# FERRAMENTAS PARA APOIO AO DESENVOLVIMENTO DE APLICAÇÕES PARA O SISTEMA NIVO

Paulo Miguel da Silva Oliveira

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto

2011

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: Paulo Miguel da Silva Oliveira, Nº 1060410, 1060410@isep.ipp.pt
Orientação científica: Paula Maria Marques Moura Gomes Viana, pmv@isep.ipp.pt

Empresa: Nonius Software

Supervisão: Raul Miguel Pinheiro de Carvalho, rmc@noniussoftware.com

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto
27 de Junho de 2011

Para a minha namorada Xana, que comigo partilhou todos os altos e baixos resultantes deste estágio, que em vários momentos consumiu muito de mim.

Para os meus pais, porque sem eles não seria quem sou hoje.

# *Agradecimentos*

Aproveito esta secção para agradecer à Nonius Software pelo estágio que me proporcionou, contribuindo para o meu desenvolvimento quer a nível técnico, quer a nível de relacionamento pessoal. Agradeço ao meu supervisor na empresa, Eng.º Raul Carvalho, pela disponibilidade e ajuda em vários aspectos particulares do meu trabalho.

Agradeço em especial à minha orientadora, Eng.ª Paula Viana, por todo o apoio e orientação prestada. Gostaria também de agradecer-lhe todas as sugestões manifestadas e todo o tempo despendido ao longo da realização da minha tese.

Agradeço também a todos os docentes do Mestrado em Engenharia Electrotécnica e de Computadores – Telecomunicações, pelos conhecimentos transmitidos durante o período lectivo.

Agradeço ainda ao António Mendes e ao André Rodrigues, pela boa disposição e pelo ambiente de trabalho que partilhamos na Nonius Software.

# *Resumo*

Ao longo dos últimos anos tem-se verificado um desenvolvimento significativo em soluções Internet Protocol Television (IPTV), com vista a ser implementadas em diversos sectores, incluindo o mercado hoteleiro. O presente trabalho desenvolveu-se em torno de um produto específico ligado a esta área, o sistema de entretenimento NiVo. Este produto foi criado pela Nonius Software, empresa associada ao ramo das telecomunicações, que continua a expandi-lo, conferindo-lhe novas características e melhoramentos.

O objectivo principal do projecto consiste no desenvolvimento de uma aplicação simuladora dos temas gráficos do sistema NiVo, com a finalidade de facilitar o processo de criação dos mesmos. Esta aplicação pretende ter uma apresentação e interactividade o mais próxima possível do sistema real, permitindo realizar o teste/validação de um tema, sem a necessidade de utilização de equipamento que compõe o sistema NiVo. Um segundo objectivo baseia-se no desenvolvimento de uma aplicação de ajuda à composição de temas, evitando que estes sejam desenvolvidos manualmente, sem qualquer tipo de suporte, o que pode resultar em erros.

Para o cumprimento dos objectivos estabelecidos, focando-se no desenvolvimento do simulador, utilizou-se a tecnologia Flash, sendo esta muito utilizada para a criação de animações interactivas. Desta forma, a aplicação desenvolvida tira partido das vantagens fornecidas por esta tecnologia.

***Palavras-Chave***

IPTV, Nonius, NiVo, Tema, Aplicação, Simulador, Flash, ActionScript.

# *Abstract*

Over the recent years there has been a significant development in Internet Protocol Television (IPTV) solutions in order to be implemented in different areas including the hotel market. Thus, this work relates to a specific product linked to this area, the entertainment system NiVo. This product was created by the Nonius Software, a company in the telecommunications business, which continues to expand it, including new functionalities and improvements.

The main goal of this project is to develop a simulator application of NiVo system graphical themes, in order to facilitate the process of creating them. This application must have a presentation and interactivity as close as possible to the real system, allowing thus, perform the tests and validations of the topic in question, without the need to use the NiVo system equipment. A second objective is based on the development of an application to help in the composition of themes, avoiding them to be developed manually, without any support, which may result in errors.

To fulfill the objectives set, focusing on the development of the simulator, it was used the Flash technology, which is widely used for creating interactive animations. Therefore, the developed application takes advantage of the benefits provided by this technology.



***Keywords***

IPTV, Nonius, NiVo, Theme, Application, Simulator, Flash, ActionScript.

# *Índice*

# *Índice de Figuras*

# *Índice de Tabelas*

# *Acrónimos*

AIR – Adobe Integrated Runtime

AS – ActionScript

AVM – ActionScript Virtual Machine

BD – Blu-ray Disc

CD – Compact Disc

DES – Digital Entertainment Solution

DHCP – Dynamic Host Configuration Protocol

DMA – Distributed Media Architect

DVB – Digital Video Broadcasting

DVD – Digital Versatile Disc

FLV – Flash Video

HD – High-Definition

HTML – HyperText Markup Language

I&D – Investigação e Desenvolvimento

IP – Internet Protocol

IPTV – Internet Protocol Television

IT – Information Technology

MP3 – MPEG-1/2 Audio Layer 3

MPEG – Moving Picture Experts Group

nDVR – Network Digital Video Recorder

nVOD – Near Video On Demand

OO – Orientada a Objectos

PMS – Property Management System

PNG – Portable Network Graphics

STB – Set-Top Box

SWF – Shockwave Flash File

TCP – Transmission Control Protocol

TV – Televisão

URL – Uniform Resource Locator

VOD – Video On Demand

XML – Extensible Markup Language

Wi-Fi – Wireless Fidelity

# 1. INTRODUÇÃO

Neste primeiro capítulo pretende-se fazer uma apresentação do tema do estágio realizado no âmbito da unidade curricular Tese/Dissertação do 2º ano do Mestrado em Engenharia Electrotécnica e de Computadores, área de especialização em Telecomunicações.

Seguidamente, realiza-se uma contextualização do trabalho, apresentam-se os objectivos que se pretendem atingir, bem como a organização do relatório.

## 1.1. LOCAL DO ESTÁGIO

O trabalho apresentado neste relatório foi elaborado nas instalações da Nonius Software, situada na Tecmaia – Parque de Ciência e Tecnologia da Maia, enquadrado no departamento de Investigação e Desenvolvimento (I&D).

A Nonius Software é uma empresa nacional de engenharia, que se dedica ao desenvolvimento de soluções para a gestão de redes de comunicação, tendo iniciado a sua actividade em Janeiro de 2005. A Nonius começou por oferecer soluções de acesso à internet para o mercado hoteleiro português. Com foco nesse mesmo mercado, resultou a criação de novos produtos e serviços. Este princípio levou a Nonius a desenvolver uma solução avançada de *in-room entertainment* baseada em tecnologias Internet Protocol Television (IPTV), o NiVo. Este novo produto possui funcionalidades que permitem oferecer aos hóspedes, por exemplo, funcionalidades de televisão (TV) e rádio, vídeo a pedido (Video On Demand), acesso à Internet e jogos. O controlo do equipamento é

possível através de um comando e de um teclado *wireless*. Existe também a possibilidade de interacção através de IPhone, utilizando-se para isso a aplicação "NiVo Remote" [1].

## 1.2. OBJECTIVOS

Este projecto surgiu com o intuito de facilitar o processo de desenvolvimento de temas gráficos para o sistema NiVo, incidindo principalmente na criação de um simulador. Neste contexto, é de realçar a sua importância para a equipa de desenvolvimento. Permite a simulação do tema em causa, verificando-se assim a conformidade ou não com o desejado, antes de o implementar na realidade. Assim, realiza-se o teste/validação da aplicação, sem a necessidade de utilização de equipamento que compõe o sistema NiVo. Além do mais, outra finalidade importante do simulador do NiVo passa pela possibilidade do seu envio para o cliente final, tornando-se mais fácil a percepção do estado do sistema desejado. Desta forma, possibilita um melhor e mais objectivo *feedback* da sua satisfação.

Existe ainda um segundo objectivo que se baseia no desenvolvimento de uma aplicação de ajuda à composição de temas, evitando que estes sejam desenvolvidos manualmente, sem qualquer tipo de suporte, o que pode resultar em erros.

As principais contribuições deste trabalho foram:

1. Aperfeiçoamento do sistema NiVo;
2. Optimização do processo de criação/alteração de temas;
3. Criação de um simulador do sistema NiVo;
4. Possibilidade de utilização comercial imediata desse simulador;
5. Possibilidade da sua utilização pela equipa de desenvolvimento, facilitando a verificação do aspecto do tema e a sua interactividade.

## 1.3. ORGANIZAÇÃO DO RELATÓRIO

Este relatório encontra-se dividido em seis capítulos.

No primeiro capítulo, faz-se uma breve introdução, onde se contextualiza o trabalho realizado e onde são colocados em destaque os objectivos e contribuições mais importantes deste projecto.

No segundo capítulo, realiza-se um levantamento de soluções IPTV no mercado hoteleiro, conferindo-se maior profundidade ao produto tratado neste trabalho, o NiVo.

No terceiro capítulo, aborda-se a tecnologia utilizada no projecto, incluindo-se uma descrição de vários aspectos específicos estudados sobre a linguagem de programação que lhe está associada. Além disso, faz-se uma breve introdução à ferramenta de desenvolvimento utilizada.

No quarto capítulo, apresenta-se o simulador desenvolvido para o sistema NiVo, descrevendo-se a estrutura da aplicação e o seu princípio de funcionamento, sendo fornecido um exemplo de simulação.

No quinto capítulo, apresentam-se os desenvolvimentos efectuados numa segunda aplicação, a qual tem como objectivo a composição de temas.

Por fim, no sexto capítulo, finaliza-se o relatório com as respectivas conclusões e perspectivas do trabalho futuro para optimização deste projecto.

# 2. SOLUÇÕES IPTV NO MERCADO HOTELEIRO

Neste capítulo, faz-se um levantamento de algumas soluções IPTV desenvolvidas com vista ao mercado hoteleiro. Referem-se alguns aspectos comparativos em termos de tecnologia utilizada, bem como no que diz respeito a funcionalidades disponibilizadas.

Em primeiro lugar, é realizada a abordagem a três diferentes produtos associados à área em causa, sendo que por último se procede a uma descrição mais detalhada do produto tratado neste trabalho, o NiVo.

## 2.1. EONA DIGITAL ENTERTAINMENT SOLUTION (DES)

A EONA, empresa sediada em França, enquadrada na área de IPTV e Video On Demand (VOD), desenvolveu o DES, produto que fornece vários serviços de informação e entretenimento na televisão sobre Internet Protocol (IP). Alguns destes serviços são por exemplo, disponibilização de canais de TV e de rádio, VOD, internet na TV, e também serviços de manutenção do quarto (ocupação, limpeza). Além disso, possui uma interface multilingue.

Um outro aspecto deste produto associa-se ao facto de ser projectado para ter uma interface gráfica que pode ser personalizada, fornecendo aos clientes o aspecto final desejado [7][8].

No que diz respeito à arquitectura do DES, a Figura 1 apresenta as comunicações entre os vários componentes da sua solução digital, como por exemplo, a recepção IPTV, o servidor ou mesmo as Set-Top Box (STB) [7].

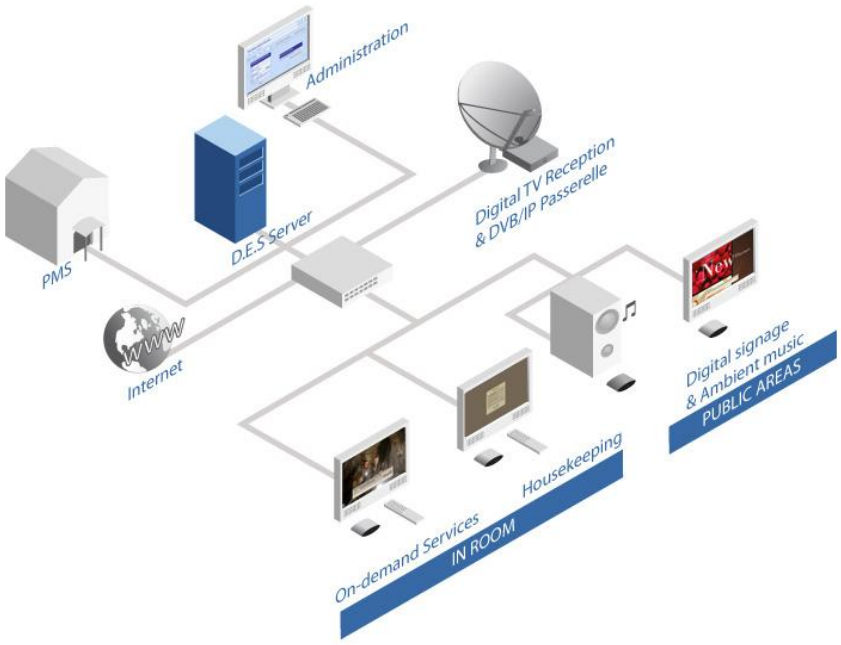


**Figura 1 Arquitectura do sistema DES da EONA [7]**

De seguida, faz-se uma breve referência a alguns dos principais componentes, abordando-se as suas características e finalidades.

### 2.1.1. RECEPÇÃO IPTV

O DES está associado a *gateways* Digital Video Broadcasting (DVB)/IP para fornecer, via satélite, terrestre ou por cabo, uma variedade de canais de televisão e de rádio com qualidade digital [8].

### 2.1.2. SERVIDOR

O DES é instalado num servidor Windows, dimensionado de acordo com o número de *set-top boxes* e serviços oferecidos. A este respeito, tanto pode ser implementado num cenário que utilize apenas um servidor local para um único edifício, como também pode ser

utilizado como servidor central que serve vários edifícios, através de ligações de fibra óptica ou gigabyte [8].

### 2.1.3. REDE

Em relação ao equipamento de rede, o DES pode ser adaptado a diferentes tipos de cabos: CAT 5/6, CAT 3 ou IP sobre coaxial.

O DES inclui um modo que assegura um acesso permanente aos canais de televisão em caso de avaria [8].

### 2.1.4. SET-TOP BOX

O DES utiliza *set-top boxes* Moving Picture Experts Group (MPEG) - 2 e H.264 para High-Definition (HD), as quais são certificadas pelos operadores. As STB podem ser associadas a um teclado com infra-vermelho para, por exemplo, navegar na internet na TV [8].

### 2.1.5. PROPERTY MANAGEMENT SYSTEM (PMS)

O DES possui uma interface para o sistema de gestão do hotel, o PMS, que automatiza vários processos (*check-in*, *check-out*, facturação, acesso restrito a alguns serviços, etc.). Associado a esta funcionalidade, o DES tem compatibilidade com vários PMS existentes no mercado, incluindo também um modo de gestão da facturação, na ausência do PMS [8].

