Faculdade de Engenharia da Universidade do Porto

Portal de Visualização de Conteúdos Multimédia de Acesso Condicionado

José Carlos Marques Rodrigues

VERSÃO Final

Relatório de Projecto realizado no Âmbito do
Mestrado Integrado em Engenharia Informática e Computação

Orientador: Prof. Doutor Eurico Manuel Carrapatoso

Julho de 2008

Título de Relatório de Projecto

Nome do Autor

Relatório de Projecto realizado no Âmbito do
Mestrado Integrado em Engenharia Informática e Computação

Aprovado em Provas Públicas pelo Júri:
Presidente: Prof. Doutor António Fernando Coelho

Arguente: Prof. Doutor Nuno Magalhães Ribeiro
Vogal: Prof. Doutor Eurico Manuel Carrapatoso

17 de Julho 2008

Resumo

Este documento apresenta o trabalho de investigação e desenvolvimento, do projecto desenvolvido na empresa MOG Technologies no âmbito do estágio curricular do Mestrado Integrado em Engenharia Informática e Computação. Este trabalho foi desenvolvido dentro dos prazos limites impostos e com sucesso.

O principal objectivo do projecto foi o desenvolvimento de um player de vídeo capaz de obedecer a determinados requisitos específicos, capaz de ser usado num ambiente web. Para tal foram efectuados diversos estudos sobre diversas tecnologias capazes de tocar vídeo e analisadas se seriam as indicadas para a resolução do problema apresentado no trabalho a ser desenvolvido. É também apresentado o contexto geral em que o projecto se insere e a arquitectura, e em seguida os detalhes de implementação.

O player deve ser também capaz de comunicar com a plataforma em que se insere, por forma a que a informação a ser introduzida na base de dados através do uso do player não se perca de alguma forma.

Todo o trabalho desenvolvido resultou num player de vídeo integrado numa página web com controlos ActiveX, todo este trabalho foi baseado num player de vídeo já existente na empresa, o MOG Player, que era uma aplicação *desktop* . O trabalho foi realizado com sucesso.

Abstract

This document describes the investigation and development, of the solution developed in the company MOG Technologies on the project curriculum of the Master's in Integrated Information Technology and Computer Engineering. This project was completed within the deadlines established for his completion.

The main goal of the project was the development of a video player capable of obey to a pre established specific requisites, to be used in a web environment. For such objective some studies were made about diverse technologies that are able to play video, and were analyzed if they were the indicated for the resolution of the problem presented by the project. The general context where the project is inserted is shown too, like as the architecture and followed by the implementation details.

The player must be able to communicate with the platform where is integrated, so the information obtained by the use of the player can be stored in the data base.

All the tasks done resulted in a video player integrated in a web page with ActiveX controls, of an existing video player in the company. All the work was made with success.

Agradecimentos

A realização deste projecto não seria possível sem o apoio dado por muitas pessoas que aqui gostaria de agradecer:

À minha namorada Antonieta Olivera por todo o apoio que me deu durante todo este período.

Aos meus pais e irmã pelo ajuda incansável que me prestaram.

Ao professor Eurico Manuel Carrapatoso da Faculdade de Engenharia pela ajuda neste documento e pelo esclarecimento de muitas dúvidas que me surgiram.

A todas as pessoas que trabalham na MOG Technologies e MOG Solutions pelo apoio e disponibilidade em ajudar.

Aos meus amigos que sempre me acompanharam no meu percurso académico e me incentivaram a fazer o meu melhor.

Muito Obrigado,

José Carlos Marques Rodrigues

Conteúdo

Lista de Figuras

Lista de Tabelas

Glossário

ASF - Advanced Streaming Format

CAB – Ficheiros Cabinet

COM - Component Object Mode

Crash – Programa deixa de funcionar como o devido a ocorrência de um erro

Demuxer – Abreviatura de demultiplexer

DOM - Document Object Model

DV - Digital Vídeo

DVD – Digital Vídeo Disc

EXE – Ficheiros executáveis

FLV – Flash Video

Frame – Unidade de tempo mais pequena do vídeo

GXF - General eXchange Format

HD – High Defenition

Keyframe – Uma frame em que certos parâmetros estão guardados ou assinalados

Logging – Acto de acrescentar conteúdos, como metadados, sobre um determinado vídeo

Mark in – *Timecode* inicial de uma cena

Mark out – *Timecode* final de uma cena

MOV – Formato de ficheiro usado pelo Quicktime

MPEG 4 - Moving Pictures Experts Group 4

MXF – Material Exange Format

Muxer – Abreviatura de multiplexer

Software Stand Alone – Não depende de outro software para correr a não ser do sistema operativo.

SDI - Serial Digital Interface

Streaming – Reproduzir vídeo imediatamente enquanto este faz *download* da Internet, ao contrário de o guardar no computador e em seguida o reproduzir

Timecode – Formato do tempo, representado no formato Horas:Minutos:Segundo:Frames

VCD – Vídeo Compact Disc

WMA – Winsdows Media Audio

WMV – Windows Media Vídeo

W3C - World Wide Web Consortium

Capítulo 1

Introdução

O presente trabalho consiste no relatório de estágio, realizado como parte integrante e conclusiva do Mestrado Integrado Engenharia Informática e Computação, leccionado na Faculdade de Engenharia da Universidade do Porto.

O estágio teve como tema de trabalho "Portal de Visualização de Contéudos Multimédia de Acesso Condicionado", mais precisamente, desenvolver um Player de vídeo Web que interprete certos formatos de vídeo, e que obedeça a requisitos chave.

Neste capítulo pretende-se dar a conhecer o contexto em que o projecto foi desenvolvido.

1.1 Contexto /Enquadramento

O Projecto Portal de Visualização de Conteúdos Multimédia de Acesso Condicionado, foi desenvolvido na empresa MOG Technologies, S.A. Este protótipo terá de ser integrado num projecto mais amplo desenvolvido por parte da empresa, esse projecto será uma *ingest station* para produtoras ou estações de televisão ou cinema. O *Player* de vídeo a ser desenvolvido tem como principais funções a possibilidade de se visualizar os vídeos que estão a ser transmitidos para a *ingest station*, que estão armazenados, ou que estão a ser passados para o servidor central da empresa cliente.

1.2 Projecto

O projecto aqui apresentado consiste na realização de um *player* de vídeo, capaz de ser carregado numa página *web* que possa ter todas as funções básicas de qualquer *player* encontrado na *web* mas também ser capaz de reproduzir filmes no formato *mxf*, ser capaz de marcar tempos,

isto é, marcar os instantes tanto iniciais ou finais de uma cena do vídeo que o utilizador pretende fazer, esses tempos são chamados *mark in timecode* para o tempo inicial e *mark out timecode* para o tempo final o intervalo entre esses dois tempos corresponde a uma cena do filme, de se mover o filme *frame* por *frame*, representar os seus *timecodes*, e escolher os canais de áudio que se pretende ouvir.

O *player* irá ser integrado numa plataforma de *ingest* desenvolvida na empresa, com tecnologia *flash* na *framework flex builder*.

1.3 Motivação e Objectivos

É de esperar que quando o estágio for dado como concluído os seguintes objectivos tenham sido cumpridos.

- Analisar as tecnologias existentes para o desenvolvimento de uma aplicação adequada nomeadamente protocolos de *streaming* de vídeos, e escolher a adequada
- Definir os requisitos do sistema
- Definir a arquitectura e identificar os vários componentes a desenvolver
- Desenvolver os módulos/componentes da aplicação
- Integrar os módulos/componentes e testar a aplicação

1.4 Estrutura do Documento

Pretende-se nesta secção fazer uma breve apresentação da estrutura do presente relatório assim como destacar os temas abordados, sendo esses os requisitos para o desenvolvimento do protótipo, a sua implementação e o estudo das possíveis tecnologias para o desenvolvimento do projecto e as razões de escolha de uma tecnologia. Em seguida é apresentada uma análise geral do trabalho para que seja possível enquadrar o leitor no projecto, e toda a sua envolvente. Nos capítulos seguintes são apresentadas várias tecnologias estudadas e as escolhidas para a realização do protótipo, seguindo-se os detalhes do projecto. Por fim são apresentadas algumas conclusões e melhorias possíveis para o trabalho desenvolvido.

Capítulo 2

Problema e Enquadramento

Neste capítulo pretende-se dar a conhecer o projecto desenvolvido em mais detalhe, assim como toda a sua envolvente. É apresentada também a metodologia de desenvolvimento adoptada, e o plano de trabalho delineado para que o protótipo seja desenvolvido com sucesso dentro dos prazos estipulados.

2.1 Visão Geral

O projecto de desenvolvimento do protótipo de um *player*, enquadra-se num projecto mais alargado da empresa, projecto esse que tem como mercado alvo a multimédia profissional, logo todas as decisões a serem tomadas tem de ter em atenção este ponto, pois este objectivo tem de estar sempre presente.

Esse projecto mais alargado é conhecido como sendo uma *ingest station*, chamado de *MISS*, isto é, um produto que serve como intermediário de passagem dos vídeos filmados pelos jornalistas ou operadores de câmaras e o repositório de vídeos nas editoras, produtoras ou estúdios de televisão. Tem como função uma primeira edição básica dos filmes, assim como transcodificação de alguns formatos. Podem existir vários formatos diferentes de entrada dos filmes na estação, estes estão exemplificados e explicados na secção 4.2 Diagrama de Arquitectura do presente documento. O protótipo irá se enquadrar numa aplicação do tipo cliente-servidor, num ambiente controlado, pois todos os acessos serão controlados. Esta aplicação será uma *ingest station* para estações de televisão ou produtoras. O *player* irá correr do lado do cliente e não do lado do servidor.

O *player* terá de ser capaz de tocar filmes no formato *mxf* (Material Exchange Format) entre os formatos de vídeo mais conhecidos, ser capaz de tocar os filmes através de protocolos de *streaming*, pois podemos ter de passar vídeos de *gigabytes*, logo esta opção não poderá ser descuidada.

A *ingest station* é um componente desenhado para ser um assistente dos especialistas de captura, edição e *logging*. Este componente é frequentemente o primeiro da linha de produção, pois é o primeiro ponto de aquisição de vídeo tanto pelo modo tradicional a cassete, ou os novos sistemas sem cassete. Isolando esta tarefa no servidor, outros computadores ou servidores estão livres para efectuarem tarefas mais específicas no processo colaborativo. A estação pode fornecer virtualmente suporte para qualquer formato, desde DV até HD não comprimido, dependendo das necessidades dos utilizadores.

2.2 Descrição do Problema

O projecto consiste na realização de um *Player* capaz de ser executado num *web* browser, tendo como base um *player* já desenvolvido pela empresa, mas sendo um programa *stand alone*. O *player* tem de ter a capacidade de ler ficheiros do tipo *MXF*, entre os formatos vídeo mais conhecidos. Tem também de ser capaz de marcar *timecodes*, para poder ser feita uma lista de edição do vídeo, para isso tem de ser capaz de navegar no vídeo de *frame* a *frame*. A possibilidade de se poder escolher os diferentes canais de áudio é também relevante, pois como se trata de um *player* para o meio da multimédia profissional, podendo no máximo serem quatro canais de áudio e um de vídeo.

2.3 Metodologia Usada

A metodologia a ser usada no decorrer do estágio, irá ser uma metodologia em espiral, pois é um processo iterativo que percorre todas as fases do projecto várias vezes, esta metodologia será a ideal a ser usada pois existem presentes duas metas, dadas pela parte da empresa, com um protótipo funcional, sendo a primeira no início de Abril e outra no final do período de desenvolvimento do estágio.

Esta metodologia consiste na separação das iterações em quatro grandes áreas, sendo essas áreas as seguintes, definição de objectivos e requisitos, identificação e resolução de riscos, implementação, testes e validação e planeamento da nova iteração.

Um pequeno esquema é apresentado a seguir para que se possa perceber como o modelo usado é apresentado e a distribuição das diferentes fases é homogénea, isto é, têm aproximadamente a mesma duração temporal.

