende da velocidade e eficiência com que a base de dados funcionar, e portanto é de extrema importância que haja um desenho e dimensionamento da base de dados e do sistema a utilizar.

No passado, as bases de dados eram tipicamente relacionais. Este tipo de bases de dados são baseadas no Modelo Relacional e a informação é mantida em diferentes tabelas, relacionadas ou não entre si, alojadas tipicamente no mesmo servidor.

Para se desempenharem diferentes operações, como por exemplo inserção ou remoção de informações na base de dados, é utilizada uma linguagem intitulada de SQL. Cada query SQL representa uma transacção. As bases de dados relacionais têm como base o princípio ACID, descrito pelas seguintes propriedades[7]:

- Atomicity - uma transacção é bem sucedida na totalidade ou então não é realizada;
- Consistency - as transacções alteram o estado da base de dados, para um novo estado consistente, sem que haja perda da consistência da informação;
- Isolation - transacções concorrentes não se podem influenciar entre si, sendo que o resultado esperado deve ser o mesmo do que executando as duas transacções sequencialmente;
- Durability - os resultados de uma transacção devem ser guardados e não devem ser perdidos por uma falha de sistema

Nas bases de dados relacionais, o problema de escalabilidade é resolvido de uma forma vertical, ou seja, o sistema é escalado recorrendo a *hardware* de maior capacidade (novos servidores). Esta abordagem é bastante limitada, não só porque é extremamente custosa do ponto de vista financeiro, mas também porque atinge um ponto de estagnação em que não é possível escalar mais o sistema e os tempos de resposta da base de dados se tornam tão elevados que a aplicação perde a usabilidade.

Com a avalanche de informação causada pelo *boom* das redes sociais, começou a ser necessário pensar em alternativas para as base de dados relacionais, que fossem capazes de lidar com toda esta informação de uma forma mais eficiente. Em 2009, iniciou-se um movimento para criar um novo paradigma de base de dados, no qual a informação deixasse de estar centralizada num só servidor e passasse a estar distribuída pela internet. Surgem assim as base de dados não relacionais, também conhecidas como NoSQL. Este novo paradigma vem romper com os princípios estabelecidos pelas bases de dados relacionais. As bases de dados NoSQL deixam de assentar no ACID. Em vez disso, baseiam-se no teorema CAP criado por E. Brewer.

Este teorema define que, numa base de dados não relacional apenas é possível ter duas das seguintes propriedades:

- Consistency - todas as operações de leitura acedem à mesma informação;
- Availability - qualquer pedido à base de dados é respondido, independentemente da circunstância;
- Partitional tolerance - o sistema funciona independentemente de uma partição arbitrária, causada por problemas físicos na rede.

Este teorema fundamenta o modelo de transacções BASE, que pode ser como uma oposição directa ao modelo ACID:

- Basically Available - os dados estão disponíveis de acordo com o teorema de CAP, no entanto todos os pedidos têm uma resposta, mesmo que esta possa ser uma mensagem de erro;
- Soft Stage - indica que o estado do sistema pode mudar, mesmo que não exista uma transacção de input. Isto pode acontecer devido a operações de background para garantir consistência;
- Eventual Consistency - a consistência não é verificada após cada operação, continuando a responder pedidos.

Nas bases de dados NoSQL, a informação deixa de estar contida em tabelas, e passa a ser contida em gráficos, documentos ou arrays associativos, distribuídos pela internet[8]. Na verdade pode ser estabelecido um paralelismo entre as linhas de uma tabela numa base de dados relacional, que podem ser vistas como documentos numa base de dados não relacional, e as tabelas podem ser vistas como colecções de documentos.

Estas características fazem com que as bases de dados não relacionais sejam facilmente escaláveis e replicáveis, numa abordagem horizontal, ao contrário das bases de dados relacionais. O custo e o tempo de acesso é também muito mais reduzido, o que permite que a performance das aplicações seja também melhorada.

2.3.2.1 MongoDB

As bases de dados não relacionais são cada vez mais usadas, o que fez com que diferentes tecnologias, com diferentes abordagens surgissem adoptando este novo paradigma. No entanto, de todas estas tecnologias, aquela que mais vingou foi o Mongo DB [8].

Esta solução surgiu em 2007, criada por uma empresa chamada 10gen e foi apresentada como uma base de dados capaz de escalar de acordo com as necessidades da aplicação. Este serviço nunca chegou a ser totalmente aceite pelos *developers*, o que fez com que em 2009 a empresa decidisse disponibilizar o seu sistema como *open-source*. Este sistema tinha como por base uma base de dados MySQL, que foi alterada de forma a tornar-se numa base de dados distribuída e orientada a documentos. Para tal, recorre a BSON (JSON binário) para estruturar os seus documentos.