## 2.2. NETRIS IPTV SOLUTION

A Netris, empresa sediada na Rússia, desenvolve *software* e integração de sistemas para vários provedores de serviços. Os principais produtos que esta empresa fornece são o iVision IPTV Middleware, o iStream VOD Server e o Netris Distributed Media Architect (DMA). Estes últimos podem ser encontrados na Figura 2, a qual apresenta, de uma forma sintética, a arquitectura do sistema [9].

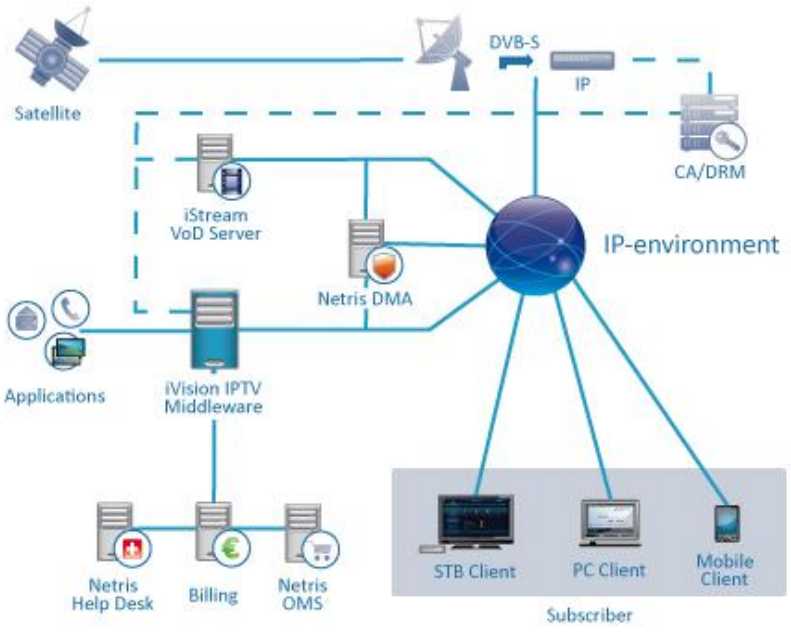


**Figura 2 Arquitectura do sistema IPTV da Netris [9]**

### 2.2.1. SERVIDOR DE VIDEO ON DEMAND

Os servidores de VOD são projectados para ter duas partes principais: a plataforma de *streaming* e o armazenamento de conteúdos [9].

### 2.2.2. NETRIS DMA

Nas arquitecturas de rede distribuídas, as operadoras necessitam de ferramentas para gerir o acesso e o fornecimento de conteúdos. O Netris DMA foi projectado para ser uma ferramenta que proporciona partilha de conteúdos e redundância num ambiente distribuído. Permite redireccionar as solicitações dos utilizadores para o servidor de vídeo mais próximo. Além disso, também distribui conteúdos de vídeo a partir do servidor central para os servidores locais, através de *multicast*. Desta forma, permite optimizar a arquitectura de rede, reduzindo o número de servidores de VOD [9][10].

### 2.2.3. IVISION IPTV MIDDLEWARE

O iVision IPTV Middleware é um componente chave neste sistema de IPTV. Este fornece uma plataforma inteligente baseada em arquitectura de rede, para suportar o armazenamento de vídeo, distribuição de conteúdo, personalização e funções de *streaming*, permitindo às operadoras oferecer essa mesma personalização, interactividade e conteúdo

local. Este sistema suporta a gama completa de aplicações de entretenimento de vídeo de próxima geração, das quais se pode mencionar o VOD, o Near Video On Demand (nVOD), o Network Digital Video Recorder (nDVR) e o TV *time/shifting*.

Além dos referidos serviços de vídeo interactivo, esta plataforma fornece serviços de comunicação (voz, vídeo, mensagens instantâneas, etc.) e serviços multimédia interactivos (jogos, aplicações, etc.) [9][10].

## 2.3. NEOS iTV

A NEOS Interactive, empresa sediada em Londres, Reino Unido, desenvolve produtos de entretenimento digital IP, com foco em tecnologias para a indústria hoteleira. O seu principal produto é o NEOS iTV, o qual inclui um menu integrado de entretenimento interactivo, informação, serviços de comunicação e sistemas de gestão.

Utilizando este sistema, é possível navegar na internet, quer através da televisão, quer através de um computador pessoal, uma vez que a STB pode ser "transformada" num ponto de acesso Wireless Fidelity (Wi-Fi). Desta forma, os hotéis não necessitam de investir numa solução de Wi-Fi separada, nem nos sistemas associados. É também fornecida uma interface para iPhone/iPod, integrada na própria STB, tornando-se assim desnecessário investir em iPod *docking stations*. Além disso, outra interface que vem incorporada com o NEOS iTV, é a de Compact Disc (CD)/Digital Versatile Disc (DVD)/Blu-ray Disc (BD), sendo que todos estes recursos reduzem significativamente o investimento necessário a realizar pelo hotel.

No que diz respeito ao aspecto gráfico da interface do NEOS iTV com o utilizador, pode ser personalizada à imagem do cliente (maioritariamente hotéis), tal como se observam dois casos distintos na Figura 3 [6].

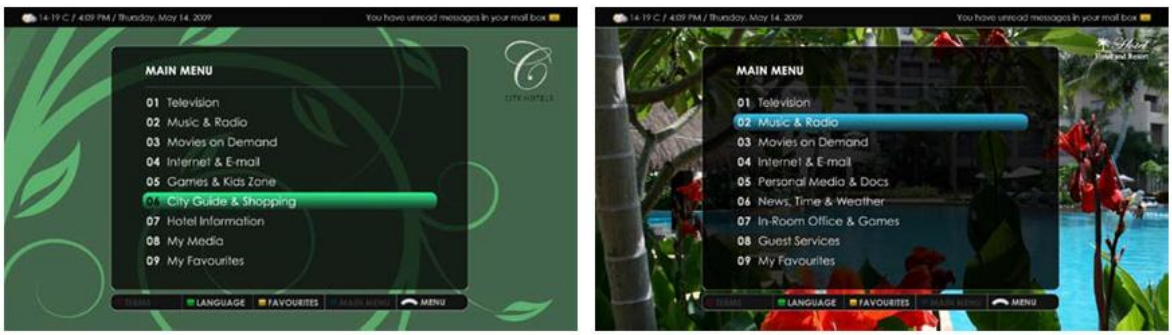


**Figura 3 Diferentes personalizações de temas do NEOS iTV [6]**

O NEOS iTV opera sobre uma plataforma Microsoft Windows, permitindo o recurso a uma mais vasta gama de conteúdos. Dependendo das necessidades específicas do cliente, existem diferentes STB disponíveis [6].

No que diz respeito a mais funcionalidades, os utilizadores têm a possibilidade de gravar e armazenar os seus canais de televisão, comprar filmes, álbuns de música ou ouvir rádio. Estes podem ser acedidos através do menu "My Favourites", o qual cria uma lista com as preferências do utilizador. Além disso, é também permitida a reprodução de áudio ao mesmo tempo que se utilizam outros serviços. Por exemplo, pode ouvir-se música enquanto se navega na *Web* ou se visualiza informação do hotel [6].

Noutro contexto, o sistema oferece ainda variados jogos, notícias actualizadas de jornais internacionais, bem como sites de notícias. Oferece também uma informação actualizada de meteorologia da localização do hotel e de outras zonas [6].

Uma outra característica importante é o facto de os utilizadores poderem associar à STB alguns dos seus dispositivos próprios. Por exemplo, pode associar-se iPods, cartões de memória ou CDs, permitindo reproduzir as próprias músicas, visualizar fotografias ou mesmo vídeos [6].

Pode referir-se também que o NEOS iTV inclui um sistema que gera relatórios mensais detalhados da utilização de cada máquina. Desta forma, permite que sejam verificadas as preferências dos utilizadores e os períodos em que os serviços são mais populares, orientando-se melhor as decisões de programação e os preços dos conteúdos. É também importante referir a possibilidade de integração do sistema de PMS do hotel com o NEOS iTV [6].

Por fim, e até servindo como termo de comparação com o projecto desenvolvido e descrito ao longo deste relatório (simulador do sistema NiVo), encontra-se no site oficial da NEOS Interactive uma demonstração em Flash do sistema NEOS iTV [5]. Aí pode visualizar-se vídeos demonstrativos de funcionalidades possíveis de realizar no sistema. Por outro lado, no simulador do sistema NiVo que será apresentado mais à frente, no capítulo 4, é possível interagir-se realmente com várias funcionalidades do NiVo e serviços associados, não se limitando unicamente à apresentação de vídeos demonstrativos.

## 2.4. NiVo

O NiVo é um sistema multimédia interactivo, principalmente direccionado para o mercado hoteleiro. Com a sua utilização, os hotéis oferecem aos seus clientes uma experiência de Full HD através das interfaces de fácil navegação (Figura 4), personalizadas à imagem do hotel, permitindo um fácil manuseamento deste sistema, pelo utilizador [4].

O NiVo disponibiliza um *browser* de navegação na TV, permitindo o acesso aos últimos conteúdos da *Web* 2.0, possibilitando aos hóspedes aceder ao Facebook, YouTube XL, Messenger, entre outros serviços. Adicionalmente, esta plataforma disponibiliza uma variedade de informação pública e serviços complementares, onde os hóspedes podem, por exemplo, reservar o campo de ténis, fazer compras ou ver o estado dos voos [4].


**Figura 4 Interface geral do NiVo [3]**

No que respeita às STB – NiVo Box, integram um ponto de acesso *wireless* que permite o acesso à Internet via Wi-Fi, tornando-se assim desnecessário o investimento na infra-estrutura e equipamentos Wi-Fi adicionais [3].

Além de tudo isto, a integração com o sistema PMS do cliente, permite a inclusão de venda de serviços como o VOD, reduzindo desta forma os custos da infra-estrutura Information Technology (IT) [3].

### 2.4.1. Arquitectura do Sistema

No que respeita à sua arquitectura, o NiVo é composto por vários elementos, entre os quais podem enumerar-se as STB, o NiVo Backend Server, o IPTV Streamer e o VOD Server. Na Figura 5 encontra-se um esquema exemplificativo da interligação dos diversos módulos [3].

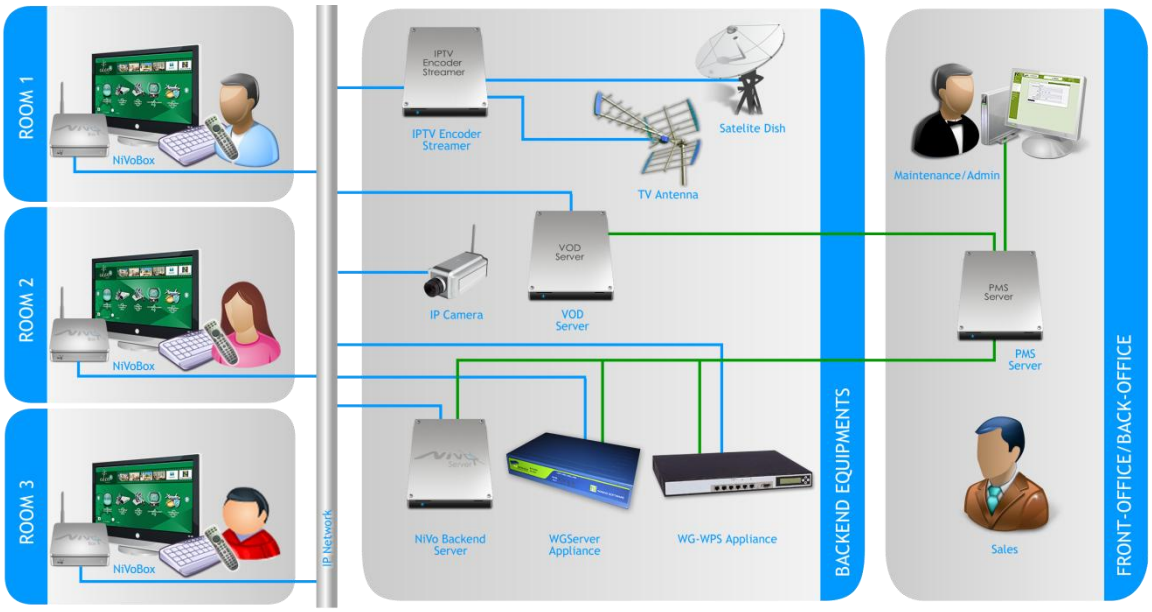


**Figura 5 Interligação dos dispositivos relacionados com o sistema NiVo [3]**

As NiVo Box encontram-se ligadas às TVs, tendo o NiVo Backend Server como responsável pela gestão e taxação de todos os serviços e pela integração com outros sistemas. No contexto do sistema, tem-se também o IPTV Streamer, o qual tem como funções receber e converter as emissões digitais ou analógicas de TV em formato IPTV, injectando-as na rede TCP/IP [3].

Para cada cliente (por exemplo, um hotel), ter-se-á um tema específico, personalizado à sua imagem. No que diz respeito, em particular, aos temas criados, estes são "alimentados" no *backend*, sendo propagados para as STB do sistema.

Olhando de uma perspectiva diferente para a arquitectura do sistema, fazendo uma abordagem em camadas, pode distinguir-se três camadas distintas que a compõem, tal como se demonstra na Figura 6.

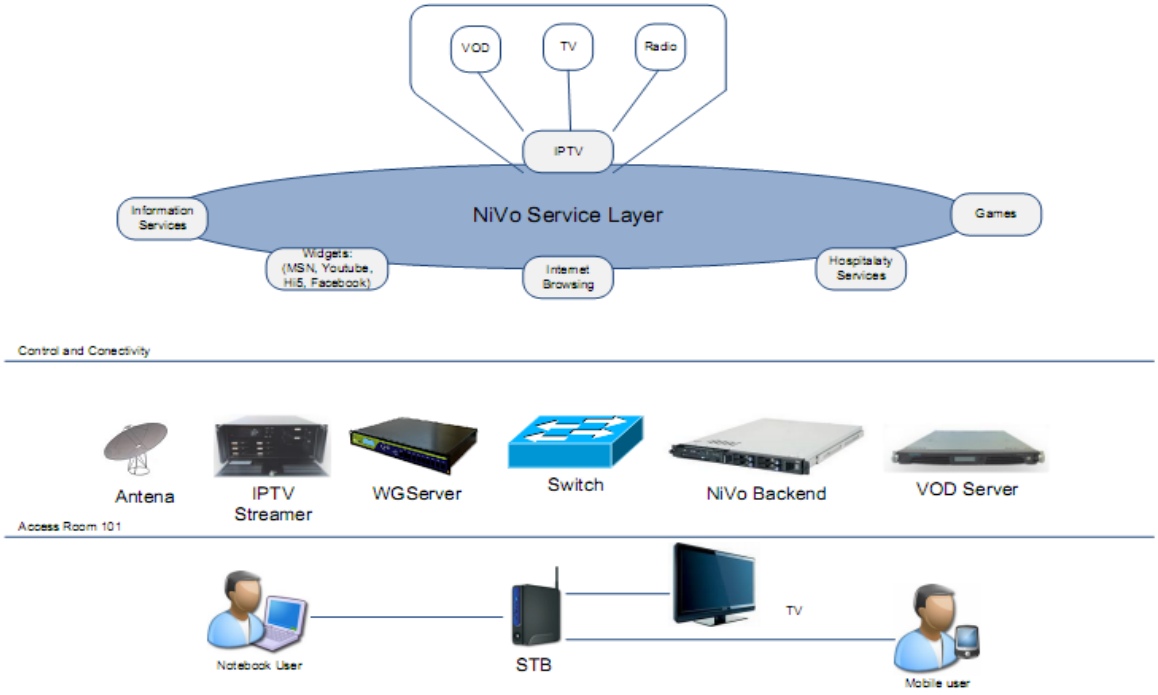


**Figura 6 Arquitectura em camadas do sistema NiVo**

A primeira, e mais importante camada, diz respeito à NiVo Service Layer, a qual incorpora todos os serviços NiVo, ou seja: IPTV com VOD, TV e Rádio; jogos; serviços para os clientes; navegação na Internet; aplicações *Web* 2.0 e serviços informativos.