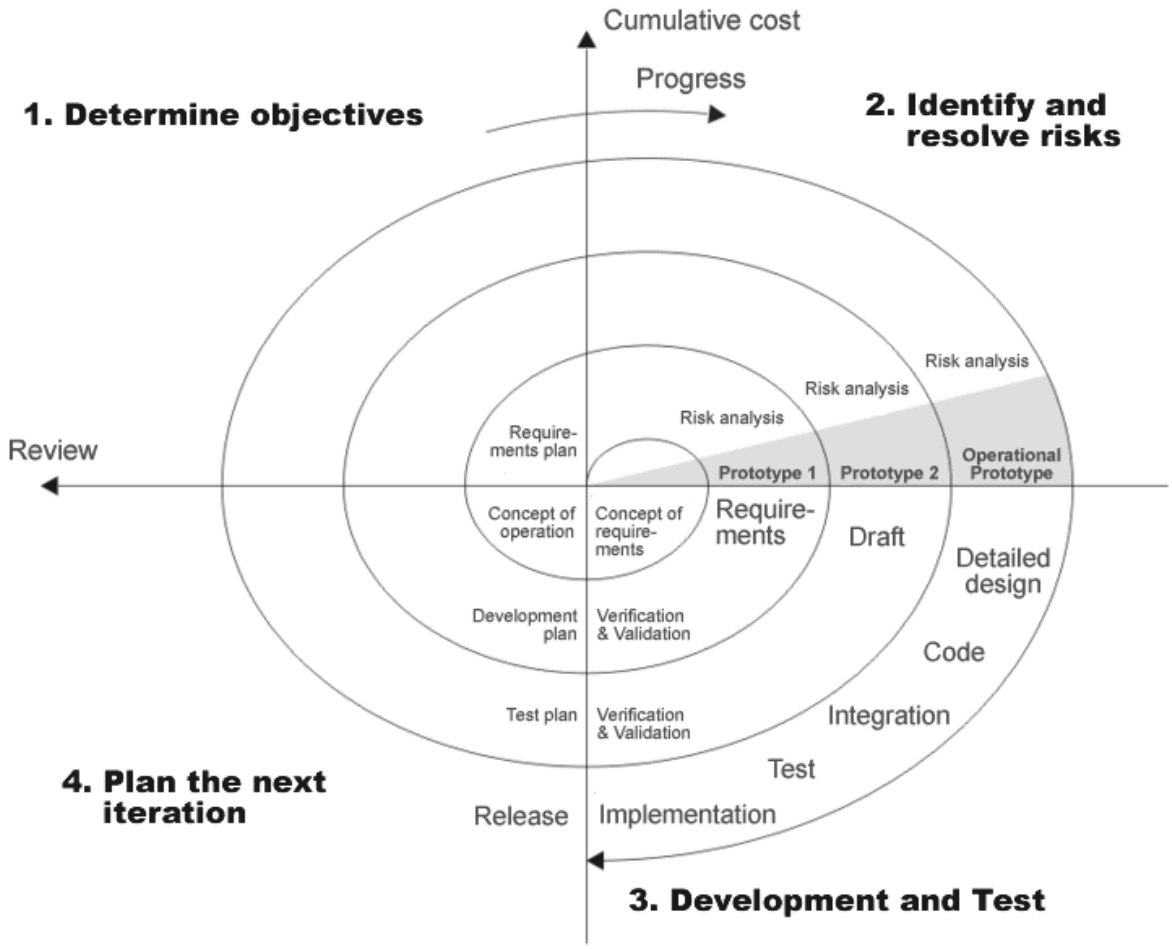


Figura 1 - Modelo em Espiral

2.4 Análise do Problema

O *player* tem de ser capaz de, carregar um vídeo a partir de um *url*, avançar e retroceder num filme *frame* por *frame*, avançar e retroceder no filme livremente, fazer *play*, *pause*, *stop*, escolher que canais de áudio usar, fazer *mark in* e *mark out* dos vídeos, enviar os tempos de *mark in* e *mark out* para a aplicação para poderem ser criadas cenas.

Deve manter uma comunicação fiável com a plataforma principal de modo a que não exista nenhum tipo de *crash* para que não seja necessário fazer nenhum tipo de reiniciar da aplicação, por parte do lado do cliente.

2.5 Plano de Trabalho

Definição de Requisitos

Duração: 15 dias

Início: 18 de Fevereiro de 2008

Fim: 7 de Março de 2008

Este processo não foi de elevada complexidade devido à finalidade do projecto a ser desenvolvido, todos os requisitos são essenciais para o sucesso do protótipo funcional, e estavam bem delineados. Foram delineados alguns testes nestes nesta fase a serem efectuados sobre os protótipos para que todos os requisitos estivessem implementados.

Estudo de Tecnologias

Duração: 20 dias

Início: 18 de Fevereiro de 2008

Fim: 14 de Março de 2008

O estudo das tecnologias possíveis de serem utilizadas decorreu simultaneamente com a definição dos requisitos do protótipo, pois como explicado mais à frente, algumas desses tecnologias não satisfaziam parte dos requisitos encontrados, não podendo ser possíveis de ser utilizadas pois o projecto não ficaria completo.

Desenvolvimento de Protótipo Funcional

Duração: 23 dias

Início: 10 de Março de 2008

Fim: 9 de Abril de 2008

Nesta fase de desenvolvimento foram implementados os controlos *ActiveX* para o *player stand-alone* da empresa, de modo a que este fosse capaz de ler ficheiros *mxf* a partir de um dado *Url*,

estando incorporado numa página *web*. O player terá de ser capaz de comunicar com o *Web* browser através de *javascript* e correcção de alguns erros.

Estudo de nova tecnologia

Duração: 11 dias

Início: 11 de Abril de 2008

Fim: 2 de Maio de 2008

Nesta segunda fase de estudo de tecnologia, foi estudada a hipótese do uso de *quicktime* para o *player* a ser utilizado. Este novo estudo sobre uma nova tecnologia levou a uma análise dos requisitos já definidos.

Redefinição de Requisitos

Duração: 10 dias

Início: 21 de Abril de 2008

Fim: 2 de Maio de 2008

Foram analisados os requisitos já definidos anteriormente e foram redefinidos caso necessário pois algumas funções não muito claras no início do desenvolvimento do projecto já seriam mais fáceis de perceber implementar.

Desenvolvimento do Produto Final

Duração: 33 dias

Início: 30 de Abril de 2008

Fim: 31 de Maio de 2008

Esta fase de desenvolvimento foi a mais extensa pois foram corrigidos alguns erros presentes na primeira versão apresentada e adicionadas funcionalidades extra assim como a estabilidade do protótipo foi bastante melhorada, e o seu desempenho. Foi também testado o uso do *ActiveX* numa aplicação Adobe *AIR*.

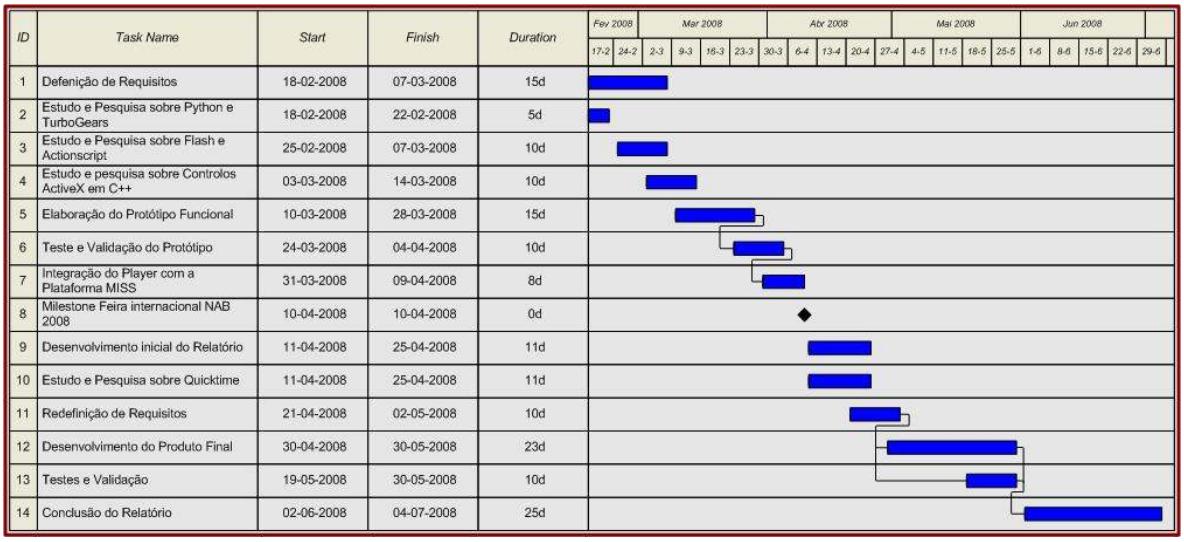
Escrita do relatório

Duração: 25 dias

Início: 2 de Junho de 2008

Fim: 4 de Julho de 2008

A escrita do relatório foi a última fase de todo o projecto, para tal foram usados alguns documentos criados durante as restantes fases do projecto. E consulta de algumas páginas *web* encontradas durante a fase de estudo de tecnologias.



ID	Task Name	Start	Finish	Duration
1	Defenição de Requisitos	18-02-2008	07-03-2008	15d
2	Estudo e Pesquisa sobre Python e TurboGears	18-02-2008	22-02-2008	5d
3	Estudo e Pesquisa sobre Flash e Actionscript	25-02-2008	07-03-2008	10d
4	Estudo e pesquisa sobre Controlos ActiveX em C++	03-03-2008	14-03-2008	10d
5	Elaboração do Protótipo Funcional	10-03-2008	28-03-2008	15d
6	Teste e Validação do Protótipo	24-03-2008	04-04-2008	10d
7	Integração do Player com a Plataforma MISS	31-03-2008	09-04-2008	8d
8	Milestone Feira internacional NAB 2008	10-04-2008	10-04-2008	0d
9	Desenvolvimento inicial do Relatório	11-04-2008	25-04-2008	11d
10	Estudo e Pesquisa sobre Quicktime	11-04-2008	25-04-2008	11d
11	Redefinição de Requisitos	21-04-2008	02-05-2008	10d
12	Desenvolvimento do Produto Final	30-04-2008	30-05-2008	23d
13	Testes e Validação	19-05-2008	30-05-2008	10d
14	Conclusão do Relatório	02-06-2008	04-07-2008	25d

Figura 2 - Diagrama de Gant

Capítulo 3

Revisão Tecnológica

No presente capítulo são apresentadas as tecnologias que existem no mercado como possíveis concorrentes para o produto em desenvolvimento. E também as tecnologias estudadas, e qual a usada, as razões da sua escolha e porque não foram escolhidas as outras.

3.1 Soluções Existentes

Existem várias soluções no mercado para reprodução de vídeos, muitas delas são gratuitas para qualquer sistema operativo, podendo grande parte delas serem incorporadas numa página *web* muito facilmente. É o caso por exemplo do *Windows Media Player*, desenvolvido pela Microsoft é um *player* de media digital e uma biblioteca de media que é usado para tocar vídeo, áudio e para a visualização de imagens num computador pessoal usando o sistema operativo da Microsoft, assim como em *Pocket Pcs*. Os formatos base que o *Windows media Player* consegue tocar são, WMV, WMA e ASF. Suporta leitura de vídeo ou áudio do computador local ou através de rede ou Internet com *download* progressivo ou *streaming*. As operações que podem ser efectuadas na media que o Windows media *player* está a tocar são só, *play*, *pause*, *stop*, *forward*, *backward* e *seek* de um determinado tempo. Contém um controlo *ActiveX* que pode ser embebido em páginas *web* para que se possa correr o *Windows Media Player* em qualquer página *web*.

O VLC media player é um *programa* gratuito, desenvolvido pelo projecto *VideoLan*[1], é um portável *player* de vídeo, áudio, codificador e *streamer*, suportando muitos formatos de áudio e vídeo, assim como *DVDs*, *VCDs* e vários protocolos de *streaming*. É capaz de fazer *stream* através da rede e transcodificar diferentes formatos de vídeo. O VLC é um dos melhores *players* multi-plataformas disponíveis no mercado com versões disponíveis para variados sistemas operativos, como BSD, *Linux*, *Mac OS X*, *Windows*, *Solaris* entre outros. É capaz também de usar um variado número de bibliotecas de codificação e descodificação grátis, muitos desses *codecs*

[1] http://www.videolan.org/vlc/

são dados pela livraria *libavcodec* do projecto *Ffmpeg*[2]. O *vlc* usa principalmente os seus próprios *muxers* e *demuxers*. Também ganhou distinção como o primeiro *player* capaz de tocar *DVDs* encriptados em *linux* usando a biblioteca *libdvdcss*.

As soluções existentes no mercado não são direccionadas para o meio da multimédia profissional, são soluções mais simplificadas que permitem visualizar um vídeo mas não permite que possamos marcar cenas ou navegar no filme *frame* por *frame*. Estas soluções não têm como alvo o mercado da multimédia profissional, mas sim o público em geral, podemos encontrar essas soluções semelhantes em *web sites* como o *youtube*[3], ou *metacafe*[4]. A finalidade destas soluções não se pode equiparar à apresentada pelo projecto. As soluções equivalentes são geralmente feitas à medida de um cliente ou com versões de *software stand-alone* e não soluções para serem disponibilizadas num *web browser*, este produto é pioneiro na área.

Existem porém três *players* com as mesmas capacidades que se pretendem para o *player* a ser desenvolvido durante este estágio, mas estes só existem em versão *stand-alone*. São soluções semelhantes ao protótipo final deste projecto, mas não iguais pois não podem ser integradas numa página web. As três empresas que desenvolveram essas soluções são, *OpenCube Technologies*[5], *Snell&Wilcox*[6] e *Drastic Technologies*[7].

3.2 Hipóteses de Tecnologias possíveis de serem utilizadas

Duas hipóteses de tecnologias que foram colocadas no início do desenvolvimento do projecto foram o uso de *Flash* com *ActionScript* ou o uso de controlos *ActiveX* sobre uma aplicação já existente na empresa. Mais tarde foi proposto o estudo do uso do *player* comercializado pela *Apple* o *Quicktime*.

[2] http://ffmpeg.mplayerhq.hu/

[3] www.youtube.com

[4] www.metacafe.com

[5] http://www.mxftk.com/pub/

[6] http://www.snellwilcox.com/news_events/press_releases/95

[7] http://www.drasticpreview.com/

3.3.1 Flash e Actionscript

3.3.1.1 O que é o Flash e o Actionscript

O *Flash* no seu início começou por ser uma tecnologia para construir páginas interactivas e com animações, começando por ser uma animação baseada em vectores, mas rapidamente a *Macromedia*, começou a fazer o *flash* mais controlável programaticamente, através da linguagem conhecida como *Actionscript*. Esta evolução deveu-se à necessidade de suporte de *streaming* bi-direccional tanto de áudio como de vídeo, à capacidade de tocar vídeos, ao suporte de *Flash* em todos os *browsers* e alguns dispositivos móveis e mais recentemente ao desenvolvimento de *rich-internet applications*.