Algumas das características do MongoDB é o suporte de *updates* parciais e totais de documentos, ter uma linguagem de pesquisa rica e flexível, ser escalável e rápida de aceder e ser relativamente fácil de utilizar. Existem ainda bibliotecas não oficiais que facilitam a ligação entre a base de dados e diferentes linguagens como PHP, Java, C, Python entre outras.

Todas estas razões fazem com que esta seja a tecnologia mais utilizada nos dias de hoje no que diz respeito a bases de dados não relacionais.

2.3.3 HTML e CSS

O HyperText Markup Language (HTML) é a linguagem markup utilizada para a criação de *webpages*, sendo reponsável por estruturar o conteúdo de uma página. A história do HTML está intimamente ligada com a história da *World Wide Web*, uma vez que têm o mesmo criador *Tim Berners-Lee* [9]. A primeira versão da linguagem foi proposta em 1992 no CERN e veio a evoluir com contribuição de diferentes entidades como a *Internet Engineering Task Force* ou o *World Wide Web Consortium* (W3C)[10]. Os *browsers* após fazerem um pedido a um servidor para aceder a uma determinada página, recebem um ficheiro HTML como resposta e transformam-no em informação visual para mostrar ao utilizador.

A estrutura da página é definida de uma forma semântica, e os elementos base são definidos recorrendo a *tags* descritas pelos símbolos <> [2]. Existem variadas *tags*, que representam diferentes elementos de uma página. Estes elementos podem ser desde formulários interactivos, textos, imagens ou outros elementos multimédia. Para além disso, o HTML é ainda responsável pelas hiperligações entre as diferentes páginas.

O HTML5 é a última versão deste *standard* e veio introduzir novos elementos, transições e comportamentos, tornando as aplicações web cada vez mais poderosas [11].

Em boa prática, o HTML deve ser utilizado apenas para estruturar o conteúdo de uma página e não para definir as características visuais e de formatação. Para tal função, foi criada uma linguagem chamada de Cascading Style Sheets (CSS). Ambos os *standards* são definidos pelo W3C, e devem ser utilizados em conjunto.

Separar o conteúdo da apresentação de uma página, recorrendo ao CSS, é uma solução inteligente e eficiente, pois permite definir estilos para múltiplas páginas em simultâneo, definindo

propriedades como cores, fontes ou layouts. Para o fazer, o CSS recorre a dois conceitos fundamentais: os selectores e as propriedades. Através dos selectores, o CSS selecciona qual o elemento da página HTML sobre o qual vai interferir[12]. Após seleccionado, um conjunto de propriedades pode ser definido acerca desse elemento.

A primeira versão do CSS surgiu em 1996, e evoluiu gradualmente até à actual terceira versão, garantindo suporte em todos os *browsers*, nos dias de hoje[13].

2.3.4 JavaScript

Em paralelo com o HTML e CSS, o JavaScript é uma linguagem core do desenvolvimento web. Surgiu em 1995, desenvolvida por Brendan Eich na Netscape. Em 1996, foi *standardizado* pela ECMA International, baseado no ECMA Script. Hoje em dia é suportado por todos os *browsers* modernos, sendo uma marca registada da Oracle Corporation, e gerido pela Mozilla Foundation[14].

É uma linguagem orientada a objectos, prototype-based, dinâmica, imperativa e funcional. Pode ser utilizada do lado do cliente, no *browser*, para, por exemplo, manipulação de dados, mas também do lado do servidor, para aceder a bases de dados[15].

Embora seja maioritariamente utilizada em ambientes web, pode ter também aplicações noutro tipo de ambientes, desde o PDF a desenvolvimento de videojogos.

2.3.4.1 NodeJS

O NodeJS é um runtime environment de JavaScript, desenvolvida no V8 Javascript Engine da Google, que surgiu em 2009 e que foi criado por Ryan Dahl. É uma linguagem assíncrona e event-driven, o que a torna muito utilizada na criação de webservers, uma vez que resolve problemas de concorrência sem recorrer a *threads*[16].Como não recorre a *threads*, não existem problemas de *dead locking*, e por isso torna possível o desenvolvimento de sistemas escaláveis e capazes de lidar com aplicações em tempo real, como sistemas de comunicação ou jogos *online*.