A segunda é uma camada de controlo e conectividade, a qual corresponde a um centro de dados com todos os componentes de *hardware* e *software* necessários para implementar uma solução NiVo base.

A terceira pode identificar-se como Access Room Layer, reflectindo, por exemplo, o subsistema no quarto de hotel com a STB, TV e componentes para disponibilização de Internet sem fios, proporcionando uma interface com o cliente final.

Depois desta análise realizada à arquitectura do sistema, encontra-se de seguida uma abordagem particular a quatro dos seus principais componentes.

#### 2.4.1.1. NiVo Set-Top Box

As NiVo STB são basicamente mini PCs, os quais têm como base um sistema Linux personalizado pela Nonius. Uma característica importante é a sua modularidade. Por outras palavras, podem ser configuradas com *plugins* e temas, de modo a permitir obter o funcionamento e aspecto desejado [3].

#### 2.4.1.2. NiVo Backend Server

O sistema NiVo Backend é responsável por gerir todos os *firmwares* e configurações das *set-top boxes*, bem como fornecer o suporte para todos os serviços disponíveis para o utilizador final, através das STB.

O *backend* é uma máquina mais potente e melhor equipada, para tratar, por exemplo, das conversões e gestão de arquivos, deixando apenas a interface gráfica numa máquina menos potente, mas muito mais silenciosa, ligada à TV, a STB. Resumidamente, o NiVo Backend tem como principal função o fornecimento de conteúdos, bem como configurações para as STB e a gestão dos endereços IP da rede, através de Dynamic Host Configuration Protocol (DHCP) [3].

#### 2.4.1.3. IPTV Streamer

O módulo IPTV Streamer converte sinais de entrada de Digital Video Broadcasting – Satellite (DVB-S), Digital Video Broadcasting – Terrestrial (DVB-T) ou entrada analógica para o formato digital, distribuído dentro de uma rede IP. Tem a função de transmitir IPTV para a infra-estrutura local [3].

#### 2.4.1.4. VOD Server

O VOD Server tem como função o armazenamento de conteúdo VOD na infra-estrutura NiVo, distribuindo *streams* de vídeo, MPEG - 2 e MPEG-4 HD, pela rede [3].

### 2.4.2. PERSONALIZAÇÃO DA INTERFACE

O NiVo permite ao cliente a personalização da interface gráfica dos menus, de acordo com a sua imagem corporativa. Para que a personalização seja possível deverão ser considerados alguns requisitos necessários.


**Figura 7 Menu principal**

Deverá ser salvaguardada a hipótese de existir um suporte multilingue. Nesse sentido, no caso em que as imagens fornecidas contêm texto, serão necessárias versões dessas imagens com o texto nos diversos idiomas suportados. Nesta forma de implementação, é então necessária a criação de todos os ícones, repetidamente, para todos os idiomas pretendidos, sendo que o sistema interpreta cada idioma como seja um tema.

Em alternativa, de modo a diminuir o número de imagens a criar, e optimizar assim o processo de criação de temas, pode criar-se as imagens sem texto para cada tema, sendo posteriormente sobrepostas a estas as imagens apenas com texto, correspondente ao idioma em causa.

Estes requisitos prendem-se apenas à interface de navegação. Para cada aplicação ou *plugin* específico, a personalização deverá passar por uma especificação mais detalhada com o cliente.

No que diz respeito à interface, existem vários componentes que a compõem. Começando pelo *background*, que deverá ser uma imagem razoavelmente neutra, uma vez que todas as imagens seguintes serão sobrepostas a esta. Na Figura 8 pode verificar-se um exemplo de imagem utilizada como *background*.

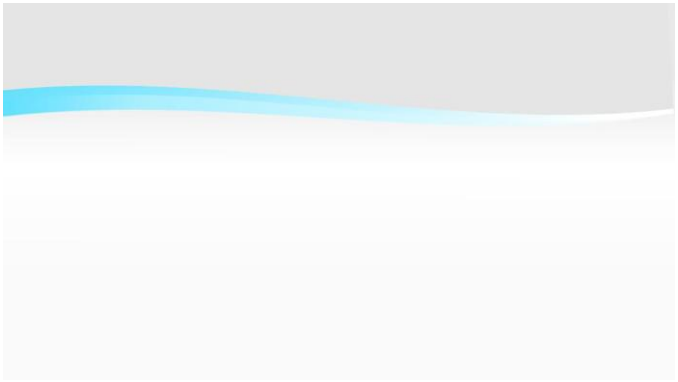

**Figura 8 Imagem de background**

De seguida, para a formação dos menus, outro componente necessário é a imagem de cabeçalho. Como exemplo, tem-se na Figura 9 um cabeçalho correspondente a um menu de selecção de canais televisivos.

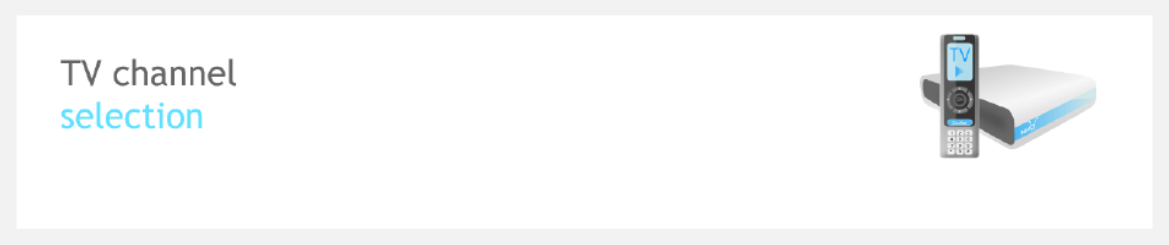


**Figura 9 Imagem de cabeçalho**

Todos os menus são igualmente constituídos por ícones. Para cada ícone serão necessárias duas imagens: uma para quando esse ícone se encontra seleccionado (*icon on*) e outra para quando não estiver seleccionado (*icon off*). Este caso encontra-se apresentado na Figura 10.

Além desta solução, outra hipótese seria, por exemplo, quando um ícone se encontra seleccionado, ficar com uma dimensão superior comparativamente a quando não se encontra seleccionado.

**Figura 10 Imagens de ícones Activo/Inactivo**

Para a navegação nos menus é necessária a criação de um ícone "Up", um ícone "Down", um ícone "Left" e um ícone "Right" (usualmente, uma seta a indicar o sentido para cima, para baixo, para a esquerda e para a direita). Estes deverão ser fornecidos em duas imagens (*on* e *off*), sendo que o estado "*on*" indica a existência de mais ícones/imagens no sentido correspondente. Caso contrário, ou seja, não havendo mais ícones/imagens a apresentar nesse sentido, a seta correspondente estará no estado "*off*". Na Figura 11 encontra-se um exemplo de ícones *Up/Down* de uma interface.

**Figura 11 Imagens de ícones Up/Down**

Por fim, a indicação da data e da hora está presente nos menus e submenus representada por ícones, como se exemplifica na Figura 12. Estes servem de fundo para o texto da data e da hora, que lhes será sobreposto de forma automática.

**Figura 12 Imagens de ícones Data/Hora**

Obviamente que, para haver conformidade, devem ser respeitados alguns parâmetros, entre os quais se encontra a dimensão de cada tipo de componente.

Na Tabela 1 apresentam-se, de forma sintética, as dimensões que cada componente deverá possuir.

**Tabela 1 Tabela sintética das dimensões dos componentes da interface**

| Componente da interface | Dimensões (em *pixels*) |
|---|---|
| **Background** | 1280x720 |
| **Cabeçalho** | 1280x241 |
| **Ícones de menu** | 170x260 |
| **Ícone da data** | 200x70 |
| **Ícone da hora** | 140x70 |
| **Ícones de navegação** | 36x36 |

Como exemplo de aspectos gráficos finais, na Figura 13 verificam-se dois casos de soluções diferentes de temas desenvolvidos à imagem dos respectivos clientes.


**Figura 13 Diferentes personalizações de temas do NiVo**

### 2.4.3. ESTRUTURA DOS TEMAS

Tal como referido anteriormente, cada idioma é interpretado como um tema, visto as imagens possuírem texto específico para cada um deles. Tendo este princípio como base, de seguida faz-se uma análise à estrutura actualmente implementada nos temas presentes nos *backends* do NiVo. Esta estrutura encontra-se representada graficamente no Anexo A.

Para cada idioma a estrutura é idêntica, obviamente diferindo apenas as imagens colocadas em cada directório, de acordo com o texto correspondente.

Como se pode verificar, dentro de cada um existem vários outros directórios, ficheiros de imagens e Extensible Markup Language (XML). No que respeita aos directórios, dividem-se por diversas categorias, contendo imagens e ícones associados. Como exemplo, pode referenciar-se os directórios “buttons”, “shared" ou “title”.

Os primeiros contêm os ícones de menus de várias categorias. Aqui não é criada uma organização, sendo que por exemplo, se encontram ícones de jogos e de outros submenus misturados.

Nos directórios “shared” encontram-se todos os ícones das setas de navegação, ao passo que o conteúdo dos directórios “title” corresponde aos cabeçalhos dos vários menus. Neste

último, todos os ficheiros das imagens dos cabeçalhos possuem a palavra “title_” no início do seu nome.

É igualmente importante verificar que neste tipo de estrutura não existe nenhum directório nem localização especificada para os ícones dos canais de TV e rádio. O directório correspondente a estes ícones encontra-se fora da estrutura dos temas, sendo transversal a todos, ou seja, para qualquer tema serão sempre utilizados os mesmos ícones de canais.

À parte dos directórios de categorias mencionados anteriormente, na própria raiz do tema (no directório do idioma em causa), encontram-se também algumas imagens. Entre estas imagens pode referir-se a imagem de *background*, as imagens de data e hora, e também setas de navegação nos menus.

Por último, igualmente na raiz dos temas, tem-se vários ficheiros XML com várias definições, como os tamanhos e fontes de textos, posições das imagens, incluindo também os caminhos para essas mesmas imagens que compõem o tema. Por exemplo, no ficheiro “theme.xml” encontram-se os cabeçalhos e ícones dos menus, no “base.xml” definem-se as imagens de *background*, data e hora, ao passo que no “nivomedia-ui.xml” especificam-se por exemplo, os cabeçalhos dos menus de TV e rádio.

# 3. FLASH

O Flash é utilizado para a criação de animações interactivas, sendo os ficheiros finais do tipo Shockwave Flash File (SWF). Estes podem ser visualizados quer numa página *Web*, utilizando um navegador que o suporte, quer através do Adobe Flash Player, o qual corresponde a uma aplicação apenas de leitura, distribuída gratuitamente pela Adobe, ou qualquer outro *player* de Flash.

Para além de simples animações, o Flash é utilizado como uma ferramenta para o desenvolvimento de aplicações completas. Para tal, muito contribuíram os avanços na linguagem ActionScript, sendo esta a linguagem de programação utilizada em aplicações do tipo SWF [11]. Com o ActionScript é possível realizar-se todo o tipo de projectos interactivos. Uma das maiores vantagens do Flash é a sua compatibilidade com todos os principais sistemas operativos e navegadores (*browsers*). Significa portanto, que a aplicação será visualizada da mesma forma em cada um destes sistemas.

## 3.1. ACTIONSCRIPT 3.0

O ActionScript é uma linguagem de programação orientada a objectos (OO) que está actualmente na sua terceira versão, também denominado como AS 3.0.

Na Tabela 2 pode verificar-se a evolução das várias versões do ActionScript, sendo possível constatar que o AS 3.0 foi lançado em 2006, em conjunto com o ambiente de desenvolvimento Adobe Flex 2.0 e o Adobe Flash Player 9.

**Tabela 2 Evolução da linguagem ActionScript**

| | | |
|---|---|---|
| 2000 | ActionScript 1.0 | Lançado com a versão 5 do Flash. Evoluiu a partir das *Actions* do Flash 4 |
| 2003 | ActionScript 2.0 | Surgiu com o lançamento do Flash MX 2004 e do Flash Player 7 |
| 2006 | ActionScript 3.0 | Lançada em conjunto com o Adobe Flex 2.0 e o Adobe Flash Player 9 |

De seguida, abordam-se vários aspectos específicos estudados sobre esta linguagem, necessários para o desenvolvimento do projecto. Ao longo de cada tópico apresentam-se alguns exemplos.

### 3.1.1. PRINCÍPIOS BÁSICOS

Em ActionScript é obrigatória a declaração de variáveis, bem como de funções e dos seus respectivos valores de retorno, mesmo não sendo retornado qualquer tipo de valor (*void*). Seguidamente, tem-se um exemplo de código em AS 3.0, que implementa o famoso "*Hello World!*" [12].

```
function helloWorld():void {
   trace("Hello World!");
}
```

No que respeita à declaração de variáveis, em AS 3.0 procede-se da forma apresentada abaixo, sendo recomendável separar-se cada linha do código por um ponto e vírgula, embora não seja obrigatório.

```
var nome:String = new String();
nome = "Paulo";

//Ou com um valor definido à partida
var nome:String = "Paulo";
```

### 3.1.2. UTILIZAÇÃO DO COMANDO *TRACE*

O comando *trace()* é utilizado para testar os valores das variáveis na ferramenta de desenvolvimento de Flash. No processo de desenvolvimento de um projecto, é difícil em vários casos saber qual o valor esperado para uma determinada variável. O comando *trace()* ajuda a testar esses valores, indicando-os na janela de saída [15].