Com a última versão no mercado o desenvolvimento de aplicações em *Flash* e *Actionscript* tornou-se mais simples pois a linguagem de programação tornou-se orientada a objectos, tornando a sua manipulação e organização bastante mais simples.

3.3.1.2 Estudo e Testes sobre *Flash* e Actionscript 3.0

Devido à maior parte de *players Web* serem desenvolvidos em *Flash* esta hipótese teve um grande peso na fase de investigação do projecto, foram desenvolvidos diversos testes para que fosse possível avaliar se esta tecnologia seria a ideal para o desenvolvimento do projecto. Os critérios usados para os testes tiveram em consideração não só se seria possível desenvolver todo projecto com um produto final bastante bom, mas também se a implementação do projecto seria de um nível de dificuldade razoável ou extremamente elevada. Por se tratar de uma linguagem orientada a objectos provou-se que com o uso do *flash* seria uma implementação do projecto simples, pois as bibliotecas usadas pelo *actionscript* permitem um uso simples dos objectos de vídeo e de uma personalização total de todos os aspectos visuais de um portal *web*. Mas devido a tratar-se de um *player* para uso no meio profissional da multimédia, não poderá apenas ter as opções clássicas de *play*, *stop*, *pause*, *forward* e *backward*. A contagem do tempo tem também de ser mais específica, isto é, tem de ser contado até às *frames* e não só até aos segundos como é o normal, isto porque, se trata de um produto para o meio da multimédia profissional. O tempo nessa configuração é também conhecido por *timecode*. Tem de ter esta configuração para que o filme possa ser cortado no tempo exacto, e não no aproximado.

Após ter em atenção estes pormenores é que foi possível avaliar se o *flash* mais *actionscript* seria uma opção a ser utilizada. Aqui foi detectada a primeira impossibilidade no uso do *flash* e *actionscript* para a implementação do projecto, pois não se consegue contar o tempo *frame* a *frame*, nem existe forma de saber a quantas *frames* por segundo o filme está a passar.

Outra consideração a fazer foi se seria possível andar no tempo do filme para a frente ou para trás *frame* por *frame*, isto também não é possível no *flash*, pois não existem bibliotecas para tal,

esta função é também fundamental, pois para que seja mais fácil de se marcar os *timecodes* correctos são necessárias funções no *player* que permitam navegar no filme *frame* por *frame*.

Por fim e uma das considerações a ter mais em conta, foi os formatos de ficheiro que o *flash player* consegue interpretar, grande parte dos formatos usados pelos futuros clientes deste projecto usam ficheiros do tipo *mxf* ou *mpeg4*, que são ficheiros de vídeo de alta definição. Posto isto e após investigação sobre o assunto, concluiu-se que o *flash player* consegue interpretar ficheiros do tipo *mpeg4* mas não consegue interpretar ficheiros do tipo *mxf*. Para tal seria necessário converter os ficheiros em formato *flv* o que não seria impossível, mas como parte dos filmes que serão passados pelo *player* terão como fonte um *P2* ou *XDCAM*, transcodificar os ficheiros de *mxf* para *flv* e estar a reproduzi-los no *player* ao mesmo tempo não seria viável, pois não seria possível em tempo real enquanto estão a ser transcodificados.

3.3.2 Quicktime

3.3.2.1 O que é o Quicktime

O *Quicktime* é uma *framework* multimédia desenvolvida pela *Apple*, capaz de interpretar vários formatos de vídeos digitais, som, texto, animações, música e vários tipos de imagens. O Quicktime está disponível para *Mac OS*, *Mac OS X* e *Windows*. Existem várias bibliotecas para se poder saber mais informações sobre o que o *quicktime* está a correr no momento, e alguns eventos.

Após a opção *flash* e *actionscript* ter sido posta de lado, teve de se procurar outra tecnologia possível de ser utilizada, inicialmente foi feito um protótipo do projecto usando controlos *ActiveX* para usar um *player* que já tinha sido desenvolvido pela empresa.

O *Quicktime* expõe um conjunto de objectos, propriedades e métodos para o *javascript*. Os objectos incluem o próprio *Quicktime*, um *plug-in* para o browser, os filmes embebidos, e todas as pistas incluídas nesses filmes.

Os métodos permitem controlar o filme, começar, parar, andar para a frente ou para trás, ou substituir um filme com outro. Modificar as propriedades permite controlar como o filme se comporta, seleccionar a língua, modificar o tamanho da pista de vídeo, posição, rotação, modificar o volume da pista de som, modificar a *frame rate* do vídeo e direcção, entre outros.

Obter as propriedades permite receber informação como, a versão do *plug-in Quicktime* instalado no computador, a duração do filme, que percentagem do filme já fez *download*, se o filme está a tocar ou já acabou, quantas pistas tem, e mais propriedades podem ser obtidas.

As versões mais recentes do *plug-in* do browser para o *Quicktime* 7.2.1 e seguintes incluem a possibilidade de emitir eventos DOM. Estes eventos podem ser emitidos quando o *plug-in* está instanciado e pronto para interagir como *Javascript*, podem ser despoletados em outros pontos como, durante o processo de carregar o filme, quando o filme começou a tocar, parou, acabou, ou se um evento de erro ocorreu. As funções *Javascript* podem ser feitas para ficar à "escuta" de

eventos *DOM* em particular. Quando um dos eventos acontece a função responsável pelo seu tratamento é chamada.

Isto permite criar páginas *web* que detectem eventos como um filme está a carregar, a tocar, parado ou acabado, sem ter que se criar *loops* temporais com *Javascript* ou estar sempre a perguntar ao *plug-in* pelo seu estado. Permite também executar funções *Javascript* em resposta a vários eventos do filme sem se utilizar *urls* do tipo *javascript://*.

O *plug-in* usa eventos DOM *standard*, para que qualquer browser que suporte Eventos *DOM* Nível 2 segundo a especificação da W3C[8] se use o método *addEventListener*() para monitorizar os eventos pretendidos.

Browsers como *Safari* 3.0 ou *FireFox* 2.0 são exemplos de suporte dos *standards* da W3C, no *Internet Explorer* versão 5 ou seguintes o suporte para os eventos tem de ser ligeiramente diferente, usando um método *attachEvent*(), em vez de usar o método referido mais acima, o *addEventListener()*.

Para que a página *web* funcione correctamente quer em browsers que suportem os standards W3C ou no *Internet Explorer*, deve-se incluir código *Javascript* tanto para o método *addEventListener*() ou para o método *attachEvent*().

3.3.2.2 Estudo e Testes sobre Quicktime

Após alguma pesquisa sobre outras tecnologias possíveis de serem utilizadas, foi proposto a investigação do uso do *quicktime* para o desenvolvimento do projecto. Através do estudo desta tecnologia foram identificados métodos *javascript* presentes no *quicktime* que permitem saber quantas unidades de tempo existem num segundo, o tempo total do filme, as *frames por segundo* que o filme está a tocar.

Após esta primeira análise conclui-se que as operações pretendidas a serem efectuadas pelo *player* de vídeo seriam possíveis de serem feitas, pois sabendo o *frame rate* do filme e as unidades de tempo presentes em cada segundo, poderia-se dividir esse número pela *frame rate* para saber quanto tempo duraria cada *frame*, podendo o *timecode* do filme ser assim representado.

Na biblioteca de *javascript* existem também métodos que possibilitam ir para um instante específico no filme, ou andar para frente ou para trás *frame* por *frame*. Posto isto quase todos os requisitos estariam cobertos por esta plataforma, mas como o *quicktime* só interpreta filmes no formato *.mov*, surgiu aqui um impedimento que também tinha aparecido no *flash*. O *Quicktime* não podendo ler formatos *.mpeg4* ou *.mxf* teria de haver transcodificação de formatos levando ao mesmo problema já do *flash*, não é possível existir transcodificação do vídeo e este tocar em tempo real.

[8] Especificação dos eventos DOM nível 2 - http://www.w3.org/TR/DOM-Level-2-Events/

3.3.3 Controlos ActiveX

3.3.3.1 O que é o ActiveX

ActiveX é usado pela *Microsoft* para descrever um número das suas tecnologias *COM*. No entanto quando se diz *ActiveX*, está-se a referir aos controlos *ActiveX*, a resposta da *Microsoft* às *Java Applets*. As duas tecnologias são semelhantes, pois estão desenhadas para serem transferidas e executadas pelos *Web Browsers*. A diferença é que enquanto os controlos *ActiveX* conseguem uma comunicação mais efectiva com o *Microsoft Windows* do que as *Java Applets*, não oferecem nenhum suporte de *cross-platform*.

3.3.3.2 O que são os Controlos ActiveX

Os controlos *ActiveX* são pequenos programas que podem ser usados para criar aplicações distribuídas que funcionam na *Internet* através de *web browsers*. Como alguns exemplos temos aplicações personalizadas para recolha de dados, visualizar alguns tipos de ficheiros ou visualizar animações. A sua especificação é uma extensão do *Microsoft Windows* e a *Object Linking and Embedding API*. As aplicações *ActiveX* são usadas principalmente no *Internet Explorer*.

Tal como as *Java Applets* os controlos *ActiveX* correm no computador cliente e não no servidor.

Os controlos *ActiveX* podem ser desenvolvidos em *.NET*, *Visual Basic*, *C#*, ou *C++*.

3.3.3.3 Estudo e Testes sobre Controlos ActiveX

Esta tecnologia foi estudada em paralelo com a opção *quicktime*. Esta possibilidade surgiu pois na empresa já existia uma aplicação *stand-alone* de um *player* de vídeo capaz de cumprir todas as especificações.

Foram realizados vários testes de como usar uma aplicação externa, com controlos *activeX*, para ser suportado num *web browser*. Devido a ser uma tecnologia desconhecida até aqui esta aprendizagem foi um bocado demorada, pois é bastante complexo o modo de como se implementa. Todos os testes foram desenvolvidos em *C++*.

Parte destes testes também tiveram em atenção que o *player* deveria conseguir comunicar com o *browser* e também com *flash*, pois este irá ser integrado numa aplicação *flash*. Posto isto

tiveram de ser feitos também alguns testes que tinham como objectivo testar a comunicação entre javascript e o controlo *ActiveX*, estes testes tiveram bastante sucesso, revelando-se mais simples que o esperado. O mesmo sucesso foi obtido na comunicação entre *javascript* e *flash*, de modo a que dados obtidos através do *player* possam ser enviados para a plataforma onde este se encontra a correr, de modo a que os tempos sejam enviados para a base de dados.

3.4 Tecnologia Escolhida para a implementação do protótipo.

Após todas as considerações que tiveram que ser tomadas e todos os testes a decisão da tecnologia a ser usada na implementação do protótipo foram os controlos *ActiveX*, pois a existência do *player*, o MOG Player, que já cumpria todos os requisitos foi bastante relevante. Os controlos *ActiveX* tiveram de ser implementados para que fosse possível usar a aplicação já desenvolvida num *web browser*. Esta opção teve algumas consequências no protótipo pois usando os controlos *ActiveX* a única hipótese será correr a aplicação que contem o *player* em *Internet Explorer*.

3.5 Tecnologias para a Instalação do Player de vídeo

Para que fosse o menos complicado possível para utilizador final instalar o *player* pronto a funcionar, teve de se implementar um instalador. Para tal havia duas opções, ou utilizar um ficheiro do tipo *cabinet*, ou também conhecidos por ficheiros *.cab*, ou utilizar um ficheiro de instalação do tipo executável ou também conhecido como *.exe*.

3.5.1 Ficheiro Cabinet

Este tipo de ficheiros caracteriza-se por comprimir os ficheiros de uma forma bastante eficaz, e por ser possível fazer uma instalação automática, isto é, no objecto *html* que representa o *player* de vídeo, existe um parâmetro que representa o *codebase*, se este parâmetro estiver identificado e o objecto que está referido não estiver instalado no computador, o *browser* automaticamente irá fazer o *download* e instalar para que a página funcione correctamente com todos os elementos

disponíveis. Desta forma a utilização do produto por parte do utilizador será simplificada de forma significativa.

Um problema que surgiu do uso desta forma de instalação foi a necessidade de se ter que efectuar dois registos de duas licenças no serviço de registo do *windows* de um filtro de vídeo desenvolvido pela empresa, devido à política da empresa as licenças dos produtos não puderam ir no mesmo ficheiro de instalação que o resto do produto. Pois como neste caso só deverá ser passado um ficheiro *cabinet*, tudo deverá ir no mesmo ficheiro, logo impossibilitando esta escolha.

3.5.2 Ficheiro Executável

Na realização deste tipo de ficheiro foi usado um *software* específico para a sua implementação, sendo esse software o *Ghost Edit*, este programa tem uma série de funções já implementadas que facilitam o desenvolvimento de ficheiros de instalação, assim como uma *API* bastante desenvolvida para uma fácil implementação.

Este tipo de ficheiros é por todos nós conhecido, a sua utilização é bastante fácil o que permite que qualquer utilizador os possa usar, podem ser distribuídos facilmente através de *email* e *websites*, assim como ficheiros separados podem ser distribuídos facilmente.