2.3.4.2 ReactJS

O ReactJS é uma biblioteca de Javascript, criada em 2013 pelo Facebook, para desenvolver *user interfaces*. Surge da necessidade de criar interfaces gráficos para aplicações de elevada complexidade e com informação variável [17]. O ReactJS é declarativo, isto é, permite que sejam projectadas *views* para cada estado da aplicação, que serão actualizadas e renderizadas automaticamente aquando de uma alteração na informação.

É também component-based, ou seja, os componentes podem ser desenvolvidos de forma independente, sendo capazes de gerir o seu próprio estado. É ainda possível fazer a renderização das *views* do lado do servidor, combinando a utilização de ReactJS com NodeJS.

Os componentes são tipicamente escritos em JSX, que é uma extensão de sintaxe de JavaScript, que permite a utilização de sintaxe de HTML com JavaScript.

Capítulo 3

Caracterização do Problema

Neste capítulo será apresentado o problema que este projecto se propõe a resolver, assim como uma breve descrição da solução encontrada e das tecnologias que a sustentam.

3.1 Definição do Problema

Com a evolução do panorama televisivo nos últimos tempos, os produtores de televisão têm nos dias de hoje sistemas cada vez mais complexos. Estes sistemas englobam diversos estúdios de televisão, muitas vezes separados geograficamente, que partilham recursos como servidores, que por sua vez contêm milhares de horas de gravações. Frequentemente estes *media assets* são adicionados e removidos dos servidores num ambiente em que não é possível fazê-lo com a precaução e o critério desejado, como em ambientes de emissão em tempo real. Tal prática pode originar repetições desnecessárias de *media assets*. Esta replicação pode ser extremamente dispendiosa para as companhias produtoras de conteúdos, uma vez que a memória se torna um recurso cada vez mais caro nos dias de hoje, e deve portanto ser evitada.

Toda esta complexidade exige uma gestão eficiente destes recursos e dos *media assets* neles contidos. É por isso importante ter ferramentas que auxiliem nesta gestão, para que os recursos não sejam usados de forma ineficiente, o que pode representar melhorias significativas no sistema, que por sua vez também serão reflectidas do ponto de vista de negócio.

3.2 Solução Proposta

Este projecto de dissertação propõe-se a criar uma solução de *Media Asset Management*, capaz de auxiliar e minimizar o problema previamente descrito. Esta solução terá o nome de *MATT - Media Asset Tracking Tool*, e fará a análise de uma rede de computadores e dos *Media Assets* neles presentes. Após obter estas informações, será capaz de reportar ao utilizador quais os ficheiros em cada servidor e informação adicional sobre os ficheiros, permitindo assim fazer um mapeamento do estado actual do sistema e potencialmente torna-lo mais eficiente.

A informação será obtida através de uma aplicação desenvolvida em NodeJS, que correrá nos diferentes servidores do sistema, e que comunicará com diferentes serviços responsáveis por extrair a informação da rede e dos ficheiros MXF.

Toda a informação será carregada para uma base de dados não relacional, o que permitirá consultar o estado actual e passado do sistema a qualquer momento.

Por último, uma aplicação *web* será responsável por apresentar toda a informação ao utilizador, consultando a base de dados onde a informação se encontra centralizada. A aplicação web terá ferramentas de organização, visualização e pesquisa de informação, que tornarão a experiência do utilizador mais simples e eficiente.

Capítulo 4

Plano de Trabalho

Neste capítulo será apresentada uma planificação preliminar do trabalho a desenvolver ao longo do projecto. Serão ainda apresentadas as tecnologias e ferramentas a utilizar.

4.1 Planeamento e Metodologia

Uma vez que o modelo escolhido para implementar e gerir este projecto é um modelo ágil, o plano de trabalho não segue uma sequência temporal completamente definida. Em vez disso, será implementada uma cadência de trabalho iterativa com feedback constante, fazendo com que os requisitos possam variar ao longo do desenvolvimento.

Este tipo de metodologia, induz a uma implementação vertical, em oposição à mais clássica implementação horizontal. Desta forma, procurar-se-á desenvolver as diferentes funcionalidades individualmente, passando pelas diferentes camadas, e não construir uma camada horizontal do sistema e só depois passar à construção da camada seguinte.

4.1.1 Identificação de Tarefas

Embora não seja possível definir uma sequência temporal para todas as tarefas, é possível identificar tarefas que serão a base para o desenvolvimento do projecto, fazendo uma análise de alto nível.