A sua utilização deve ser feita em conjunto com pelo menos um valor. Por exemplo, se quiser simplesmente mostrar na janela de saída o texto "Hello World", basta realizar-se o comando apresentado abaixo.

```
trace("Hello World!");
```

Para testar-se o valor de uma variável, passa-se o nome da variável para o comando *trace(),* como se demonstra de seguida.

```
var nome:String = "Paulo";
trace (nome);
```

De outra forma, para se obter um retorno mais informativo, pode adicionar-se uma sequência de caracteres para indicar o nome da variável, juntamente com o seu valor, como se verifica no excerto de código apresentado de seguida.

```
var nome:String = "Paulo";
var idade:String = "22";
trace ("O meu nome é "+ nome);
trace ("A minha idade é "+ idade);
```

Além disso, o comando *trace()* pode ser utilizado para testar qualquer outro valor, tal como para recuperar os valores das propriedades de objectos que tenham sido adicionados ao *stage* (objecto/palco principal em Flash onde se pode adicionar outros objectos). Por exemplo, se tiver um *Movie Clip* com o nome "teste_mc" pode utilizar-se o comando trace da seguinte forma:

```
trace ("A posição X do teste_mc é "+ teste_mc.x);
trace ("A posição Y do teste_mc é "+ teste_mc.y);
```

O *Movie Clip* é um objecto que pode executar acções e receber interacção. Podem conter controlos interactivos, sons e outras instâncias associadas.

### 3.1.3. TRATAMENTO DE EVENTOS

Em AS 3.0, a manipulação de eventos é o processo através do qual se cria todo o tipo de interactividade [15].

Um evento corresponde a qualquer interacção que acontece no ambiente Flash. Entre vários existentes, pode enumerar-se, por exemplo, um clique do rato, acções no teclado ou mesmo a conclusão do processo de carregamento de um arquivo externo.

Em ActionScript pode fazer-se com que qualquer objecto reaja a qualquer evento, usando-se para isso um *Event Listener*. Para que um objecto reaja a um evento, utiliza-se o método *addEventListener*, o qual regista um *Event Listener* e associa um evento a um objecto.

Todo este processo é implementado em ActionScript utilizando-se o seguinte formato:

```
myObject.addEventListener(Event, eventListenerFunction);
```

Obviamente que o *Event Listener* necessita ser especificado, declarando-se a sua função da mesma forma que se declara qualquer outra função em ActionScript, com a diferença que a função do *listener* deve incluir um argumento para representar o evento, como se demonstra no excerto seguinte.

```
function eventListenerFunction (e:Event):void{
//Código ActionScript a ser executado quando ocorre o
evento.
}
```

Por exemplo, para que um objecto gráfico colocado no *stage* actue como um botão que reaja a um clique do rato, deve registar-se um evento e um *Event Listener*, tendo-se para tal o código exposto de seguida.

```
myButton.addEventListener(MouseEvent.CLICK,
myClickReaction);

function myClickReaction (e:MouseEvent):void{
trace("Foi clicado!");
}
```

Dependendo do evento que se pretende realizar, o objecto e o evento a registar serão diferentes. Por exemplo, se estiver a ser utilizada a classe *Loader* para carregar um ficheiro externo, pode executar-se uma acção específica apenas quando o ficheiro for totalmente carregado. Para tal, é necessário registar-se o evento *Event.COMPLETE*, tal como no exemplo seguinte.

```
my_loader.contentLoaderInfo.addEventListener(Event.COMPLE
TE, startListener);

function startListener (e:Event):void{
trace("Carregamento concluído");
}
```

Por outro lado, para se remover um evento, utiliza-se o método *removeEventListener()*, do mesmo modo que se aplica o método *addEventListener()*. Este método requer que se especifique o objecto ao qual se pretende remover o *Event Listener*, o evento para parar o *listener*, e a função atribuída a esse evento específico. Para uma melhor compreensão, o código seguinte demonstra como é feita a utilização deste método.

```
myObject.removeEventListener(Event,
eventListenerFunction);
```

### 3.1.4. CARREGAMENTO DE FICHEIROS XML

O XML é utilizado como uma forma de estruturar um determinado conteúdo, facilitando a sua análise e actualização de dados. Para estruturar esses dados logicamente, utilizam-se etiquetas (*tags*) que se assemelham ao HyperText Markup Language (HTML). No entanto, ao contrário do HTML, no XML são criadas *tags* próprias, com o objectivo de identificar perfeitamente esse conteúdo, tornando-o compreensível por qualquer ser humano.

Em AS 3.0, para carregar os dados XML, é necessário proceder-se em primeiro lugar à criação de uma variável para conter uma instância da classe XML, tal como se apresenta de seguida [15].

```
var myXML:XML;
```

Além disso, deve também criar-se uma instância da classe *URLLoader* para carregar o arquivo XML. A classe *URLLoader* é responsável por todos os carregamentos de dados binários e de texto. Após ser criada a sua instância, é possível utilizar-se o seu método *load()* para carregar o arquivo XML. Como exemplo do que acabou de ser referido, tem-se o excerto de código seguinte.

```
var myXML:XML;
var myLoader:URLLoader = new URLLoader();
myLoader.load(new URLRequest("myData.xml"));
```

Sendo solicitado o carregamento do arquivo XML, de seguida, para proceder-se ao processamento desse mesmo arquivo, deve em primeiro lugar certificar-se que este foi completamente carregado. Para tal, cria-se um evento para verificar quando o processo de carregamento é concluído e, em seguida, processar o arquivo XML. Esse evento deve ser associado à instância da classe *URLLoader*, tal como se demonstra no código abaixo.

```
myLoader.addEventListener(Event.COMPLETE, processXML);
```

Quando se conclui o carregamento, o evento irá accionar a função "processXML". A título de exemplo, esta função foi definida para especificar o conteúdo do arquivo XML, como se apresenta de seguida.

```
function processXML(e:Event):void {
myXML = new XML(e.target.data);
trace(myXML);
}
```

A referência ao "e.target" diz respeito ao objecto que carregou o arquivo XML.

### 3.1.5. INTERACÇÃO COM O TECLADO

A utilização de comandos do teclado requer que sejam atendidos eventos do teclado [15]. Este processo é idêntico ao processo de atendimento a qualquer outro evento em AS 3.0. Sendo assim, é necessário utilizar o método *addEventListener()* para proceder-se ao registo de um *KeyboardEvent*. No entanto, visto que o teclado não é necessariamente associado a nenhum objecto específico, o seu evento é normalmente registado no *stage*. No exemplo seguinte, ao objecto *stage* associa-se um *KeyboardEvent*, a ser accionado cada vez que uma tecla é pressionada, tendo-se associada a função *myKeyDown*. Esta função mostra na janela de *output* a mensagem "Key Pressed", sempre que uma tecla é pressionada.

```
stage.addEventListener(KeyboardEvent.KEY_DOWN,
myKeyDown);
function myKeyDown(e:KeyboardEvent):void{
trace("Key Pressed");
}
```

Para que se possa saber exactamente qual foi a tecla pressionada, pode utilizar-se duas propriedades pertencentes à classe *KeyboardEvent*, a *keyCode* e a *charCode*. A primeira corresponde a um valor numérico que representa a posição da tecla no teclado. A segunda propriedade enunciada corresponde a um valor numérico representando o caracter associado à tecla. A diferença entre estas duas propriedades reside no facto que, por exemplo, as letras "A" minúsculas e maiúsculas possuem o mesmo *keyCode*, mas diferentes *charCodes*.

No entanto, mais do que saber se o teclado está a ser utilizado ou não, torna-se essencial associar o código com o pressionar de uma tecla em particular e não à totalidade do teclado. Para tal, é necessário realizar um teste onde se compara o valor do *keyCode* com uma constante que representa a tecla que se pretende utilizar. Então, tem-se a classe *Keyboard* que possui constantes para todas as teclas. Por exemplo, para executar um

comando específico quando a barra de espaço é pressionada, deve associar-se uma condição como se apresenta no excerto de código seguinte.

```
stage.addEventListener(KeyboardEvent.KEY_DOWN,
myKeyDown);
function myKeyDown(e:KeyboardEvent):void{
if (e.keyCode == Keyboard.SPACE){
trace("Success!");
}
}
```

### 3.1.6. UTILIZAÇÃO DE MÁSCARAS

Uma máscara tem como objectivo limitar a visibilidade de um objecto com o formato da máscara aplicada a este, como se mostra no exemplo da Figura 14 [15].

Em AS 3.0, a qualquer objecto adicionado para exibição pode ser aplicada uma máscara. Para se realizar este procedimento, basta para isso utilizar a propriedade *mask* no formato mostrado abaixo, onde *my_object* é uma referência ao objecto a ser aplicada a máscara e *my_mask* é uma referência à própria máscara.

```
my_object.mask = my_mask;
```

Para se desactivar uma máscara de um objecto basta definir o seu valor como *null*.

```
my_object.mask = null;
```

A desactivação de uma máscara não vai eliminar o objecto, mas sim mostrá-lo como originalmente.


**Figura 14 Exemplo de utilização de máscaras em Flash [15]**

### 3.1.7. UTILIZAÇÃO DA CLASSE *TIMER*

Em AS 3.0, a classe *Timer* permite que se execute qualquer código repetidamente em intervalos de tempo específicos [15]. Para se poder utilizar a classe *Timer*, deve em primeiro lugar criar-se uma instância dessa mesma classe, definir o período de atraso e o

número de repetições. A propriedade de atraso corresponde ao intervalo de tempo, em milissegundos, no qual a classe *Timer* deve ser accionada. Por exemplo, para fazer-se com que um objecto se mova a cada três segundos, deve definir-se a propriedade de atraso com três mil milissegundos. Por outro lado, a propriedade associada ao número de repetições corresponde ao número de vezes que a classe *Timer* será accionada. No caso de se pretender que o *Timer* seja executado infinitamente, esta propriedade deve ser omitida.

Depois de criada a instância da classe *Timer*, escuta-se o *TimerEvent.TIMER*, utilizando-se para isso o método *addEventListener*. Associado a este último, cria-se a função de atendimento do evento para executar o código desejado. O *Timer* é iniciado através do método *start()*.

Todo este procedimento é generalizado no excerto de código apresentado abaixo.

```
var myIndetifier:Timer = new Timer(delay, repeat-count);
myIdentifier.addEventListener(TimerEvent.TIMER,
timerListener);
function timerListener (e:TimerEvent):void{
//commands
}
myIdentifier.start();
```

É possível parar-se o *Timer* a qualquer momento, mesmo antes do número limite de repetições que possa ter sido definido, usando-se para tal o método *stop()*. Para se utilizar este método é necessário aplicá-lo directamente na instância da classe *Timer*, ou seja:

```
myIdentifier.stop();
```

### 3.1.8. REPRODUÇÃO DE SONS

A reprodução de sons em AS 3.0 é realizada através da manipulação em conjunto de várias classes [15]. É um método mais complexo do que em versões anteriores do ActionScript, fornecendo, no entanto, um maior controlo e capacidade de gerir os sons. Nesse sentido, de seguida apresentam-se as mais importantes classes relacionadas com os sons.

A classe principal, na qual os sons residem, é a classe *Sound*. Esta é o ponto de partida de qualquer programa relacionado com o som e é utilizado para iniciar a reprodução do respectivo som associado.

Uma outra classe importante é a *SoundChannel*, fornecendo controlos adicionais sobre um objecto de som, como por exemplo, a capacidade de parar a reprodução.

Como responsável pela alteração do volume do som e pelo equilíbrio entre os altifalantes da esquerda e da direita, tem-se a classe *SoundTransform*. Além desta, existe também a classe *SoundMixer*, a qual possui um controlo global sobre todos os sons reproduzidos no *player* de Flash. A sua função mais utilizada é a de parar todos os sons, independentemente da sua origem.

Para se iniciar a reprodução de um arquivo de som, deve em primeiro lugar criar-se uma nova instância da classe *Sound*, utilizando-se depois o método *load()* para carregar o arquivo externo. Este processo é exemplificado no excerto de código seguinte.

```
var mySound:Sound = new Sound();
mySound.load(new URLRequest("Song.mp3"));
mySound.play();
```

O método *load()* da classe *Sound* é utilizado para se carregar um arquivo MPEG-1/2 Audio Layer 3 (MP3) externo. O seu Uniform Resource Locator (URL) deve ser especificado como uma instância da classe *URLRequest*.

Além da capacidade de reproduzir sons, é também importante saber como os parar depois de os estar a reproduzir. Em relação a este aspecto, a classe *Sound* não possui nenhum método para parar um som. Para executar esta tarefa deve utilizar-se a classe *SoundChannel*. Esta classe possui vários métodos e propriedades para manipular o som. Em particular, tem-se então o método *stop()*, com a finalidade de parar a reprodução do som através do canal.

### 3.1.9. UTILIZAÇÃO DE TEXTOS

O método de criação de um campo de texto é semelhante à criação de uma instância de qualquer outra classe em AS 3.0 [15]. Passa então pela criação de uma variável para conter essa instância e, em seguida, utilizar a palavra "new" para gerar o *TextField*, tal como apresentado de seguida:

```
var myText:TextField = new TextField();
```

Estando já criado o campo de texto, pode definir-se o texto a ser exibido, configurando-se a propriedade *text* da instância. No exemplo apresentado abaixo define-se como "Exemplo de texto".

```
myText.text = "Exemplo de texto";
```

Depois de criado o *TextField* e definido o seu texto, para que ele seja visível, deve-se adicioná-lo a lista de exibição, ou seja, utilizar-se o método *addChild()*.

```
addChild(myText);
```

Mesmo depois de adicionado à lista de exibição, é possível actualizar o conteúdo, alterando-se o valor da propriedade *text*, quando desejado.

Dependendo das necessidades do projecto, pode utilizar-se outras propriedades disponíveis para alterar o aspecto visual do *TextField*. No entanto, algumas das propriedades de formatação, como o tamanho, o alinhamento ou a fonte, são configuradas através da classe *TextFormat* e não através da classe *TextField*.

No que diz respeito às propriedades que se acede directamente através da instância do campo de texto, tem-se por exemplo, a propriedade *textColor*, que permite alterar a cor do texto, definindo-a com qualquer valor de cor em hexadecimal. Pode também modificar-se as dimensões de largura e altura do campo de texto, definindo as propriedades *width* e *height* da respectiva instância. Além disso, obviamente que também é possível alterar a posição do campo de texto, definindo-se para isso as propriedades *x* e *y*.

A classe *TextField* possui várias propriedades úteis, no entanto, não controla os aspectos de formatação do texto, tais como o tamanho, o alinhamento ou a fonte. Estas propriedades são configuradas através da classe *TextFormat*. É necessário criar-se uma instância desta classe, definindo as suas propriedades, atribuindo-a posteriormente como formato de texto padrão a um *TextField*. Um exemplo deste processo é apresentado no excerto de código seguinte:

```
var myFormat:TextFormat = new TextFormat();
myFormat.size = 15;
var myText:TextField = new TextField();
myText.defaultTextFormat = myFormat;
```

Como referido, é também possível configurar-se o alinhamento do texto, utilizando-se a propriedade *align* da classe *TextFormat*. Esta propriedade só aceita valores específicos passados como propriedades da classe *TextFormatalign*. Como são o caso, por exemplo, do *TextFormatAlign.CENTER*, o *TextFormatAlign.JUSTIFY* ou o *TextFormatAlign.LEFT*.