3.5.3 Tecnologia Escolhida para a instalação do Player de vídeo

Devido à elevada importância para a empresa, de alguns ficheiros terem de ser distribuídos separadamente a escolha de tecnologia a ser usada para a instalação do player de vídeo teve de ser dependente deste factor, logo, a escolha lógica foi os ficheiros executáveis.

Este tipo de ficheiro foi escolhido por duas razões principais, primeiro por ser possível disponibilizar com ficheiros separados, e pela facilidade de implementação em relação a implementação de ficheiros *Cabinet*.

Capítulo 4

Arquitectura do Sistema

Neste Capítulo irá ser apresentada a arquitectura do Sistema em geral, ou seja a arquitectura de toda a envolvente do projecto para que seja mais simples o entendimento de onde o protótipo se enquadra e como funciona. Em seguida são apresentados os requisitos funcionais e não funcionais, a arquitectura do projecto, os casos de uso, o diagrama de componentes, diagrama de actividades de todo o projecto, e dois diagramas de sequência para da utilização do *player* para fazer uma cena no vídeo ou para editar uma cena já criada.

4.1 Requisitos

4.1.1 Requisitos Funcionais

Na presente secção serão apresentados os requisitos funcionais do projecto, todos estes requisitos terão de ser cumpridos para que o protótipo seja concluído com sucesso. Pois se algum destes requisitos não estiver presente funções essenciais para o manuseamento do *player* de vídeo.

- Reproduzir o filme – O sistema permitirá reproduzir o filme após lhe ser passado com sucesso o *url* onde este se encontra, podendo ser possível fazer *play*, *stop*, *pause*, *backward*, *forward*.
- Fazer *mark in* – Deverá ser possível marcar o *timecode* inicial de uma cena.
- Fazer *mark out* – Deverá ser possível marcar o *timecode* final de uma cena.
- Mover uma *frame* para a frente – O sistema permitirá mover o filme para a frente *frame* a *frame*
- Mover uma *frame* para trás – O sistema permitirá mover o filme para trás *frame* a *frame*

- Passar os tempos para fazer cena – Deverá ser possível passar para a plataforma *MISS* os tempos marcados em *mark in* e *mark out*, de forma a ser possível criar uma cena.
- Fazer *load* do *url* – Em comunicação com a plataforma *MISS*, após o utilizador escolher um filme deverá fazer carregar o filme que se encontra no *url* fornecido pela plataforma.
- Fechar Filme – O sistema permitirá fechar um filme, isto é, apagar o ficheiro do *buffer*.
- Canais de som – O sistema permitirá ao utilizador escolher os canais, de entre os quatro possíveis, que este pretende utilizar sempre que esta operação seja possível de ser efectuada.
- Volume de Som – O sistema permitirá ao utilizador regular o volume do áudio enquanto este reproduz os vídeos.
- Controlar o *Slider* – O sistema deverá permitir controlar o *slider*, correspondente ao tempo em que o vídeo se encontra para o utilizador navegar para a frente ou para trás no vídeo mais livremente.

4.1.2 Requisitos Não Funcionais

Os requisitos não funcionais representam os requisitos de operacionalidade do sistema, onde o *player* de vídeo irá correr.

- Ligação permanente a plataforma *MISS* – O sistema deverá estar sempre em comunicação com a plataforma *MISS*, pois a qualquer momento o utilizador poderá querer fazer uma nova cena ou visualizar uma cena já feita.
- Formatos dos filmes – Deverá aceitar vários formatos dos ficheiros de vídeo, como, ficheiros *.mxf* ou *.gxf*.
- Interface Intuitiva – A interface do sistema deverá ser o mais simples possível de modo a que seja intuitiva para o utilizador de modo a que seja fácil de trabalhar.
- Sistema Fiável e Robusto – O sistema deverá fazer uma comunicação fiável com a plataforma, e bastante robusto pois deverá ser usado durante horas seguidas.
- Desempenho – O sistema deverá ser o mais rápido possível a efectuar uma resposta ao pedido do utilizador.
- Operacionalidade – O Sistema operativo deverá ser Windows, pois o produto só funciona em Internet Explorer, devido a serem utilizados os controlos *activeX*.
- Codecs – Deveram estar instalados no sistema, operativo, todos os codecs necessários para que seja possível visualizar os ficheiros de vídeo pretendidos.

- Licenças – O Sistema onde estará a correr o produto deverá ter todas as licenças instaladas necessárias para que os ficheiros de vídeo possam ser tocados.
- Software extra – O sistema operativo deverá ter instalado o *flash player*, para que *MXFSpeedRail* possa ser executado.
- Instalação – O sistema operativo deverá ter todos os ficheiros necessários, *dll* e *ax* disponibilizados na instalação, para que o produto funcione correctamente.

4.2 Diagrama de Arquitectura

No diagrama apresentado na figura 3 é desenhada a arquitectura onde o projecto aqui descrito está inserido, como se pode notar o portal de visualização de conteúdos multimédia é uma pequena parte do total, mas tem uma importância bastante elevada, pois como este produto se destina ao mercado da multimédia profissional a visualização dos vídeos para serem trabalhados é bastante importante, pois esse será o produto desenvolvido pelos utilizadores.

A presença do *player* é também bastante importante pois a possibilidade de se efectuar uma pequena pre-edição na plataforma através do seu uso, a possibilidade de visualização do vídeo e marcar tempos nos vídeos para fazer cenas só é possível devido à presença do *player* de vídeo.

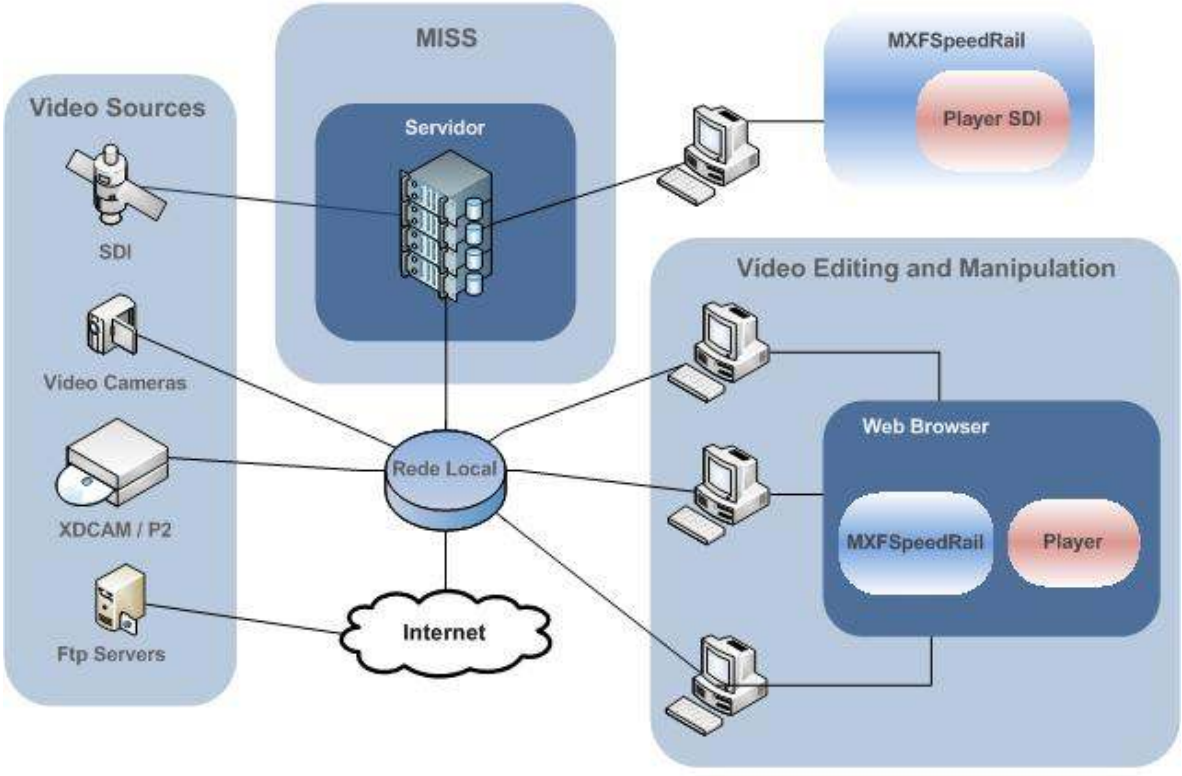


Figura 3 – Arquitectura geral do Sistema

Como podemos observar os conteúdos para serem trabalhados poderão ter diferentes origens, e também diferentes formatos.

XDCAM/P2 são discos, do tamanho aproximado de um dvd mas com capacidade de 23*GB* de memória, algumas câmaras de vídeo das marcas *Sony*[9] ou *Panasonic*[10] conseguem gravar directamente nestes discos.

SDI é um tipo de entrada de dados, que tanto pode ter origem num sinal de satélite como num leitor de cassetes de vídeo, este tipo de utilização é útil, pois converte o sinal analógico, as cassetes, em sinal digital, podendo ser armazenado em seguida para uma mais fácil pesquisa e armazenamento.

Também é possível ir buscar vídeos a um servidor *FTP* para depois serem trabalhados e armazenados no servidor. Outra forma de ingerir os ficheiros é através da passagem de ficheiros de máquinas de filmar que tenham discos rígidos para o disco local do computador onde se está a trabalhar e depois esses ficheiros serem enviados para o servidor.

Todas as entradas, servidores ou terminais estão ligados em rede com a excepção da entrada *SDI*, esta está ligada directamente ao servidor e só se poderá trabalhar neste sinal de entrada no posto local ligado directamente ao servidor. Isto porque, o vídeo não pode ser controlado pelo *player* de vídeo, só pode ser controlado na máquina que está a ler cassete de vídeo, ou no caso de ser um sinal de satélite não poderá ser controlado apenas existirá a opção de gravar o vídeo que se está a receber ou não.

Como é possível observar no diagrama apresentado na figura 3, o utilizador irá interagir com o *player* através de postos de trabalho ligados ao servidor através de rede ou através de um posto directamente ligado ao servidor, utilizando-o como um posto local. O *player* está integrado numa página *web*, que está a correr o produto, *MXFSpeedRail*.

---

[9] http://pro.sony.com/bbsc/ssr/micro-xdcam/

[10] http://www.panasonic.com/business/provideo/home.asp

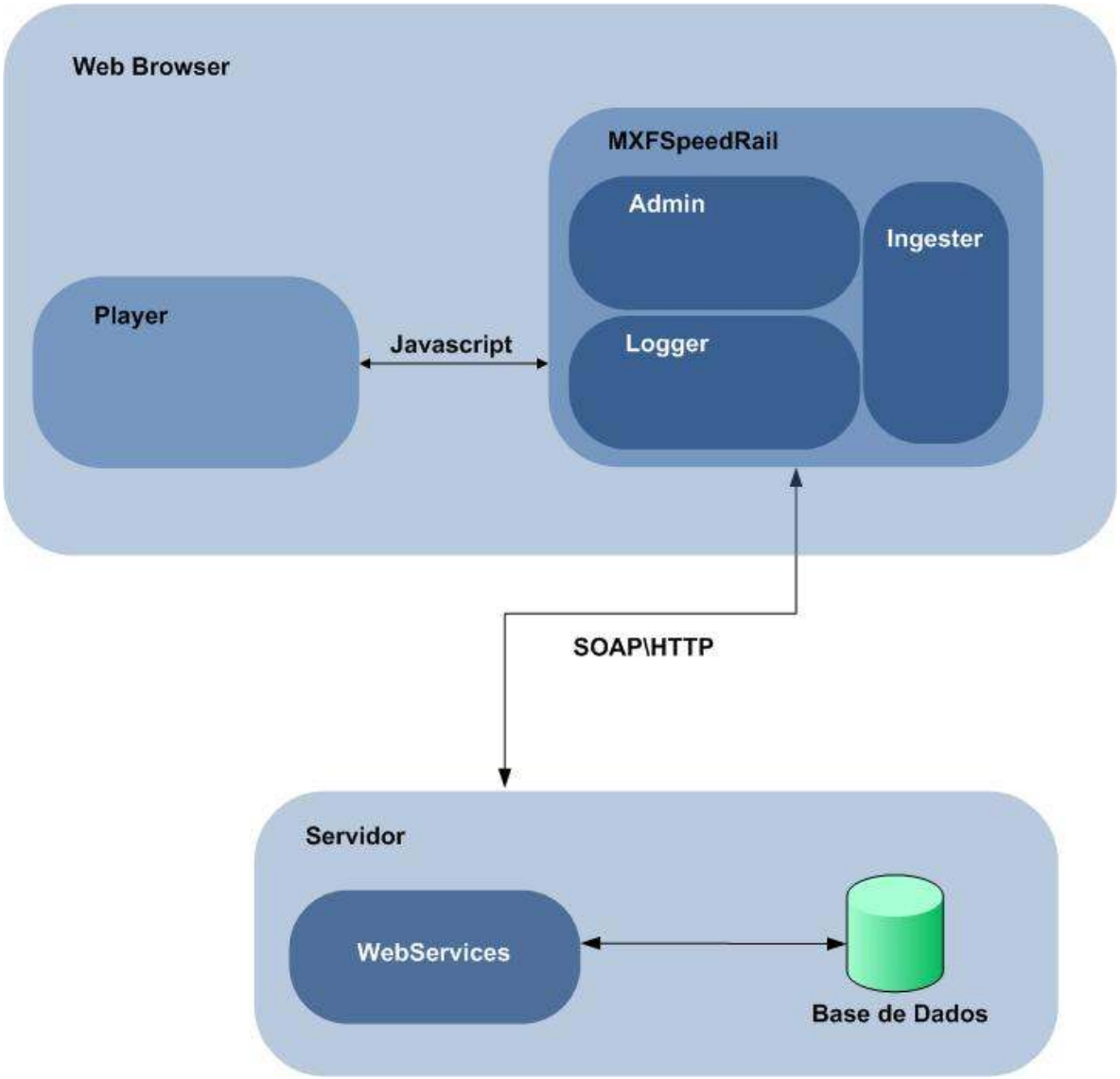


Figura 4 – Arquitectura do Projecto

No diagrama apresentado na figura 4 podemos observar como é efectuada a comunicação entre o *player* de vídeo e o *MXFSpeedRail*, e entre este e o servidor. O *player* e o *MXFSpeedRail* correm simultaneamente no *web browser* e a sua comunicação é intermediada pelo *browser* através de *javascript*, pois não existe forma de o *flash* e *activeX* comunicarem directamente, a não ser por *javascript*. A comunicação do *mxfspeedrail* e o servidor é feita através dos protocolos *http* e *soap*, esta comunicação é responsável pelo fornecimento dos *urls* com os vídeos ao *player*. Para que se dê inicio ao pedido do *Url* o utilizador usa o módulo *Logger*, para escolher um vídeo, em seguida esse módulo pede o *url* aos WebServices, e envia para o *player* através de uma função *javascript*.