Estas serão tarefas realizadas ao longo das cerca de 20 semanas de desenvolvimento do projecto. Para as primeiras semanas, podem prever-se tarefas relacionadas com levantamento de requisitos e dimensionamento e desenho da arquitectura do sistema, em oposição às últimas semanas, nas quais se espera realizar algumas tarefas de documentação. As restantes tarefas dividir-se-ão ao longo das semanas, podendo ser desempenhadas de uma forma iterativa, de acordo com o modelo ágil.

- Levantamento de requisitos e desenho da arquitectura do sistema;
- Dimensionamento da Base de Dados;

- Desenho de *mock ups* para a Interface Gráfica da Aplicação Web;
- Desenvolvimento de aplicação NodeJS responsável por agregar diferentes serviços e comunicar com a base de dados;
- Desenvolvimento dos diferentes serviços responsáveis por extrair informação dos ficheiros e do sistema;
- Desenvolvimento da Base de Dados;
- Implementação do Interface Gráfica da Aplicação Web;
- Desenvolvimento de documentação e manuais de utilização da ferramenta;

4.2 Tecnologias e Ferramentas

Uma vez que o trabalho consiste numa ferramenta de *media asset management*, o *core* da solução será implementado em *software*, pelo que não serão necessários equipamentos de *hardware* complexos, à excepção de servidores para testar a aplicação.

Em termos de tecnologias, as licensas de *software* necessárias serão fornecidas pela empresa *MediaGaps*, nomeadamente licensas para acesso a ferramentas da Microsoft Developer Network, em concreto para o Visual Studio. Os restantes *softwares* a utilizar são de uso gratuito, não sendo necessárias licensas.

Está ainda previsto implementar um sistema de testes da ferramenta, que poderá eventualmente recorrer a sistemas reais da empresa *MediaGaps*.

Serão utilizados sistemas de controlo de versões e de gestão de projecto.

Referências

[1] Pedro Magalhães. Produção de Televisão UHD para Eventos em Direto. Tese de mestrado, Universidade do Porto, Julho 2016.

[2] Ricardo Serra. Interfaces tácteis baseadas em HTML5/CSS3/JavaScript. Tese de mestrado, Universidade do Porto, Julho 2001.

[3] Bruce Devlin, Jim Wilkinson, Matt Beard, e Phil Tudor. *The MXF Book: An Introduction to the Material eXchange Format*. Focal Press, Amsterdam; Boston, 1 edition edição, Março 2006.

[4] Pedro Ferreira. MXF- a progress report. 2010. URL: `https://tech.ebu.ch/docs/techreview/trev_2010-Q3_MXF-1.pdf`.

[5] M. Yousif. Microservices. *IEEE Cloud Computing*, 3(5):4–5, Setembro 2016. doi:10.1109/MCC.2016.101.

[6] D. S. Linthicum. Practical Use of Microservices in Moving Workloads to the Cloud. *IEEE Cloud Computing*, 3(5):6–9, Setembro 2016. doi:10.1109/MCC.2016.114.

[7] L. Vokorokos, M. Uchnár, e L. Leščišin. Performance optimization of applications based on non-relational databases. Em *2016 International Conference on Emerging eLearning Technologies and Applications (ICETA)*, páginas 371–376, Novembro 2016. doi:10.1109/ICETA.2016.7802079.

[8] Carlos Soares. NoSQL (NoRDBMS) - Slides da UC Sistemas de Informação e Bases de Dados, 2016.

[9] André Restivo. HTML5, 2017. URL: `https://web.fe.up.pt/~arestivo/presentation/html5/#1`.

[10] W3C. HTML & CSS - W3c, 2017. URL: `https://www.w3.org/standards/webdesign/htmlcss`.

[11] MDN. HTML5 - Web Developer guides | MDN, 2017. URL: `https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5`.

[12] André Restivo. CSS 3, 2017. URL: `https://web.fe.up.pt/~arestivo/presentation/css3/#1`.

[13] MDN. CSS | MDN, 2017. URL: `https://developer.mozilla.org/en-US/docs/Web/CSS`.

[14] MDN. JavaScript | MDN, 2017. URL: `https://developer.mozilla.org/en-US/docs/Web/JavaScript`.

[15] André Restivo. Javascript, 2017. URL: https://web.fe.up.pt/~arestivo/presentation/javascript/#1.

[16] Node.JS. Node.js, 2017. URL: https://nodejs.org/en/about/.

[17] React. A JavaScript library for building user interfaces - React, 2017. URL: https://facebook.github.io/react/.