Além das propriedades referidas, a classe *TextFormat* possui ainda outras, como a *bold* ou a *letterSpacing*, entre outras.

## 3.2. FUNCIONAMENTO DO ACTIONSCRIPT

Depois de ser compilado, o código é incluído num arquivo SWF para ser executado pela ActionScript Virtual Machine (AVM), actualmente na terceira versão, que é um componente disponível no Adobe Flash Player (*plug-in* encontrado em *browsers*) e no Adobe Integrated Runtime (AIR) [16]. Este último corresponde a uma tecnologia que permite a criação de aplicações a partir de tecnologias de desenvolvimento de páginas *Web*, como HTML, Ajax ou Flash.

O código ActionScript pode ser desenvolvido e editado com recurso a diversos editores com suporte a ActionScript. Além do *software* da Adobe, actualmente o Adobe Flash CS5, existem outros pacotes de *software* gratuitos para o efeito. O escolhido para o desenvolvimento deste projecto foi o FlashDevelop, o qual é igualmente indicado pela própria Adobe.

## 3.3. FLASHDEVELOP

O FlashDevelop, além de ser gratuito, possui algumas características importantes, as quais serão abordadas de seguida.

Na Figura 15 pode visualizar-se a interface do *software*, onde na página inicial se encontram os projectos recentes ou informações e actualizações do produto. Além disso, no lado direito, tem-se o painel do projecto, onde por exemplo, é possível a criação de novas classes a partir de modelos [14].

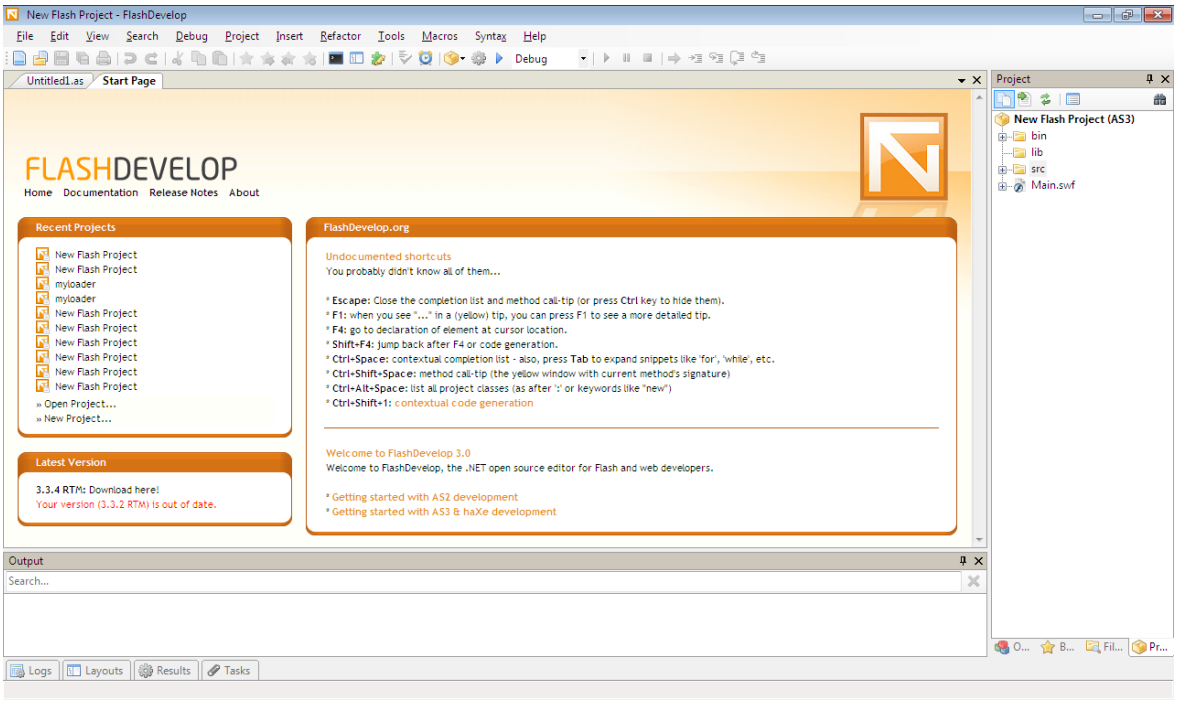


**Figura 15 Interface do *software* FlashDevelop**

Uma característica importante do FlashDevelop diz respeito à sua capacidade de completar o código [13]. Por exemplo, como se pode observar na Figura 16, escrevendo "*testeMovieClip.add"*, são fornecidas as opções possíveis para escolher. Pode fazer-se a selecção utilizando-se as setas do teclado, sendo fornecida uma descrição específica. Obviamente que se continuar a escrever, restringe-se mais a escolha abaixo.

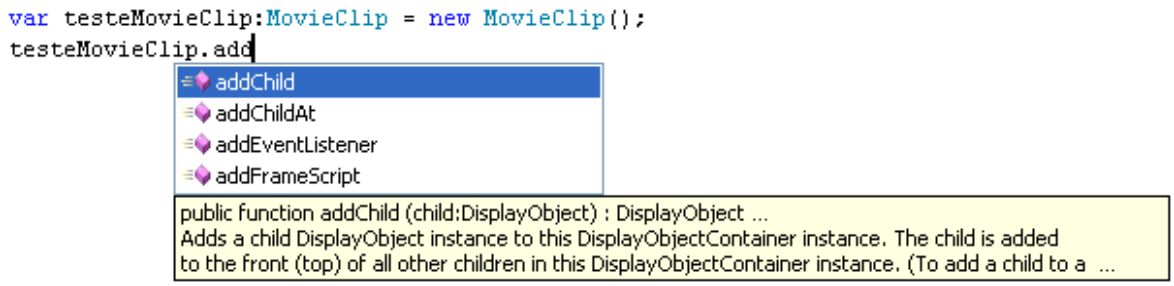


**Figura 16 Capacidade do FlashDevelop a completar código**

Uma outra particularidade do FlashDevelop é o facto de fazer adição automática de *imports* [13]. Como na figura anterior, quando se declara o *MovieClip* é automaticamente adicionado o *import* correspondente, no local correcto, na parte superior do arquivo da classe.

```
import flash.display.MovieClip;
```

Além disso, uma outra característica do FlashDevelop é a criação automática de objectos [13]. Acontece quando se escreve o nome de uma variável ou função que ainda não foi criada. É frequente quando se escreve funções de manipulação de eventos ao definir um *listener*, tal como no próximo caso.

```
myButton.addEventListener(MouseEvent.CLICK,
onClickMyButton);
```

Não tendo ainda definido a função *onClickMyButton()*, uma forma rápida de o fazer, utilizando as funcionalidades deste *software*, é colocar o cursor sobre a palavra-chave ainda não definida e pressionar Ctrl+Shift+1, abrindo-se o menu demonstrado na Figura 17.

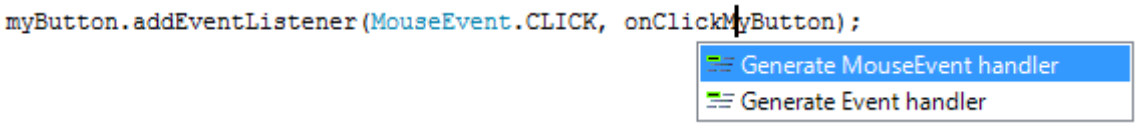


**Figura 17 Menu de criação automática de objectos no FlashDevelop**

Pressionando *enter*, o FlashDevelop irá criar automaticamente uma nova função, com o parâmetro correcto, como se comprova no código seguinte.

```
myButton.addEventListener(MouseEvent.CLICK,
onClickMyButton);

private function onClickMyButton(e:MouseEvent):void
{

}
```

Este procedimento pode ser realizado da mesma forma com variáveis que não tenham ainda sido definidas, entre vários outros casos. Ctrl+Shift+1 é portanto um dos atalhos mais eficazes do FlashDevelop.

Passando agora para mais uma funcionalidade muito útil no FlashDevelop, faz-se referência ao menu *Refactor* [13]. Este é utilizado quando se criou um objecto com um determinado nome, mas por alguma razão, posteriormente se pretende alterá-lo. Para evitar a perda de tempo a realizar o procedimento de localizar e substituir, torna-se muito mais eficaz aceder ao menu *Refactor*. Tal como se apresenta na Figura 18, pressiona-se o botão do lado direito do rato sobre qualquer nome da função ou variável que se pretende alterar o nome e escolhe-se *Refactor -> Rename…*

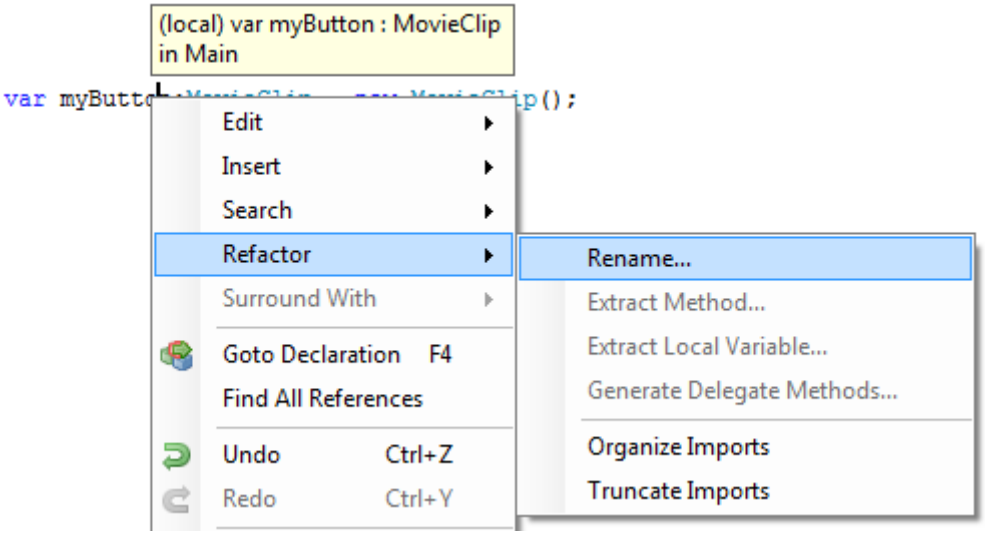


**Figura 18 Utilização do menu *Refactor* no FlashDevelop**

Desta forma, todas as referências à variável ao longo do projecto, incluindo mesmo a própria variável, são renomeadas para o que for introduzido. Além disso, o menu *Refactor* permite por exemplo, organizar os *imports*, ou mesmo remover *imports* que já não estejam a ser necessários.

Por último, faz-se referência a situações em que se encontram "bugs" ou alguma coisa que se pretende corrigir, mas que se deixa pendente. A solução rápida é deixar um comentário no local do código específico para regressar lá mais tarde [13]. No entanto, obviamente que é difícil com o passar do tempo fazer a gestão mental de todas as notas que forem surgindo. No FlashDevelop é possível fazer essa gestão adicionando comentários com o prefixo `//FIXME`. Desta forma, todas as anotações aparecerão no painel de tarefas.

# 4. SIMULADOR NIVO

Neste capítulo aborda-se a aplicação desenvolvida, em Flash, para simular a interactividade dos temas criados para o sistema NiVo. Este simulador traz várias vantagens para o processo de desenvolvimento de temas gráficos. Permite que, em qualquer plataforma, sem ser necessário implementar no sistema real, se possa verificar o aspecto e interactividade do tema em desenvolvimento. Torna mais rápido e eficaz este processo, visto poder mais facilmente verificar-se a conformidade das imagens e ícones criados. Por exemplo, no caso de existirem ícones de tamanhos diferentes, são facilmente encontrados.

Esta ferramenta pode ter vários utilizadores finais. Desde logo, tem-se o designer das imagens e toda a equipa responsável pelos aspectos gráficos relacionados com a criação de um tema, ou mesmo a equipa de desenvolvimento de *software*. Além disso, visto ser um simulador que recorre à tecnologia Flash, facilmente integrada num navegador *Web*, uma aplicação passaria pela sua colocação no site oficial da Nonius, um pouco à semelhança do que foi apresentado na secção 2.3, referente ao caso da empresa NEOS Interactive. Em termos comparativos com este último, constata-se que o simulador NiVo permite uma maior interactividade através dos serviços disponibilizados, fornecendo ao utilizador uma melhor percepção das capacidades do produto. Ao invés de ser fornecida apenas uma demonstração do sistema, um simulador próximo do real torna-se mais apelativo.

Uma outra finalidade importante prende-se com a possibilidade de envio do simulador para o cliente que solicitou o serviço específico. Desta forma, este pode mais facilmente visualizar o aspecto do seu produto, podendo fornecer à Nonius um *feedback* mesmo ainda

na fase de desenvolvimento do tema. Este aspecto é também importante para aumentar a rapidez e eficácia do processo de criação de um novo tema.

## 4.1. ESTRUTURA DA APLICAÇÃO

Como se mostra na Figura 19, a aplicação desenvolvida é constituída por três ficheiros base: o ficheiro SWF, "NiVo.swf", o ficheiro XML, "nivoTheme.xml" e o ficheiro de texto, "variables.txt", bem como cinco directórios essenciais: o "playlists", que contém todos os ícones de canais de TV e rádio, os quais são transversais a qualquer tema, o "games", que contém os SWF de cada jogo, o "videos", armazenando os ficheiros Flash Video (FLV) demonstrativos de cada canal de TV, o "radios" que contém ficheiros MP3 de gravações de rádios, bem como ficheiros FLV de visualizações gráficas das oscilações do som e, por fim, o "archive", onde se localizam *screenshots* de cada submenu que não possua interacção, bem como a imagem de *background* utilizada para os canais de TV e rádio.

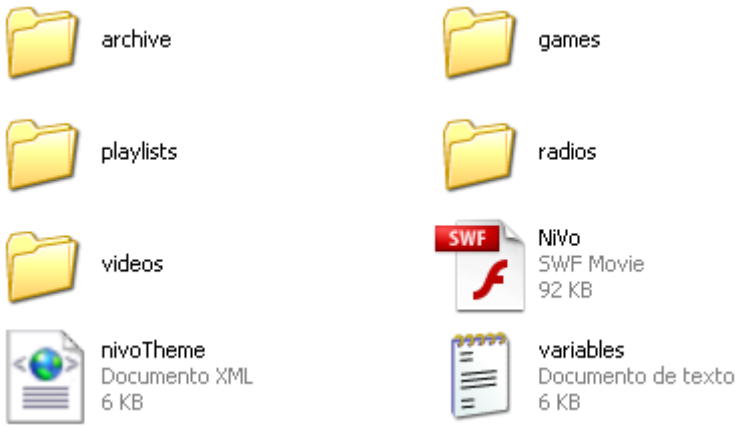


**Figura 19 Estrutura de ficheiros do simulador do NiVo**

De seguida, faz-se uma breve abordagem a cada um dos ficheiros base anteriormente referidos.