4.3 Diagrama de Casos de Uso

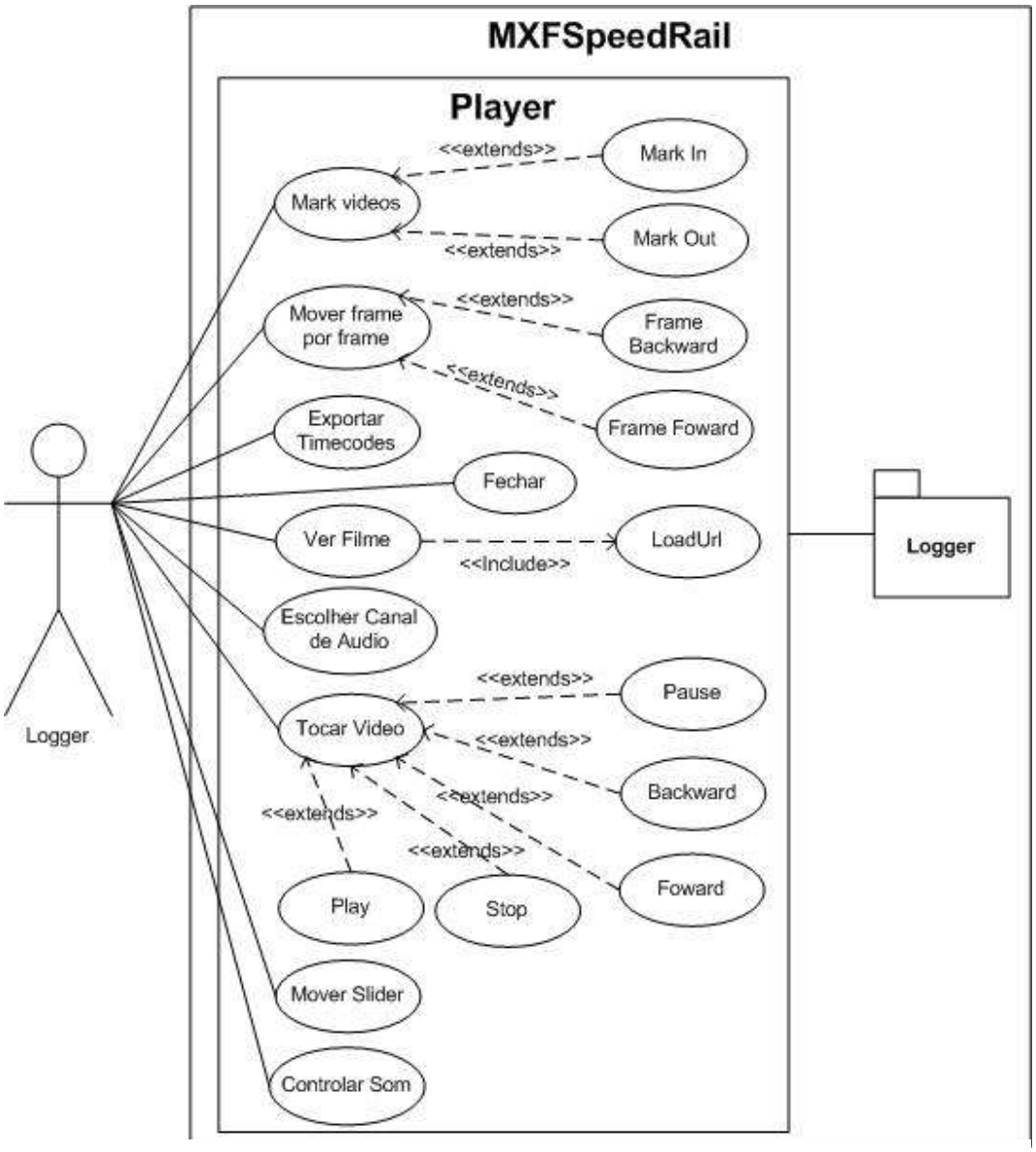


Figura 5 - Diagrama de Casos de Uso

A utilização do *player* só tem intervenção de um actor, aqui chamado de *logger*, este nome é lhe dado devido a ser o utilizador que vai visualizar o vídeo, marcar cenas e/ou anotar os metadados correspondentes ao filme ou cenas que este realizar. Estas funções de utilização são conhecidas como *Logging*.

O *player* apenas interage com o *MXFSpeedRail* através de dois casos, um é o *LoadUrl*, que quando o actor *logger* selecciona o vídeo o módulo *Logger* envia o Url pedido para o *player* com a função chamada *LoadUrl*. E no caso de Exportar *Timecodes* que o *player* envia para o módulo

Logger os *timecodes* da cena submetida pelo actor, *Logger*. Esses *timecodes* submetidos são definidos pelos casos de uso *Mark In* e *Mark Out*, que representam o *timecode* inicial e final da cena respectivamente.

O *logger* terá a possibilidade de efectuar as operações mais clássicas sobre o vídeo, isto é, as operações que estamos mais habituados que são, *play*, *pause*, *stop*, *forward*, *backward* mas também a possibilidade de mover o filme *frame* a *frame*, marcar *timecodes*, escolher os canais de áudio que pretende ouvir assim como exportar os timecodes para o *MXFSpeedRail* e fechar o vídeo para poder visualizar um novo vídeo.

Casos de Uso

Nome	Mark Vídeos
Descrição	Este caso de uso representa quando o utilizador do *player* de vídeo, pretende marcar os tempos inicial ou final de uma cena que pretende efectuar no vídeo
Extends	Mark In, Mark Out

Tabela 1 – Caso de Uso Mark Vídeos

Nome	Mark In
Descrição	Caso de Uso que representa o uso do *player* de vídeo para marcar o tempo inicial de uma cena a ser produzida pelo utilizador

Tabela 2 – Caso de Uso Mark In

Nome	Mark Out
Descrição	Caso de Uso que representa o uso do *player* de vídeo para marcar o tempo final de uma cena a ser produzida pelo utilizador

Tabela 3 – Caso de Uso Mark Out

Nome	Mover Frame por Frame
Descrição	Este Caso de Uso representa quando o utilizador pretende navegar no filme a ser

	tocado pelo *player* de vídeo, *frame* por *frame*, tanto para a frente ou para trás do ponto que este se encontra
Extends	Frame Forward, Frame Backward

Tabela 4 – Caso de Uso Mover Frame por Frame

Nome	Frame Forward
Descrição	Caso de Uso que representa o uso do *player* de vídeo para navegar no filme a ser reproduzido no momento para a frente *frame* a *frame*

Tabela 5 – Caso de Uso Frame Forward

Nome	Frame Backward
Descrição	Caso de Uso que representa o uso do *player* de vídeo para navegar no filme a ser reproduzido no momento para trás *frame* a *frame*

Tabela 6 – Caso de Uso Frame Backward

Nome	Exportar Timecodes
Descrição	Este Caso de Uso representa quando um utilizador pretende enviar os timecodes inicial e final de uma cena, para a plataforma que o *player* de vídeo está a comunicar, neste caso o *MXFSpeedRail*

Tabela 7 – Caso de Uso Exportar Timecodes

Nome	Fechar
Descrição	Caso de Uso que representa fechar o vídeo a ser reproduzido no momento, quando o utilizador já visualizou o vídeo e não pretende realizar mais qualquer tipo de operação sobre o mesmo

Tabela 8 – Caso de Uso Fechar

Nome	Ver Filme
Descrição	Este Caso de uso Representa quando o utilizador da plataforma pretende visualizar um determinado vídeo no *player*, o que para tal tem de ser usado o caso de uso *LoadUrl*
Includes	LoadUrl

Tabela 9 – Caso de Uso Ver Filme

Nome	LoadUrl
Descrição	Este Caso de Uso é usado sempre que o caso de uso ver filme é usado também, pois este não pode ser utilizado directamente pelo utilizador, mas sim é a aplicação onde o *player* de vídeo se encontra que o usa, por forma a disponibilizar os vídeos para utilizador visualizar

Tabela 10 – Caso de Uso LoadUrl

Nome	Escolher canal de Áudio
Descrição	Caso de Uso que representa a escolha de que canal de áudio o utilizador pretende ouvir enquanto o filme está a ser reproduzido, se feita alguma cena com apenas alguns canais de áudio seleccionados, a cena só terá esses canais de áudio gravados

Tabela 11 – Caso de Uso Escolher Canal de Áudio

Nome	Reproduzir Vídeo
Descrição	Caso de Uso que representa todas as operação mais básicas de um *player* de vídeo mais tradicional
Extends	Play, Stop, Pause, Backward, Forward

Tabela 12 – Caso de uso Tocar Vídeo

Nome	Play
Descrição	Caso de uso que representa fazer play do vídeo que se encontra carregado no player e estiver pronto para haver operações sobre o mesmo

Tabela 13 – Caso de Uso Play

Nome	Stop
Descrição	Caso de uso que representa fazer parar o vídeo e por o tempo de reprodução deste no início do vídeo

Tabela 14 – Caso de Uso Stop

Nome	Pause
Descrição	Caso de uso que representa fazer parar o vídeo mas o tempo de reprodução do vídeo ficar também parado com o valor que este tinha quando se carregou no botão

Tabela 15 – Caso de Uso Pause

Nome	Backward
Descrição	Caso de uso que representa o utilizador poder se deslocar no tempo do filme mais rapidamente para a frente, do que quando este está a fazer a operação de *play*.

Tabela 16 – Caso de Uso Backward

Nome	Forward
Descrição	Caso de uso que representa o utilizador poder se deslocar no tempo do filme a ser reproduzido naquele momento mais rapidamente para trás

Tabela 17 – Caso de uso Forward

Nome	Mover Slider
Descrição	Caso de uso que representa o utilizador poder se deslocar no filme arrastando o *slider* para uma posição que pensa ser a mais correcta para o que pretende visualizar, mover o *slider* afecta o tempo do vídeo que se está a visualizar tanto para a frente ou para trás, será o modo mais rápido de se deslocar no filme.

Tabela 18 – Caso de Uso Mover Slider

Nome	Controlar Som
Descrição	Caso de Uso que representa o que o utilizador poderá fazer com o som que está a ser tocado, enquanto o filme é reproduzido, poderá aumentar ou diminuir o som ou corta-lo

Tabela 19 – Caso de Uso Controlar Som

4.4 Diagrama de Componentes

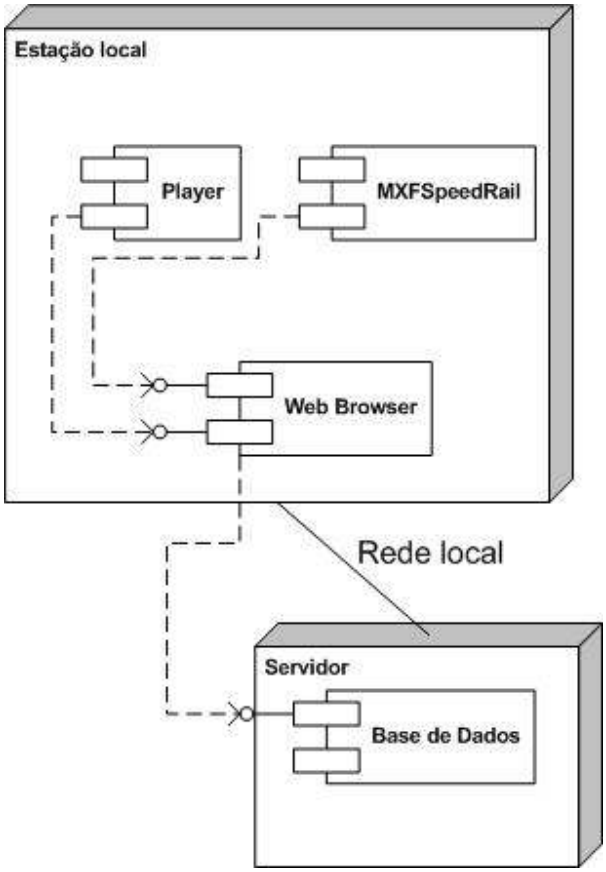


Figura 6 - Diagrama de Componentes

No diagrama apresentado na figura 6 pretende-se apresentar os componentes essenciais ao funcionamento do projecto, dos quais se algum estiver em falta não funcionará. Como é possível observar os principais componentes não são muitos, mas basta que alguma coisa falhe em qualquer dos componentes para que exista algum tipo de problema no uso do *player* ou mesmo da aplicação no global.

4.5 Diagrama de Actividades

Neste diagrama de actividades pretende-se representar os esquemas de operações possíveis de se efectuar quando se trabalha com o *player* de vídeo na aplicação, que corre no *web browser*.

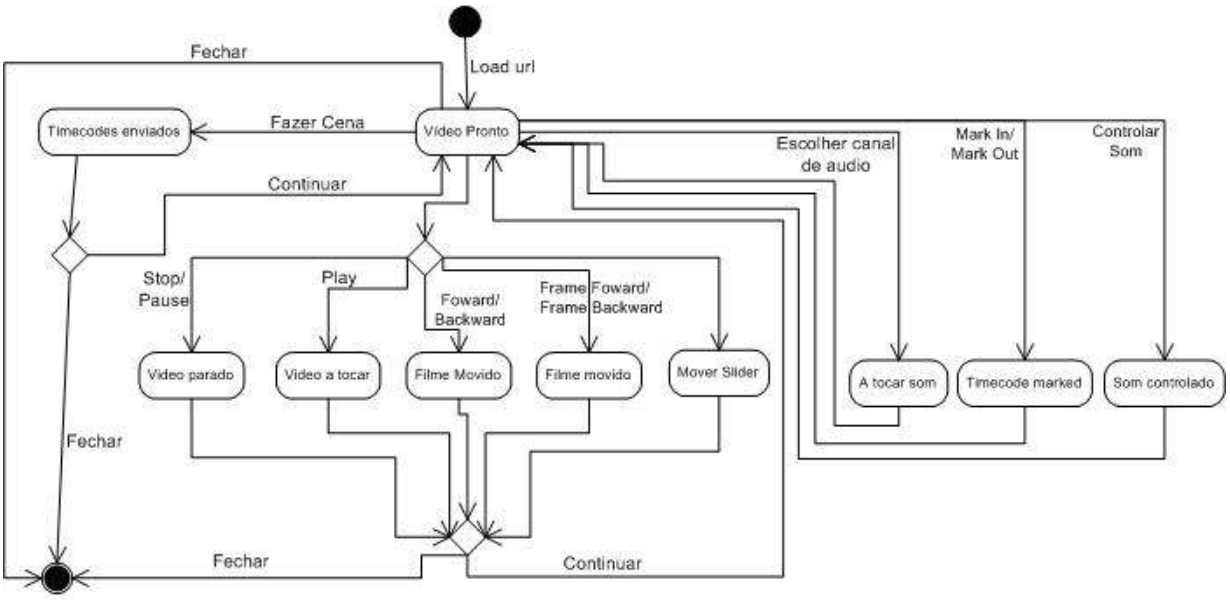


Figura 7 - Diagrama de Actividades

O início das actividades no *player* são despoletadas pelo evento *Load Url*, este é causado pelo actor *Logger* mais acima representado na figura 5 – Diagrama de Casos de uso, através da escolha de um filme para ser visualizado ou editado. Assim que o vídeo está pronto a ser tocado ou parte dele, passa para o estado vídeo pronto. Em Seguida poderá fazer-se qualquer tipo de operação que o *player* permita no vídeo, qualquer uma das escolhas de *Stop*, *Play*, *Pause*, *Forward*, *Backward*, *Frame Forward* ou *Frame Backward*, não podem acontecer simultaneamente.

O envio dos *timecodes* para a plataforma *MISS* pode ser feito logo a seguir ao vídeo carregar pois os tempos inicial e final quando não é feito qualquer *Mark in* ou *Mark Out* serão os tempos iniciais ou finais do vídeo.

As operações de escolher os canais de áudio que se pretende que estejam activos, assim como marcar os *timecodes* para *mark in* ou *mark out*, ou controlar o volume de som, podem ser

feitas simultaneamente, às outras operações anteriormente, pois são acções que são instantâneas e não com intervalos de tempo que podem ser variados.

4.6 Diagrama de Sequencia

Os dois diagramas de sequência apresentados na figura 8, representam a abertura de um novo vídeo que esteja no sistema, e após terem sido efectuadas as operações necessárias, é feita uma cena do vídeo e os dados transmitidos para o *MXFSpeedRail*, e em seguida o vídeo é fechado, nesta sequência de operações, o número de operações e de cenas que o actor pode realizar pode variar conforme o que achar necessário, pois só quando pretende fechar o vídeo é que o faz.

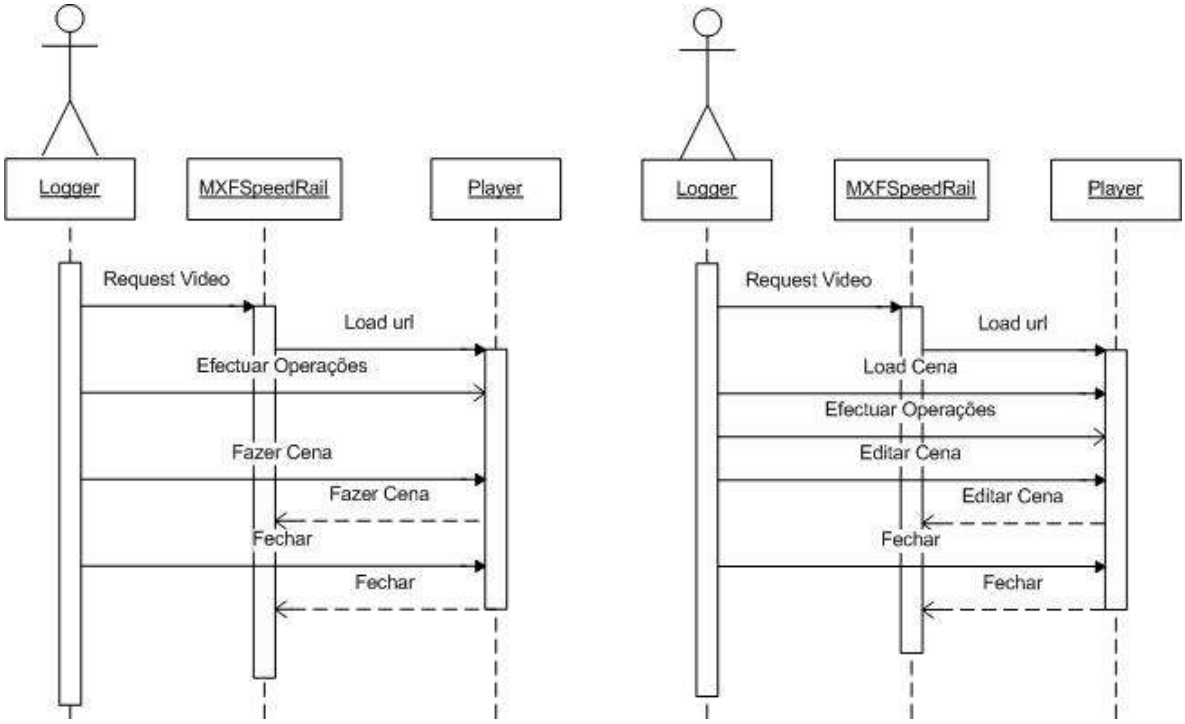


Figura 8 - Diagrama de Sequência

No segundo caso é representada a edição[11] de uma cena de um vídeo a ser aberta para visualização. Em primeiro lugar, o passo é igual ao primeiro caso, este passo serve para o serviço passar o *url* do vídeo ao *player* e este abrir o vídeo para visualização. Em seguida é que se faz o *load* da cena que o utilizador pretender, é enviado para o *player* o timecode inicial da cena que se

[11] Por se tratar apenas de pre-edição, neste caso entende-se por edição de uma cena apenas a alteração do tempo inicial e/ou final desta.

pretende ver, e o *player* deverá começar a dar o filme no tempo recebido, e não no início se o tempo da cena não for o tempo inicial do filme. Em seguida poderá efectuar as operações que achar necessárias sobre o vídeo, e em seguida edita a cena do vídeo, depois poderá fechar o vídeo se assim o entender e achar necessário.

Capítulo 5

Implementação

5.1 Introdução

Neste capítulo serão apresentadas duas soluções que foram rejeitadas mais a solução implementada como solução para o projecto, as soluções rejeitadas são duas soluções desenvolvidas, uma em *Flash* / *Actionscript* e outra usando o *Quicktime*, que como já explicado anteriormente foram rejeitadas, em seguida iremos expor alguns dos detalhes de implementação da solução adoptada e tiradas algumas conclusões, por fim no capítulo seguinte serão dadas algumas perspectivas de trabalho futuro para melhorias que se possam fazer para que o *player* possa ser usado não só no *MXFSpeedRail* mas também noutros produtos que se possam vir a desenvolver, ou num portal para *web*, para qualquer tipo de utilizador.

5.2 Implementação do Flash e Actionscript

A implementação de um *player* em *Flash* e *actionscript* foi a primeira opção onde se realizaram diversos testes para que fosse possível saber se se poderia usar esta tecnologia para o desenvolvimento do *player* ou não, nesta secção não iremos discutir as razões para o facto de esta solução ter sido abandonada, mas sim mostrar alguns detalhes de implementação que poderiam tornar o desenvolvimento de todo o projecto bastante mais simples se esta solução fosse viável.

Mais abaixo na Figura 9 é apresentada uma imagem de um *player* desenvolvido em flash capaz de tocar ficheiros *Mpeg4*, capaz de mover o filme para a frente e para trás livremente arrastando o *slider* correspondente ao tempo do filme, o qual está representado do lado direito da barra, este tempo é representado em segundos.

Para saber se esta seria uma solução viável foi necessário aprofundar uma pouco mais, então foi criado a botão, que se pode encontrar na Figura 9 – *Player* de *Flash* desenvolvido para testes, com o aspecto de duas setas juntas, este botão tem como função tentar mover o filme para a frente a duração de um segundo, para um primeiro teste, mas logo que este foi experimentado

chegou-se a conclusão que não se poderia usar o flash para a solução que se pretende para o projecto, isto porque, sempre que se avança um segundo no botão com o formato das duas setas, o número ao seu lado corresponde aos segundos para onde o vídeo deveria ir, o número que indica o tempo em que o filme se encontra, o que se encontra ao lado da barra do *slider*, não correspondiam muitas das vezes, pois como no flash só se consegue navegar no filme de *keyframe* em *keyframe*, quando o tempo indicado não é uma *keyframe*, o *flash* passa o vídeo para a *keyframe* mais próxima do tempo pretendido pelo utilizador.

A caixa com o texto "Go To:" ao seu lado serviu também para fazer o teste dos tempos, mas onde se encontra o número trinta e nove, poderá ser inserido um número através do teclado em segundos para navegar no filme até a esse tempo, mas mais uma vez se o tempo de destino não for uma *keyframe*, o *flash* reproduz o vídeo na *keyframe* mais próxima.

Todos os testes destes *players* de flash foram desenvolvidos no *Adobe Flash*[12] e integrado numa página *web* para que pode-se ser testado em ambiente *web*. Todo o código desta aplicação está presente no Anexo A.

Figura 9 – Player de Flash desenvolvido para testes

[12] Adobe Flash CS3 Professional - http://www.adobe.com/products/flash/?promoid=BPDEE

5.2.1 Ambiente de Desenvolvimento

O software usado para o desenvolvimento deste teste foi a versão *trial* do *Adobe Flash CS3 Professional* e um editor de *HTML*, o *notepad*++. Para desenvolver o código *Flash*/*Actionscript* foi usada a versão *trial* do *Adobe Flash CS3 Professional* devido a este ser um dos melhores ambientes de desenvolvimento para o efeito, foi usada a versão *trial* devido a se tratar de um programa de software pago. O *notepad*++ foi usado por ser um programa *open source* ao abrigo da licença *GPL* e de uso livre, com bastantes capacidades para o efeito pretendido, desenvolver código *HTML*.

5.3 Implementação em Quicktime

A solução *quicktime* foi desenvolvida posteriormente á solução de *flash*, e algum tempo depois de se ter começado o desenvolvimento dos controlos *ActiveX*, mesmo assim esta hipótese foi estudada, pois após alguma pesquisa, foi encontrada documentação de como se usar as bibliotecas *javascript* que são disponibilizadas[13] gratuitamente para trabalhar com a versão grátis do *quicktime*.

O *javascript* pode interagir com o *quicktime* de várias maneiras. Pode-se usar *javascript* para o browser detectar se o *quicktime* está instalado ou não no computador, para criar *tags* do objecto *quicktime* com parâmetros para o browser ler, e para controlar o *plug-in* do *quicktime* directamente, sendo esta última a ser usada.

Para a realização dos testes para analisar se a opção *Quicktime* seria viável foi implementada uma página *web* com um objecto *Quicktime* embebido na página, e funções *Javascript* para utilizar os eventos *DOM* usados pelo *Quicktime*, e controlar o objecto. Foi uma implementação simples, para que se pode-se avaliar se o *quicktime* seria capaz de cumprir todos os requisitos, que seriam necessários para o sucesso deste projecto.