Em relação ao SWF, este é criado quando se faz a compilação do projecto. Neste caso, o ficheiro principal do projecto é o "Main.as", o qual é abordado na secção 4.1.3, podendo destacar-se a sua constituição por vários métodos, com a finalidade de obtenção da interface e interactividade pretendida para o simulador. Nele é feito o carregamento do XML, "nivoTheme.xml", desenvolvido com o intuito de definir uma hierarquia de menus, ou seja, níveis de menus a apresentar. No que respeita a este último, demonstra-se na

secção 4.1.2 a sua importância. Além disso, é importante referir que todas as variáveis utilizadas são carregadas a partir do ficheiro "variables.txt", ao invés de serem definidas directamente no código. Desta forma, qualquer alteração que seja necessária realizar, por exemplo a posição de imagens ou espaços entre ícones, não obriga a nova compilação do programa, bastando executá-lo novamente para se verificar as modificações.

### 4.1.1. FICHEIRO DE TEXTO (VARIABLES.TXT)

Como foi referido, no ficheiro de texto "variables.txt" tem-se a definição de variáveis que serão posteriormente carregadas no ficheiro ActionScript. Esta forma de tratamento visa criar uma abstracção no que respeita a valores que teriam que ser repetidos várias vezes ao longo do código ActionScript, tornando-o de mais fácil leitura e mais facilmente alterável. É importante realçar que basta alterar os valores no "variables.txt", para se verificar o resultado na execução do SWF final do simulador, não sendo necessária nova compilação do ficheiro ActionScript, o "Main.as".

No excerto seguinte apresenta-se parte deste ficheiro, com a definição de algumas variáveis.

```
velDeslizamento=0
&espacoEntreIcon=27
&myXPos=0
&myXPos2=0
&my_x=0
&my_y=310
&my_icone_width=213
&my_icone_height=287
(…)
```

Como pode verificar-se, encontram-se por exemplo, definições de valores de espaçamentos entre ícones, bem como as suas dimensões e posicionamentos no espaço dos contentores desses mesmos ícones. No que respeita à sua sintaxe, as várias variáveis devem ser separadas por um "e comercial" ("&").

### 4.1.2. FICHEIRO XML (NIVOTHEME.XML)

Para se realizar a simulação do sistema, tem-se como necessária a utilização de alguns dos ficheiros XML presentes no tema em causa. Esses ficheiros são o "base.xml", o "nivomedia-ui.xml" e o "theme.xml". Apesar de já existirem estes XML, surgiu a necessidade de associar um novo ao simulador, o "nivoTheme.xml". A razão do seu desenvolvimento prende-se com o facto de, no "theme.xml", não ser definida a hierarquia

dos vários menus, ou seja, não ser possível, por exemplo, distinguir as imagens que estão associadas ao menu principal de outras que pertencem a submenus. No excerto de código seguinte, correspondente ao "theme.xml", demonstra-se exactamente esta problemática.

```
<buttondef name="INFO_FLIGHT_PORTO">
 <image>buttons/INFO_FLIGHT_PORTO_off.png</image>
 <activeimage>buttons/INFO_FLIGHT_PORTO_on.png</activeimage>
 <offset>0,0</offset>
</buttondef>

<buttondef name="BILLING">
 <image>buttons/BILLING_off.png</image>
 <activeimage>buttons/BILLING_on.png</activeimage>
 <offset>0,0</offset>
</buttondef>
```

Como se pode verificar, associados à mesma *tag* do XML ("buttondef"), tem-se por exemplo, os ícones (*on* e *off*) do aeroporto do Porto, que pertencem ao menu "Estado dos voos", e os ícones (*on* e *off*) dos "Consumos" (*Billing*), os quais pertencem ao menu principal.

Para contornar esta situação e conseguir ter uma hierarquia bem definida dos menus a simular, criou-se portanto o "nivoTheme.xml". Este ficheiro é utilizado para criar uma ligação às *tags* correspondentes no ficheiro "theme.xml" já existente, garantindo-se então o conhecimento da hierarquia. No "nivoTheme.xml", as únicas referências directas a imagens no formato Portable Network Graphics (PNG) são aos ícones dos canais de TV e rádio e aos *screenshots* obtidos. No que respeita aos ícones dos canais de TV e rádio, localizados no directório "playlists", são transversais a qualquer um dos temas do sistema NiVo, daí não haver problema de estarem definidos neste novo XML. Além destas imagens referidas, no "nivoTheme.xml" incluem-se também referências aos vídeos FLV dos canais de TV e os SWF dos jogos.

No próximo excerto de código, apresenta-se parte do "nivoTheme.xml", podendo-se verificar as características referidas anteriormente.

```
<?xml version="1.0"?>
<nivotheme>
 <buttons>
(…)
       <buttondef name="RADIO">
         <list>
            <image>playlists/radio_logos/cidadefm.png</image>
            <image>playlists/radio_logos/comercial.png</image>
            <image>playlists/radio_logos/megafm.png</image>
         </list>
```

```
      </buttondef>

      <buttondef name="TV">
        <list>
            <image>playlists/tv_logos/rtp.png</image>
            <image>playlists/tv_logos/rtp2.png</image>
            <image>playlists/tv_logos/sic.png</image>
            <image>playlists/tv_logos/tvi.png</image>

            <video>videos/rtp1_meo.flv</video>
            <video>videos/rtp2_meo.flv</video>
            <video>videos/sic_meo.flv</video>
            <video>videos/tvi_meo.flv</video>
        </list>
      </buttondef>

      <buttondef name="GAMES">
        <buttongame name = "PACMAN">
            <gameswf>games/Pacman.swf</gameswf>
        </buttongame>

        <buttongame name = "FORMULA1">
            <gameswf>games/Formula_1.swf</gameswf>
        </buttongame>
(…)
      </buttondef>
(…)
 </buttons>
</nivotheme>
```

### 4.1.3. FICHEIRO ACTIONSCRIPT (MAIN.AS)

De modo a criar-se o aspecto gráfico e a interactividade associada ao simulador, desenvolveram-se vários métodos em ActionScript, tendo em consideração cada finalidade específica (carregamento de ficheiros, criação de componentes gráficos, etc.). Para tal, muito contribuíram as características e propriedades da linguagem ActionScript, algumas delas abordadas no capítulo 3.

Tal como foi referido anteriormente, em primeiro lugar tem-se os métodos de processamento dos vários ficheiros XML, bem como das variáveis contidas no ficheiro de texto "variables.txt". Desta forma, obtém-se as localizações de todas as imagens necessárias para a composição do tema, como também se associam várias variáveis obtidas externamente.

De seguida, utilizando-se as informações obtidas, desenvolveram-se vários métodos com a finalidade de criar toda a interface gráfica do simulador. Aqui incluem-se, por exemplo, a criação dos *MovieClips* para as imagens e ícones dos temas (*background*, cabeçalhos,

menus), bem como a definição das respectivas posições, dispondo-os na área de visualização.

Posteriormente, tendo já a interface gráfica criada, desenvolveu-se um método para atender às acções do teclado, criando assim a interactividade pretendida. Associado a esta interacção, tem-se ainda outros métodos referentes a vários menus em particular. Entre estes, pode referir-se, por exemplo, a reprodução dos canais de TV e de rádio, como também o carregamento dos ficheiros SWF dos jogos.

No Anexo B encontra-se uma descrição mais pormenorizada das partes principais do código ActionScript desenvolvido.

## 4.2. FUNCIONAMENTO DA APLICAÇÃO

Para se proceder à simulação de um tema utilizando a aplicação desenvolvida (simulador), deve em primeiro lugar, copiar-se todos os ficheiros e arquivos associados à sua estrutura (descrita na secção 4.1) para o directório do tema em causa. Depois de realizado este passo, há alguns aspectos de "configuração" a considerar. Um ponto importante é a necessidade de alterar os *screenshots* localizados na pasta "archive". A simulação de alguns menus que não possuem submenus é realizada com *screenshots* obtidos directamente de um sistema real do NiVo e, como tal, sempre que se simula um tema diferente, a pasta "archive" deverá conter *screenshots* correspondentes ao mesmo. Em relação a este aspecto, uma possível alteração futura ao modo de desenvolvimento dos próprios temas, seria a adição da pasta "archive" directamente no tema, já com os *screenshots* necessários para a simulação. De realçar que os nomes destes ficheiros PNG deverão estar sempre de acordo com os definidos no "nivoTheme.xml".

No que respeita aos menus a simular e ícones que cada um contenha, tem-se essa definição no ficheiro "nivoTheme.xml". Portanto, a adição ou remoção de determinado menu, passa pela adição ou remoção do nó correspondente na estrutura.

Tendo-se realizado todos os procedimentos necessários, a simulação de um tema é iniciada pela execução do ficheiro "NiVo.swf".

## 4.3. EXEMPLO DE SIMULAÇÃO

Em primeiro lugar, surge o menu principal, tal como é apresentado na Figura 20. Nesse menu, através da utilização das teclas para a esquerda e direita do teclado é possível navegar-se pelos seus ícones, sendo o ícone central sempre o seleccionado. A entrada no próximo menu desejado é feita pela acção da tecla *enter* do teclado. Mediante essa mesma acção, é apresentado o novo contentor de ícones e alterado o título para o correspondente. É importante referir que sempre que se pretende retroceder de menu, tem-se para o efeito a tecla *esc* do teclado.


**Figura 20 Exemplo de simulação do sistema NiVo: menu principal**

Por exemplo, entrando no menu de selecção de canais de TV (Figura 21), tem-se então a lista de canais disponíveis, tendo a mesma interactividade que o menu principal. Neste caso e no caso dos ícones de canais de rádio, não existe ícones *off* e ícones *on*. De outra forma, tem-se apenas um tipo de ícones, havendo no ícone seleccionado um *background* no contentor de ícones. É este *background* que se encontra localizado na pasta "archive" da aplicação, e pode ser alterado mediante o tema a utilizar/simular.


**Figura 21 Exemplo de simulação do sistema NiVo: menu de selecção de canais TV**

Tendo sido seleccionado o canal desejado e pressionada a tecla *enter*, é iniciado neste caso a reprodução de um vídeo (em formato FLV), demonstrativo desse mesmo canal, tal como se pode constatar na Figura 22.


**Figura 22 Exemplo de simulação do sistema NiVo: reprodução de canal de TV**

No menu de jogos, apresentado na Figura 23, é mesmo possível jogar realmente um dos jogos disponíveis na lista. Para tal, contribui o facto de esta aplicação ser desenvolvida usando a plataforma Flash, proporcionando a interactividade necessária para abrir os jogos que são igualmente ficheiros SWF. Ou seja, sempre que se entra no jogo seleccionado, é iniciada a reprodução do correspondente SWF.

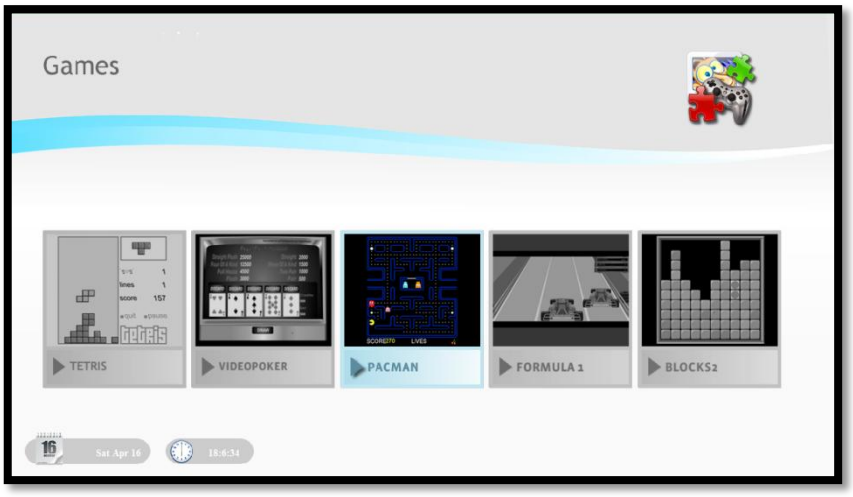


**Figura 23 Exemplo de simulação do sistema NiVo: menu de jogos**

Até este ponto exemplificou-se o funcionamento de menus cuja interacção no simulador se pode considerar dinâmica, visto no caso dos canais de TV se iniciar a reprodução de um vídeo para cada canal e para o caso dos jogos se iniciar efectivamente a reprodução do jogo escolhido.

Passando agora para um menu que possui um aspecto gráfico diferente, o menu de Video On Demand, a solução adoptada para realizar a sua simulação passa pela obtenção de vários *screenshots*, nas várias posições possíveis, simulando a interactividade através dessas mesmas imagens. Na Figura 24 expõe-se o aspecto do menu em causa. Além do referido aspecto gráfico diferente, este possui igualmente um tipo de interactividade diferente. Como se pode verificar, tem-se um menu com várias categorias de filmes, disponibilizado no lado esquerdo do ecrã. A navegação neste menu é realizada com as teclas para cima e para baixo do teclado, fazendo alterar no menu do lado direito a lista de filmes, mediante a categoria seleccionada. No exemplo apresentado apenas é disponibilizado um filme na categoria "Action". No entanto, caso existissem mais filmes, a respectiva escolha seria realizada utilizando as teclas para a direita e esquerda do teclado, alterando o filme seleccionado.

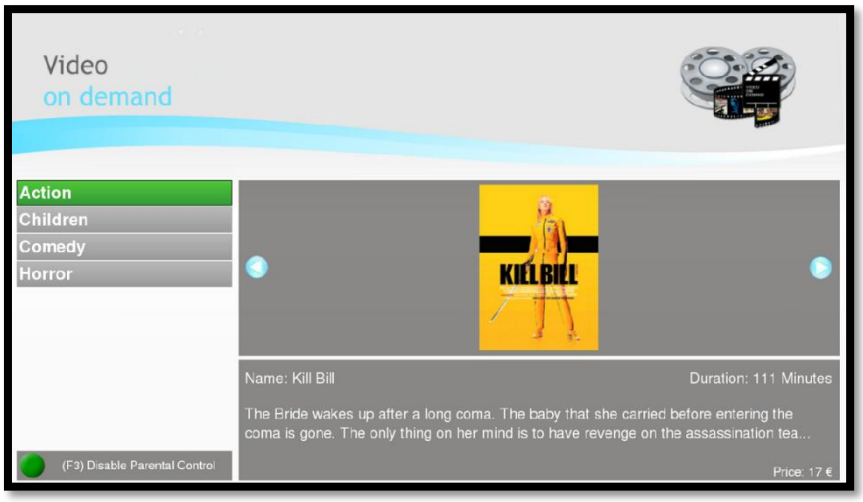


**Figura 24 Exemplo de simulação do sistema NiVo: menu de VOD**

Ainda no menu de VOD, outra interacção disponível através do simulador é obtida utilizando-se a tecla *enter* sobre o filme seleccionado, surgindo uma descrição mais detalhada sobre o filme em questão, tal como se demonstra na Figura 25.