Na figura 10 é apresentada uma imagem da página *web* que foi desenvolvida, pode-se observar que o *player* que está a tocar o vídeo é o *quicktime* e tudo o que está para baixo como os botões e o texto foi desenvolvido em *html* com funções *javascript* associadas, para se poder testar o *player* para que fosse possível averiguar se esta solução seria possível de ser utilizada.

Os quatro botões que se encontram imediatamente abaixo do *player* de vídeo, usam funções do *javascript* para interagir directamente com o *player*. Para ser possível andar uma *frame* para a frente ou para trás com os métodos disponibilizados na documentação do *quicktime* usa-se a função *step* (*int i*), que aceita um inteiro negativo ou positivo, andando para a frente ou para trás da *frame* em que se encontra o i *frames*. Caso o número i seja positivo vai para a frente em relação à *frame* a que se encontra caso seja negativo vai para trás em relação à *frame* em que se

[13] Ver Referências – Documentação Quicktime

encontra. No caso da página desenvolvida o número de *frames* que deveria andar para trás ou para a frente seria só de uma *frame* de cada vez.

As quatro caixas de texto mais abaixo representa o *timecode* do filme que está a ser reproduzido, o cálculo deste *timecode* é baseado na *frame rate* do vídeo que é calculada através do número de unidades de tempo que um tem segundo, e quantas unidades de tempo tem uma *frame*, dividindo estes dois valores temos a *frame rate* do filme que está a ser reproduzido. O cálculo da *frame* rate, dos *timecodes* completos serão expostos no Anexo B, assim como todo o código desta implementação.

Todos os testes tiverem um bom resultado pois esta opção passou em todos os requisitos, com a excepção de o *plug-in* do *quicktime* embebido não ser capaz de reproduzir ficheiros do tipo MXF ou MPEG4.

Figura 10 - Quicktime player embebido para testes

5.3.1 Ambiente de Desenvolvimento

O Software utilizado para o desenvolvimento deste teste foi um editor de *Html* o *notepad++*.

5.4 Implementação dos Controlos ActiveX

A implementação dos controlos *ActiveX* teve quatro fases, pois sendo uma tecnologia nova, a primeira fase foi dedicada ao estudo e aprendizagem de como utilizar e manipular os controlos *activex*, para tal foram realizados vários *tutoriais* encontrados, e consultada a ajuda *online* da Microsoft disponível no MSDN[14].

O desenvolvimento dos controlos *ActiveX* tiveram como base um *player* de vídeo já desenvolvido pela empresa, então inicialmente, teve de ser feito um estudo do código já feito para que fosse possível fazer uma implementação sem redundâncias ou erros facilmente evitáveis. Na figura 11 é apresentado o *player* desenvolvido pela empresa que foi adaptado para *web* com os controlos *ActiveX*.

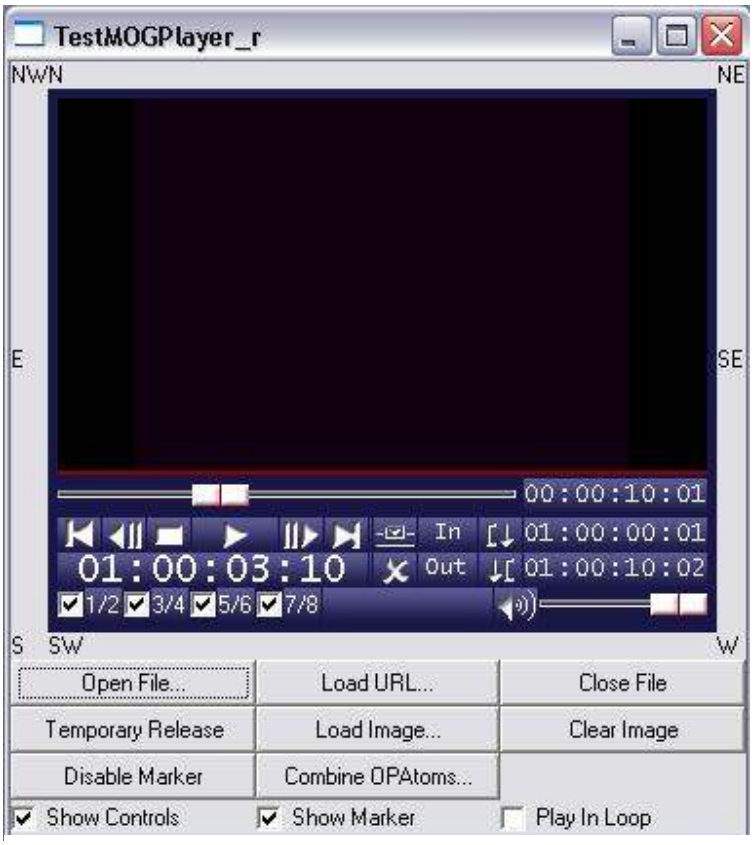


Figura 11 – Player de Vídeo Stand-Alone

A segunda fase de desenvolvimento dos controlos *ActiveX* foi a implementação de uma classe, tendo como classe pai, a classe que representa o *player* de vídeo, versão *stand alone*, já desenvolvido pela empresa. A classe desenvolvida implementa funções que simplesmente passam os parâmetros para a classe pai, para que o *player* funcione correctamente, esses parâmetros que são passados são por exemplo, o *Url* onde o vídeo de encontra, ou o tipo de ficheiro de vídeo que o *player* irá tocar, entre outros.

Outras funções tiveram de ser implementadas de modo diferente, pois lêem os valores que estão na interface, que podem ser introduzidos manualmente pelo utilizador, e em seguida podem ser enviados para o *Javascript* para se obter mais informações a cerca do vídeo, ou dos tempos.

Foram criadas algumas funções com o propósito de apenas comunicarem com o *Javascript*, de modo a que a interacção entre o *player* de vídeo desenvolvido com os controlos *activex*, e o *web browser* e mais tarde o *flash* fosse possível e bastante simplificada.

Após os controlos *ActiveX* terem sido implementados, e compilados com sucesso foram criados diversos testes para uma análise de pequenos erros de implementação, testes que começaram de uma forma bastante simples, inicialmente foi feita uma página *html* muito básica que apenas continha os parâmetros para embeber o protótipo na página, e duas funções *Javascript*, que pudessem comunicar com o protótipo, de forma a que este pudesse abrir um vídeo guardado localmente no computador e fechar o mesmo.

Para que seja possível embeber um objecto *ActiveX* numa página *html* é preciso usar o seguinte código.

```
<OBJECT ID="MOGPlayerAx" WIDTH="390" HEIGHT="360"

CLASSID="CLSID:88D58774-64FA-4C50-9972-439BE5681C40">

</OBJECT>
```

Onde o *CLASSID* é uma variável definida no código da classe que gera o objecto *ActiveX*, todas as outras variáveis podem ser modificadas conforme as necessidades de quem o implementar. Mas para que seja possível o browser reconhecer o que o *CLASSID* representa o ficheiro *.dll* gerado depois de a classe que representa o objecto *ActiveX* ter sido compilada, existe um passo intermédio, este é, utilizar o serviço de registo do *Windows* para registar a *dll* no sistema, isto faz-se introduzindo a seguinte linha na linha de comandos.

```
regsvr32 ./mogplayerax.dll
```

Após estes testes terem sido completados com sucesso, foi necessário testar como seria feita a comunicação entre o flash embebido na mesma página *web* que o objecto *activex*, o *player* de vídeo, para tal foram criados em *flash* quatro botões e duas caixas de texto, de forma a estes

[14] Microsoft Developer Network

chamarem as funções de *javascript* que comunicam com o objecto *activex*. A figura 12 representa a interface de essa implementação que foi desenvolvida.

Figura 12 – Interface do teste de comunicação entre o Flash e o Objecto Activex

Para o sucesso deste teste foi necessário usar a função *call* do *Actionscript*, da classe *ExternalInterface*, para que os valores introduzidos pelo utilizador no flash possam ser passados para as funções *Javascript*, que enviam para o objecto *ActiveX* esses valores para que o filme possa ser carregado, este teste foi fundamental no desenvolvimento do trabalho pois como este protótipo foi integrado numa aplicação a ser desenvolvida pela empresa no decorrer do projecto, e essa aplicação estar completamente a ser desenvolvida no *Adobe Flex Builder*, o que produz uma aplicação *flash* para *web*. Logo a comunicação do *player* com *flash* é fundamental para o bom funcionamento do projecto, pois todas as cenas marcadas no *player* para em seguida serem passadas para a aplicação e guardadas na base de dados, terão de recorrer ao método acima indicado.

Em seguida, a integração do *player* de vídeo na aplicação *MXFSpeedRail*, após todos os testes anteriores apresentados terem sido executados com sucesso, foi facilmente desenvolvida, e a interface acondicionada as necessidades da integração, isto é, o *player* foi integrado na página *Html* que também integra a aplicação *MXFSpeedRail*, e foram adicionadas algumas propriedades, foi adicionado uma imagem de fundo ao *player*, para que este ficasse o mais parecido com a aplicação, passou também a ser possível mover o *player* dentro da página *web* de modo a que não ficasse sobreposto a nenhuma outra interface, e de modo a que o utilizador possa posicionar todos os conteúdos que pretende visualizar da forma que lhe mais convém.

O resultado final do Projecto é apresentado na Figura 13.

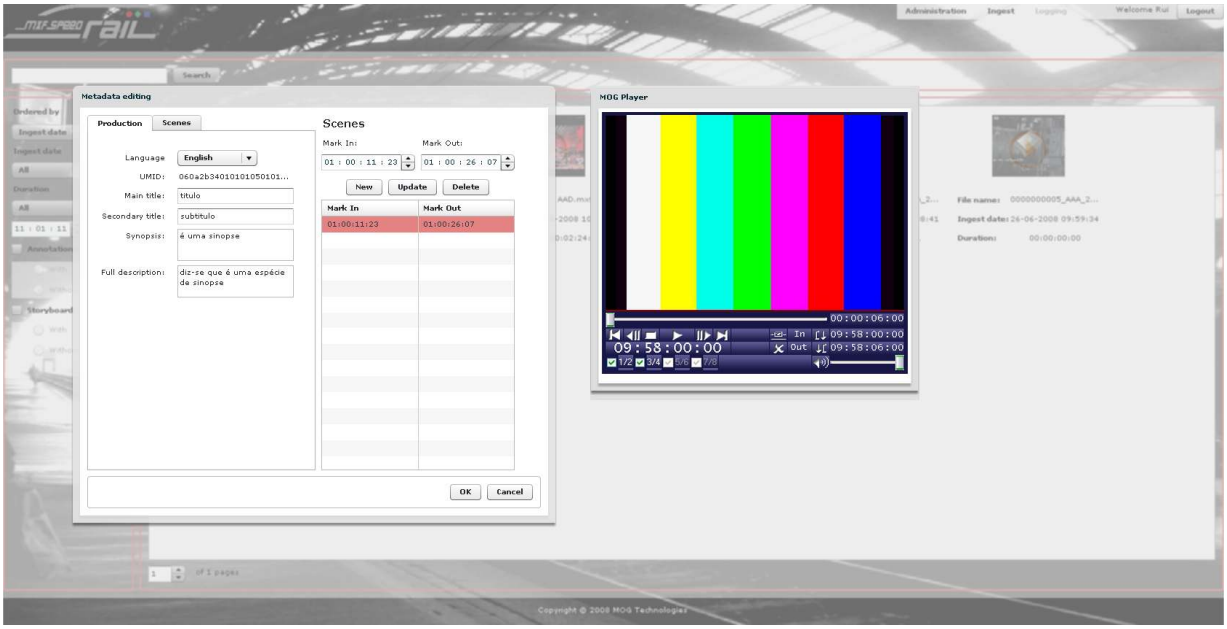


Figura 13 - Interface do Protótipo

5.4.1 Ambiente de Desenvolvimento

Para a realização dos controlos *ActiveX* foi usado o *Visual Studio .Net*, com diversas bibliotecas extra instaladas como por exemplo, o *Qt* 3.3.8, *Xerces* 2.7.0, *Xalan* 1.10.0, *DX90SDK*. Estas tecnologias tiveram de ser usadas devido ao *SDK* já desenvolvido por parte da empresa usar estas bibliotecas e os projectos gravados foram todos desenvolvidos no *Visual Studio .Net*.

Para o desenvolvimento da interface com o utilizador foi usado um editor de *Html*, o *notepad*++ para embeber o *ActiveX* na página *web*, com a aplicação *MXFSpeedRail* desenvolvida no *Flex Builder* da *Adobe*.

Capítulo 6

Conclusões e trabalho futuro

A realização deste trabalho foi um desafio, desafio esse bastante interessante, motivador e enriquecedor, a área de trabalho deste projecto sendo multimédia trouxe valor acrescentado para a motivação na sua realização, e a possibilidade de este ser integrado numa aplicação maior comercializada pela empresa.

6.1 Satisfação dos Objectivos

Os principais objectivos delineados para a execução do projecto foram todos cumpridos e todos os requisitos estabelecidos foram cumpridos durante toda a execução do protótipo.

A reprodução dos vídeos, marcação dos tempos, comunicação com o MXFSpeedRail, eram os objectivos principais deste projecto, todos eles foram realizados com sucesso após variados testes.