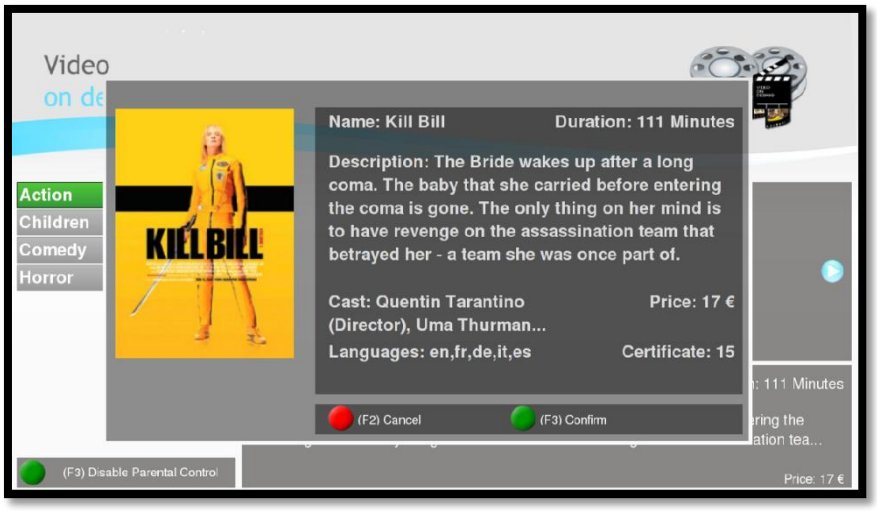


**Figura 25 Exemplo de simulação do sistema NiVo: menu de VOD com descrição de filme**

Passada a exposição do menu de VOD, foca-se agora num caso em que se depara com uma hierarquia de vários submenus encadeados. Neste exemplo é realizado o foco no menu de "Serviços Informativos", o qual é disponibilizado no menu principal. Entrando neste menu, obtém-se portanto um novo contentor de ícones, obviamente dentro do contexto em causa. Tal como é apresentado na Figura 26, tem-se por exemplo os ícones de "Notícias", "Estado dos voos" e "Farmácias de serviço". Embora não seja visível, outro ícone disponível é o de "Estado do tempo", podendo ser seleccionado através da utilização da navegação com o teclado, sobre o *sliding menu* (menu que roda infinitamente) apresentado.

Outro aspecto importante de realçar neste exemplo é a organização dos ícones. Neste caso, o menu de "Serviços Informativos" possui quatro ícones. Visto o ícone seleccionado ser sempre o central, o número de ícones apresentados deve ser sempre ímpar, de modo a surgir o mesmo número de ícones de cada lado do seleccionado. Daí surgirem três ícones e o quarto ficar "escondido", sendo o menu circular. Como é de fácil visualização na Figura 26, as posições dos ícones no espaço do menu são diferentes do caso em que se tem cinco ou mais ícones disponíveis.


**Figura 26 Exemplo de simulação do sistema NiVo: menu de Serviços Informativos**

Como foi referido, neste caso tem-se uma hierarquia de vários submenus. Isso acontece quando se entra no "Estado dos voos", fazendo com que surja um novo contentor de ícones, no qual se pode seleccionar o aeroporto pretendido, como se verifica na Figura 27. O exemplo apresentado inclui três aeroportos, cuja navegação é semelhante a qualquer dos menus apresentados anteriormente. No entanto, aqui tem-se apenas três ícones, não havendo nenhum quarto ícone "escondido" como no menu anterior. Por exemplo, caso se pressione a tecla para a direita, passa a ser o "Aeroporto de Faro" o seleccionado, aparecendo o "Aeroporto de Lisboa" à direita.


**Figura 27 Exemplo de simulação do sistema NiVo: menu de Estado dos voos**

No que respeita à interacção com o "Estados dos voos", quando se entra num determinado aeroporto, o que surge na simulação são *screenshots* obtidos previamente de um caso real. Para cada aeroporto tem-se dois casos possíveis, os de chegadas e partidas. A alternância entre um ou outro caso é realizada utilizando as teclas para a direita e para a esquerda do teclado.

Como exemplo, na Figura 28 e na Figura 29, tem-se as situações de chegadas e partidas do "Aeroporto do Porto".


**Figura 28 Exemplo de simulação do sistema NiVo: menu de Estado dos voos (Arrivals)**


**Figura 29 Exemplo de simulação do sistema NiVo: menu de Estado dos voos (Departures)**

# 5. NIVO COMPOSER

A composição de temas para o sistema NiVo pressupõe um processo minucioso, tendo-se já verificado ao longo deste relatório alguns dos procedimentos necessários. Entre estes, encontram-se, por exemplo, a introdução de imagens e ícones, os quais devem possuir determinados tamanhos e resoluções, bem como um nome particular pelo qual será referido na criação do tema.

De maneira a facilitar o trabalho do designer neste processo de criação, o NiVo Composer surge como uma ferramenta de grande utilidade. Entre as potencialidades oferecidas, é importante realçar a capacidade de validar os temas, por exemplo, verificando a conformidade dos parâmetros das imagens e ícones a introduzir focando, particularmente, como já referido, nas suas resoluções e nomes. Todo este processo de criação de temas utilizando o NiVo Composer vem prevenir possíveis falhas que pudessem existir, diminuindo o erro humano.

No que diz respeito a desenvolvimentos efectuados para esta aplicação, criou-se um protótipo da sua interface gráfica, tendo-se mesmo implementado alguns dos métodos de escolha e validação de imagens para o tema a criar, embora não tenha chegado ao estado final. Pelo contrário, no caso do simulador do NiVo descrito no capítulo anterior, chegou-se mesmo a um estado comercial, tendo sido a aplicação à qual se deu o foco principal.

Tal como no simulador do NiVo, no NiVo Composer utilizou-se também a tecnologia Flash, gerando-se o ficheiro "NiVo_Composer.swf". Ao longo deste capítulo, serão abordadas as características desenvolvidas para esta última aplicação.

Na Figura 30 apresenta-se a interface gráfica, na qual se encontram duas áreas distintas. Na parte superior, tem-se um menu, onde se incluem vários componentes (botões, *combo boxes*, *check boxes*). Imediatamente abaixo do menu, tem-se a área de visualização do tema a ser construído, na qual vão surgindo as imagens para se realizar o seu *upload*.

A criação do tema realiza-se por etapas, havendo uma ordem a seguir. Como se pode verificar, inicialmente os únicos componentes com os quais é permitido interagir são: o botão de instruções, para se obter uma descrição dos procedimentos a seguir para a criação de um tema utilizando o NiVo Composer; e uma *combo box* para uma selecção inicial da resolução de TV à qual se destina. Só depois de ser realizada esta selecção, serão desbloqueados os botões correspondentes ao *upload* da imagem de *background*, e ao *reset* da aplicação, quando pretendido. Obviamente que o *background* é a base da criação de um tema, sendo as restantes imagens adicionadas sobre este.

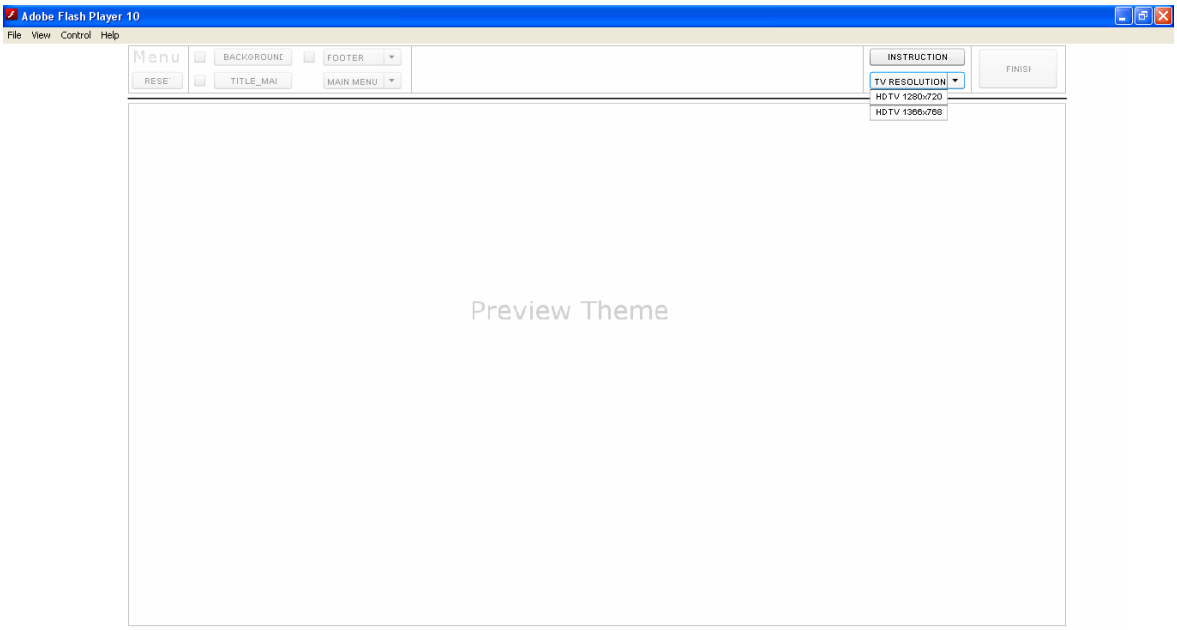


**Figura 30 NiVo Composer: Interface gráfica**

Como se pode verificar na Figura 31, quando se pressiona o botão "Background", surge uma nova janela para se proceder à escolha da imagem a adicionar como *background*. Nesta fase, tendo previamente sido seleccionada a resolução desejada para o tema, realiza-se a validação do cumprimento dessa escolha. Por exemplo, se a escolha passou pela resolução de 1280 por 720 *pixels*, apenas se permite o *upload* de imagens que cumpram esse parâmetro.

Este procedimento de validação é também utilizado nos restantes *uploads* de imagens para o tema, tendo em conta as dimensões permitidas em cada caso. Neste sentido, pode consultar-se na Tabela 1, apresentada na secção 2.4.2, os valores das dimensões associadas a cada componente da interface de um tema.

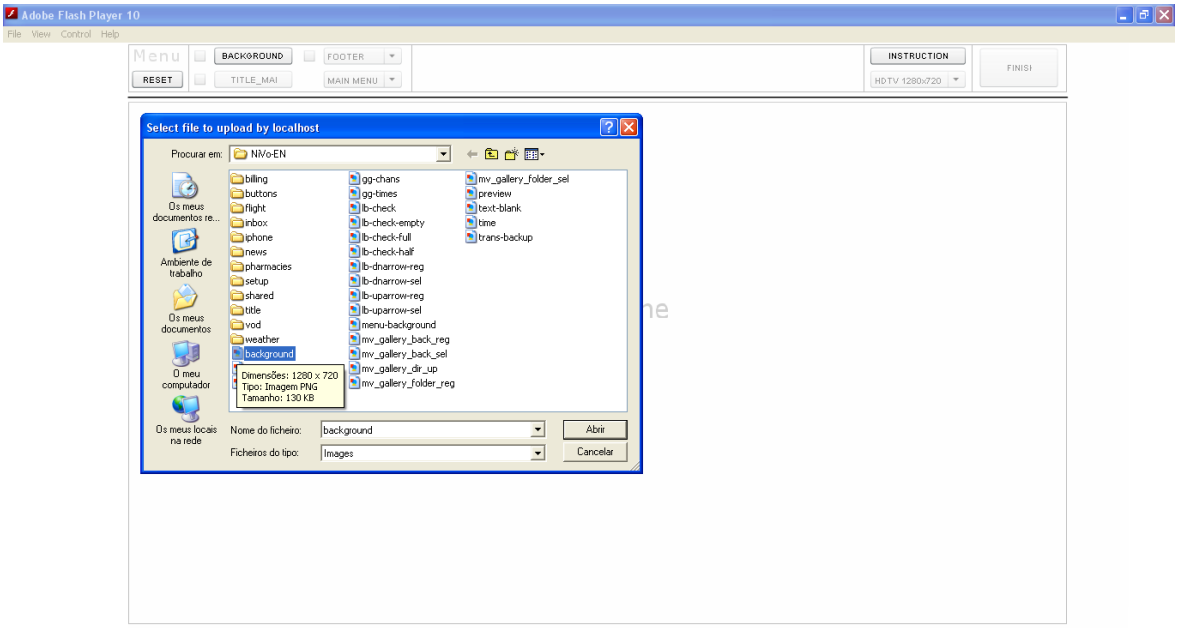


**Figura 31 NiVo Composer: Selecção de imagens**

No caso de a imagem escolhida não possuir a resolução necessária, apresenta-se uma mensagem de aviso, na qual se refere as únicas dimensões permitidas para a imagem em causa. Esta situação encontra-se exposta na Figura 32.

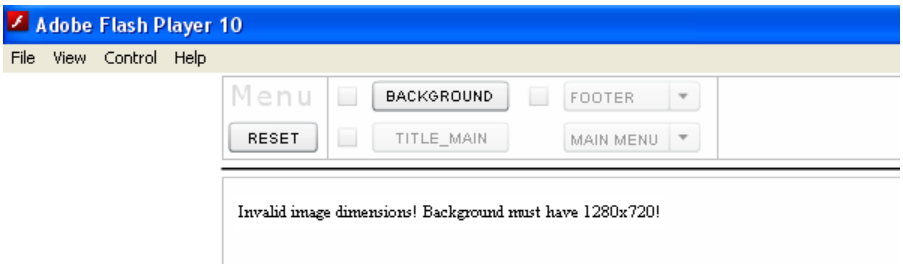


**Figura 32 NiVo Composer: Validação das dimensões de imagens**

Por outro lado, sendo o *upload* da imagem do *background* realizado com sucesso, passa-se a uma outra fase do processo de composição do tema. A imagem é apresentada na área de visualização, passando a estar seleccionada a *check box* correspondente. Além disso, pode proceder-se ao *upload* dos restantes componentes da interface (cabeçalhos, imagens de data/hora, etc.), visto que são desbloqueados os botões e *combo boxes* com essa finalidade.

Na Figura 33, apresenta-se a fase do processo em que se realizou com sucesso a introdução da imagem do cabeçalho para o menu principal, bem como as imagens para o *footer* (data/hora). Como se pode verificar, encontram-se seleccionadas todas as *check boxes* associadas às últimas imagens referidas, validando o *upload* das mesmas.