6.2 Trabalho Futuro

As perspectivas de trabalho futuro no projecto são as de acrescentar mais valias ao trabalho já desenvolvido, mais valias essas que tornarão o produto único no mercado no nosso presente, isto é, a utilização do *player* para a realização de um site ao estilo do *youtube*, mas com detecção de conteúdos, isto é, detectar se determinado filme detêm direitos de autor ou propriedade, esta detecção pode ser levada a cabo consultando os metadados presentes no ficheiro do vídeo, assim não permitindo utilizadores não autorizados à visualização de determinados vídeos, ou incluindo uma imagem ao estilo de *watermark* na reprodução do vídeo para que os utilizadores tenham conhecimento a quem o vídeo realmente pertence. Ou mesmo não permitindo a disponibilização desses conteúdos de forma a evitar qualquer tipo de pirataria, o *player* nesta situação tem um

papel fundamental pois é a última barreira que poderá impedir que os vídeos sejam disponibilizados.

Outra implementação extra que poderá ser efectuada será a possibilidade de o *player* conseguir abrir múltiplos ficheiros, pois muitas vezes os ficheiros de áudio e vídeo são separados, podendo até serem vários ficheiros de áudio com um máximo de quatro ficheiros.

Referências

[App08] Apple. JavaScript Scripting Guide for Quicktime (Online) 2008
http://developer.apple.com/documentation/quicktime/Conceptual/QTScripting_JavaScript/QTScripting_JavaScript.pdf

[CRa97] CRaG Systems. The Unified Modelling Language - Part 2 (Online) 1997
www.cragsystems.co.uk/ITMUML/the03/00the03.htm

[SGI05] SGI, Inc. SGI Media Server for Broadcast User's Guide (Online) 2005
http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi?coll=0650&db=bks&fname=/SGI_EndUser/MSB3xx_UG/go01.html

[Hig02] HighDef. The HighDef Glossary of Terms (Online) 2002.
www.highdef.com/library/glossary.htm

[Ado07]Adobe Systems, Inc.. Flash video learning guide (Online) 2007
http://www.adobe.com/devnet/flash/articles/video_guide.html

[Haa07]Peter deHaan. Flash Quick Starts: Building Flash video projects (Online) 2007
http://www.adobe.com/devnet/flash/quickstart/metadata_cue_points/

[Cdy02]Community MX, Derrick Ypenburg. Building a Custom Flash Video Player Part 1: The Basic Application (Online) 2002
http://www.communitymx.com/content/article.cfm?page=1&cid=4E1D8

[Mic08] Microsoft. ActiveX Controls Overviews and Tutorials (Online) 2008
http://msdn.microsoft.com/en-us/library/aa751969(VS.85).aspx

[AKR02] Arun K. Ranganathan , Netscape Communications. Scripting The Flash Player Plugin (Online) 2002 http://devedge-temp.mozilla.org/viewsource/2002/scripting-flash/

[Haa08] Peter deHaan. Calling ActionScript functions from your HTML templates using the ExternalInterface API (Online) 2008 http://blog.flexexamples.com/2008/03/11/calling-actionscript-functions-from-your-html-templates-using-the-externalinterface-api/#more-556

[Mic08] Microsoft. Creating Signed CAB Files for MFC and ATL Controls (Online) 2008: http://msdn.microsoft.com/en-us/library/4kex18w6(VS.80).aspx

Anexo A

Código Actionscript

```
var theVideo:Video = new Video(300,225);
stage.addChild(theVideo);
theVideo.x = 126;
theVideo.y = 10;
var k=0;
var timePlayed:Number;

var nc:NetConnection = new NetConnection();
nc.connect(null);
var ns:NetStream = new NetStream(nc);

var video:Object = new Object();
ns.client = video;

ns.addEventListener(AsyncErrorEvent.ASYNC_ERROR, asyncErrorHandler);
function asyncErrorHandler(event:AsyncErrorEvent):void {
}
```

```
theVideo.attachNetStream(ns);
ns.play("//pc-jrodrigues/Shared/teste/backcountry_bombshells_4min_HD_H264.mp4");
addChild(theVideo);
var videoInterval = setInterval(videoStatus, 0.001);
var amountLoaded:Number;
var duration:Number;

video.onMetaData = metaDataHandler;
function metaDataHandler(obj :Object):void {
        duration = obj.duration;
}
function videoStatus(){
        amountLoaded = ns.bytesLoaded / ns.bytesTotal;
        loader.loadbar.width = amountLoaded * 352;
        loader.scrub.x = ns.time / duration * 352;
        timePlayed = Math.round(ns.time);
        myLabel.text = timePlayed.toString();
}
var scrubInterval;
loader.scrub.addEventListener(MouseEvent.MOUSE_DOWN, scrubPress);
loader.scrub.addEventListener(MouseEvent.MOUSE_UP, scrubMove);
function scrubPress(evt:MouseEvent) {
        clearInterval(videoInterval);
        scrubInterval = setInterval(scrubit,0.001);
        var range:Rectangle = new Rectangle(0,this.y-12,352,this.y);
        loader.scrub.startDrag(false,range);
}
function scrubMove(evt:MouseEvent) {
        clearInterval(scrubInterval);
        videoInterval = setInterval(videoStatus, 0.001);
        this.stopDrag();
```

```
}
function scrubit() {
	ns.seek(Math.floor((loader.scrub.x/389.9) * duration));
}

playBtn.addEventListener(MouseEvent.CLICK, playfunc);
stopBtn.addEventListener(MouseEvent.CLICK, stopfunc);
function playfunc(evt:MouseEvent){
	ns.resume();
}
function stopfunc(evt:MouseEvent){
	ns.pause();
}
iText.addEventListener(Event.CHANGE, doTextInput);
var tempo:Number;
function doTextInput(e:Event) {
		tempo = e.target.text;
}
goBtn.addEventListener(MouseEvent.CLICK, gototime);
function gototime(evt:MouseEvent){
	ns.seek(tempo);
	k=tempo;
	trace(k);
}
fwdBtn.addEventListener(MouseEvent.CLICK, fwdFilme);
function fwdFilme(evt:MouseEvent){
	ns.seek(k);
	kValue.text = k;
	k = k+1;
}
```

Código Html

```
<object width="1024" height="768">
<param name="movie" value=".\teste.swf">
<embed src=".\teste.swf" width="1024" height="768">
</embed>
</object>
```

Anexo B

```
<head>
<script language="JavaScript">
frameRate = 0;
intervalo = 0;
frame = 0;
horas=minutos=segundos=frame=0;

function openUrl(){
	try{
		alert(document.movie1.GetUrl());
		url = document.ficheiro.url.value.toString();
		alert(url);
		document.getElementById("movie1").SetUrl(url);
		}catch(e){
			alert(e.description);
			}
}
function calcFrameRate() {
	document.movie1.rewind();
	document.movie1.Step(1);
	ts = document.movie1.getTimeScale();
	time = document.movie1.getTime();
	alert(document.movie1.getTimeScale());
	alert(document.movie1.getTime());
	frameRate = ts/time;
```

```
	document.movie1.Step(-1);
}

function timecode(){
	timeScale = document.movie1.getTimeScale();
	time = document.movie1.getTime();
	duracao = document.movie1.getEndTime();
	intervalo = Math.floor(100 / frameRate);
	v = (Math.floor(time / (timeScale *60*60)) % 60) + ":" + (Math.floor(time / (timeScale *60)) % 60) + ":" +
(Math.floor(time / timeScale) % 60) + ":" + (Math.floor((Math.floor((time*100) / timeScale) % 100) / intervalo));

	duracaoFilme = (Math.floor(duracao / (timeScale *60*60)) % 60) + ":" + (Math.floor(duracao / (timeScale
*60)) % 60) + ":" + (Math.floor(duracao / timeScale) % 60) + ":" + (Math.floor((Math.floor((duracao*100) /
timeScale) % 100) / intervalo));
	document.getElementById("time1").innerHTML=v;
	document.getElementById("time2").innerHTML=document.movie1.getTimeScale().toString();
	document.getElementById("time3").innerHTML=document.movie1.getDuration().toString();
	document.getElementById("time4").innerHTML=duracaoFilme;
	}
function updateFields(){
	horasTemp = (Math.floor(time / (ts*60*60)) % 60);
	minutosTemp = (Math.floor(time / (ts*60)) % 60);
	segundosTemp = (Math.floor(time / ts) % 60);
	framesTemp = (Math.floor((Math.floor((time*100) / ts) % 100) / intervalo));
	document.tempo.Horas.value=horasTemp;
	document.tempo.Minutos.value=minutosTemp;
	document.tempo.Segundos.value=segundosTemp;
	document.tempo.Frame.value=framesTemp;
}
```

```
function goToTime(){
        horas = document.tempo.Horas.value;
        minutos = document.tempo.Minutos.value;
        segundos = document.tempo.Segundos.value;
        frame = document.tempo.Frame.value;
        tempHoras = horas * 60 * 60 * 600;
        tempMins = minutos * 600 * 60;
        tempSegundos = segundos*600;
        tempFrames =  ((1/frameRate)*600) * frame;
        tempAux = tempHoras + tempMins + tempSegundos + tempFrames;
        //document.getElementById("time6").innerHTML=tempAux;
        document.movie1.setTime(tempAux);
        document.movie1.Stop();
}

/* define function that shows percentage of movie loaded */
function showProgress()
{
var percentLoaded = 0 ;
percentLoaded = parseInt((document.movie1.GetMaxTimeLoaded() / document.movie1.GetDuration()) * 100);
document.getElementById("loadStatus").innerHTML = 'Movie loading: ' + percentLoaded + '% complete...';
}

/* define function that executes when movie loading is complete */
function movieLoaded()
{
        alert("movie loaded");
        calcFrameRate();
        RegisterListeners();
        document.getElementById("loadStatus").innerHTML = "Movie Loaded" ;
}
```

```
function movieCanPlay(){
        alert('movie can play');
}

/* define function that adds another function as a DOM event listener */
 function myAddListener(obj, evt, handler, captures)
{
if ( document.addEventListener )
        obj.addEventListener(evt, handler, captures);
else
        // IE
        obj.attachEvent('on' + evt, handler);
}

 /* define functions that register each listener */
function RegisterListener(eventName, objID, embedID, listenerFcn)
{
var obj = document.getElementById(objID);
if ( !obj )
        obj = document.getElementById(embedID);
if ( obj )
        myAddListener(obj, eventName, listenerFcn, false);
}

/* define a single function that registers all listeners to call onload */
function RegisterListeners()
{
        RegisterListener('qt_progress', 'movie1', 'qtmovie_embed', showProgress);
        RegisterListener('qt_load', 'movie1', 'qtmovie_embed', movieLoaded);
        RegisterListener('qt_canplay', 'movie1', 'qtmovie_embed', movieCanPlay);
}
```

```
setInterval(timecode,60);

</script>

</head>

<body>

<object id="qt_event_source" classid="clsid:CB927D12-4FF7-4a9e-A169-56E4B8A75598"
codebase="http://www.apple.com/qtactivex/qtplugin.cab#version=7,2,1,0" ></object>

<OBJECT classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
codebase="http://www.apple.com/qtactivex/qtplugin.cab" width="320" height="260" id="movie1"
style="behavior:url(#qt_event_source);">

    <PARAM name="src" value="test01.mov">

            <PARAM name="controller" value="true">

            <PARAM name="autostart" value="false">

            <PARAM name="postdomevents" value="true">

        <EMBED width="500" height="400" src="test01.mov" autostart="false" name="movie1"
id="movie_embed1" controller="true" postdomevents="true"> </EMBED>

</OBJECT> <br>

<input name="prevFrame" type="button" value="Previous Frame" onclick="javascript:document.movie1.Step(-1);
timecode();updateFields();">

<input name="play" type="button" value="play" onclick="javascript:document.movie1.Play();updateFields();" >

<input name="pause" type="button" value="pause" onclick="javascript:document.movie1.Stop(); updateFields();">

<input name="nextFrame" type="button" value="Next Frame" onclick="javascript:document.movie1.Step(1);
timecode(); updateFields();">

<form name="tempo">

        <INPUT TYPE=TEXT NAME="Horas" maxlength="2" size="1" value="00">:<INPUT TYPE=TEXT
NAME="Minutos" maxlength="2" size="1" value="00">:<INPUT TYPE=TEXT NAME="Segundos"
maxlength="2" size="1" value="00">:<INPUT TYPE=TEXT NAME="Frame" maxlength="2" size="1"
value="00">

</form>

<input name="prevFrame" type="button" value="Calc Frame Rate" onclick="javascript:calcFrameRate();"><br>

<input name="goToTime" type="button" value="Go To Time" onclick="javascript:goToTime();"><br>


<span id="time1"> </span> - Tempo<br>

<span id="time2"> </span> - Unidades de Tempo<br>
```

  - Duração do Filme em unidades de Tempo

  - Durações do Filme com frames

</body>

Anexo C

1- Frame Backward
2- Backward movie
3- Stop
4- Play/Pause
5- Fast Forward movie
6- Frame Forward
7- Slider
8- Duration
9- Mark In Timecode
10- Mark Out Timecode
11- Volume Controler

12- Mute / Unmute sound

13- Mark Out Current Timecode

14- Mark In Current Timecode

15- Make Scene with mark in and out Timecodes

16- Clear mark in and out Timecodes

17- Audio Channels

18- Current Time of the movie

19- Movie Area