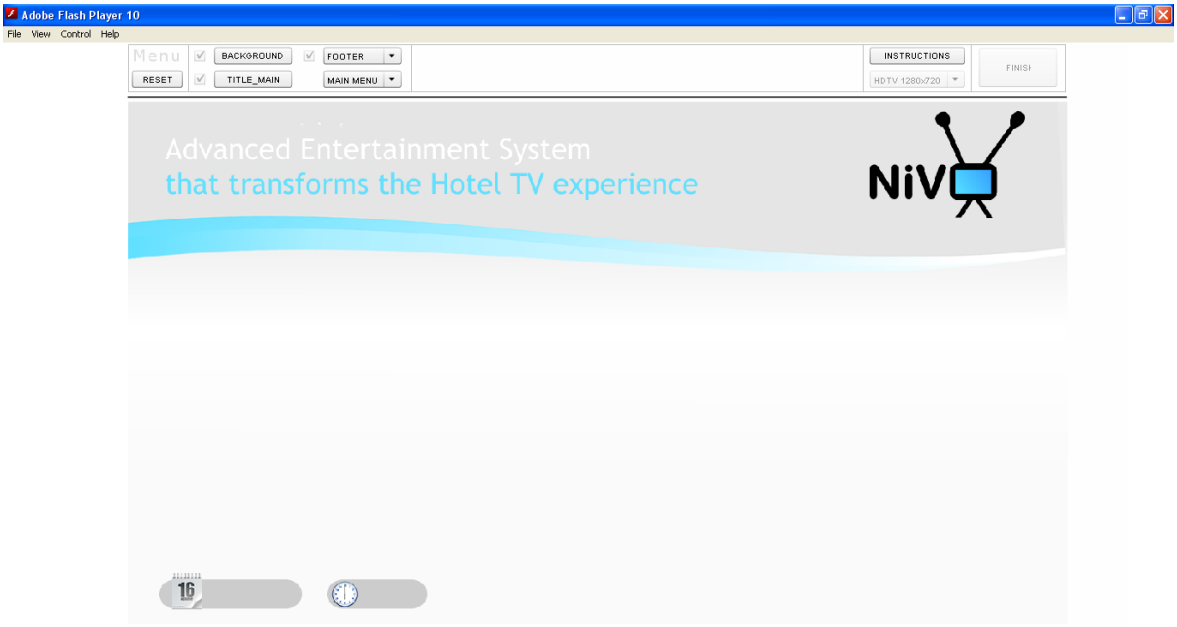


**Figura 33 NiVo Composer: Upload com sucesso de várias imagens**

No caso particular do *footer* (Figura 34), a sua *check box* só fica seleccionada quando se realiza com sucesso o *upload* de ambas as imagens.

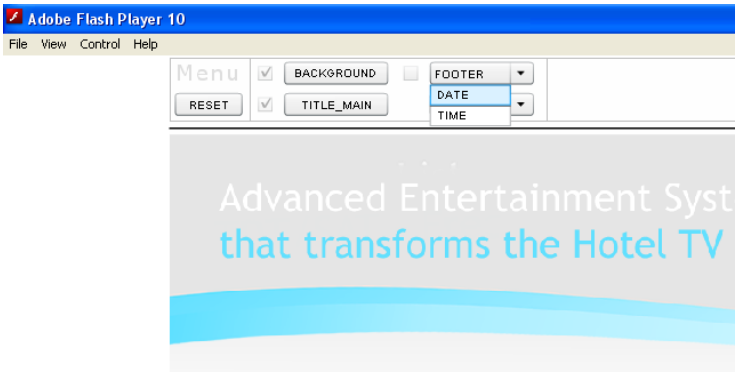


**Figura 34 NiVo Composer: Upload das imagens para o *footer* (data/hora)**

Posteriormente, na *combo box* "Main Menu", pode optar-se pelo tipo de menu que se pretende adicionar. Como se observa na Figura 35, fornecem-se duas hipóteses: um menu que irá possuir submenu; ou, pelo contrário, um menu que não irá possuir submenu. Em relação ao primeiro, não foi implementada a sua acção no NiVo Composer, tendo-se

apenas desenvolvimentos para o segundo tipo de menus. No entanto, o método é semelhante, devendo apenas ter-se em consideração a hierarquia.

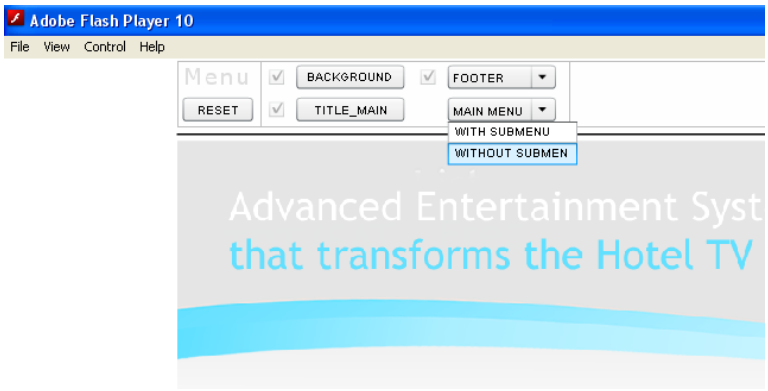


**Figura 35 NiVo Composer: Selecção do tipo de menu a adicionar**

Referente ao caso em que se opta pela adição de um menu que não possua submenu, apresenta-se na Figura 36 o resultado dessa selecção. Verifica-se que surgem mais seis botões: um para introdução do nome do menu, três para fazer o *upload* de imagens (cabeçalho e ícones *on*/*off*) e os últimos dois para finalizar ou cancelar a introdução desse novo menu.

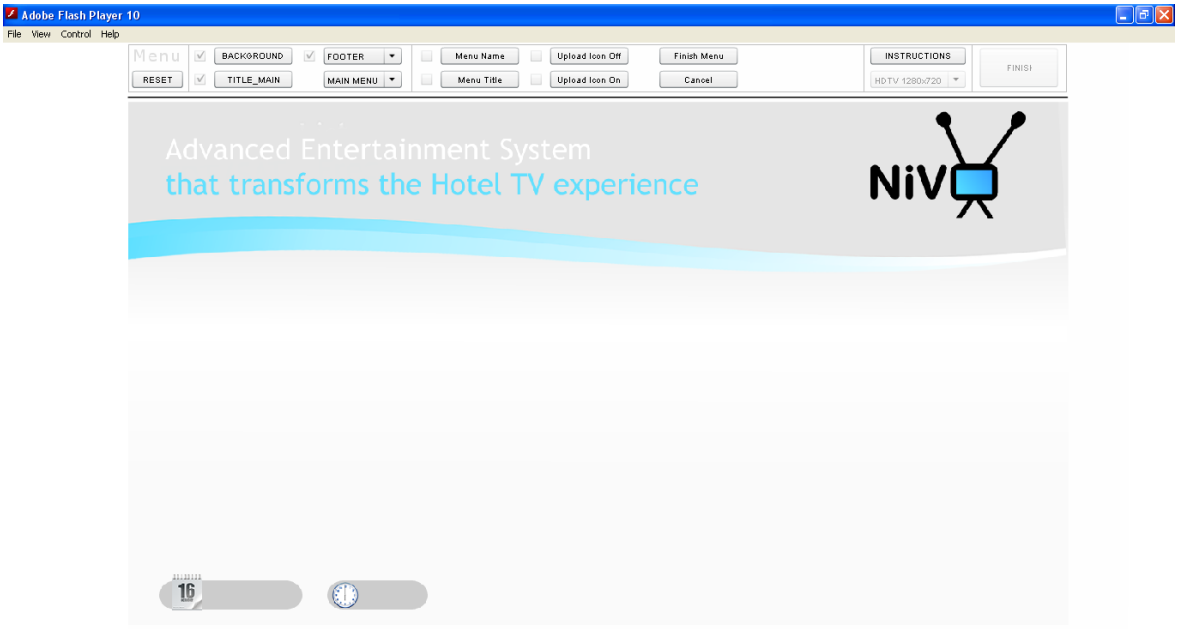


**Figura 36 NiVo Composer: Adição de um novo menu ao tema**

O objectivo final passa por só permitir a finalização do menu quando as quatro *check boxes* presentes ficarem seleccionadas. Ou seja, deve em primeiro lugar introduzir-se um nome válido e realizar o *upload* de imagens com as dimensões apropriadas para cada caso. No entanto, até esta fase, não foram criados os métodos para atender às acções dos botões

correspondentes ao *upload* dos ícones *on* e *off*. Desta forma, para poder testar-se os desenvolvimentos efectuados (validação do nome introduzido e do *upload* da imagem para o cabeçalho), permite-se a finalização do menu apenas com estas duas *check boxes* seleccionadas, como se apresenta na Figura 37.


**Figura 37 NiVo Composer: Validação e finalização de um novo menu para o tema**

Após a finalização de um novo menu, surge uma nova *combo box*, denominada "Menus Added". Na Figura 38, verifica-se a sua localização, no lado direito da aplicação, ao lado do botão "Instructions". Neste componente, apresenta-se uma lista de todos os menus adicionados. Através da selecção de cada um, pode conferir-se o seu aspecto, na área de visualização. Neste caso, é demonstrado o menu de TV criado no exemplo da Figura 37.

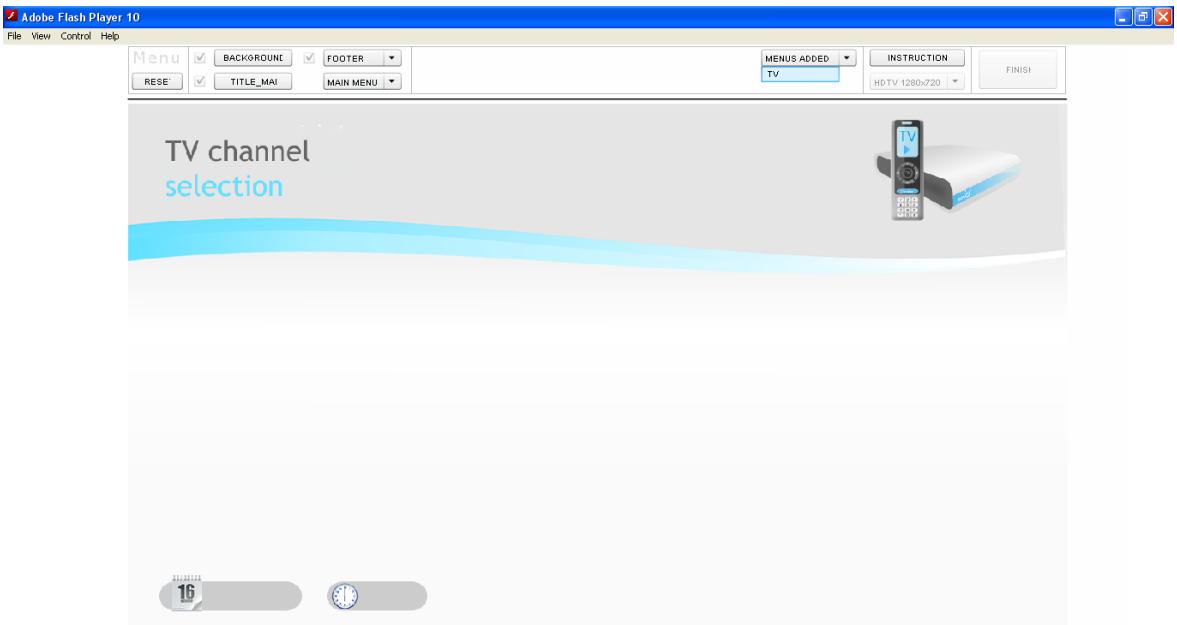


**Figura 38 NiVo Composer: Visualização de um novo menu criado**

Além de todas as características do NiVo Composer apresentadas, tem-se ainda disponível um botão de *reset*. Como foi referido, este fica disponível a partir do momento que se realiza o primeiro passo (selecção da resolução). Posteriormente, este botão pode ser accionado a qualquer momento. É importante realçar que, tal como se observa no exemplo da Figura 39, é solicitada a confirmação do desejo de reiniciar o processo de criação do tema.

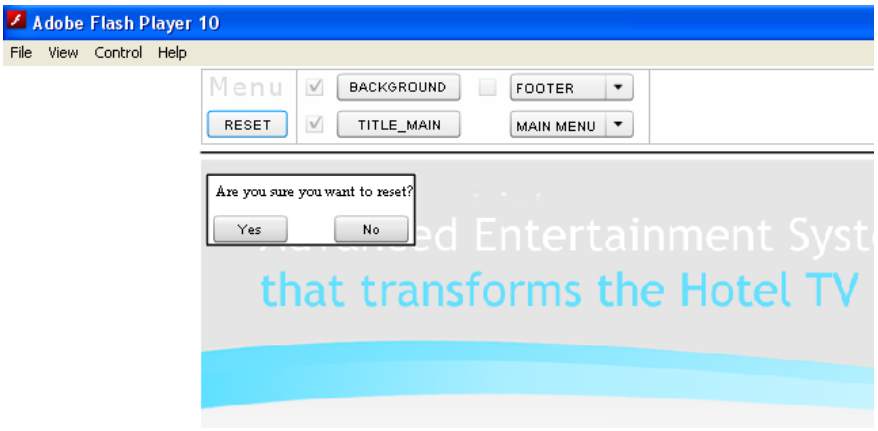


**Figura 39 NiVo Composer: Acção do botão "Reset"**

# 6. CONCLUSÕES

Ao longo deste relatório pretendeu-se resumir e demonstrar o processo de desenvolvimento e os estudos efectuados para a criação de uma aplicação simuladora do sistema multimédia NiVo. Além desta aplicação, menciona-se também uma outra aplicação desenvolvida, o NiVo Composer, que por motivos de prioridades, não alcançou um estado comercial, ao contrário da primeira referida. Desta forma, os objectivos foram cumpridos quase na totalidade, visto que, os mais importantes foram atingidos com sucesso.

A finalidade deste projecto consistiu em facilitar o processo de desenvolvimento de temas gráficos para o sistema NiVo, incidindo principalmente na criação do simulador. Neste sentido, implementaram-se as funcionalidades necessárias para esta aplicação, de modo que o seu aspecto e interactividade fossem o mais próximo possível da realidade do sistema, permitindo a realização de demonstrações do mesmo.

Para tal, muito contribuíram os estudos e implementações das capacidades da tecnologia Flash, utilizando-se a linguagem de programação ActionScript. O balanço relativo à adaptação a esta linguagem foi positivo na medida em que consiste numa linguagem de programação orientada a objectos, à semelhança da linguagem Java, sendo esta última a estudada em contexto académico. No entanto, existem muitas diferenças entre elas, tendo sido muito úteis os tutoriais utilizados para aprendizagem de características particulares do ActionScript 3.0 [15].

Uma vez concluído o desenvolvimento, no que diz respeito a alguns aspectos que não foram possíveis de realizar, mencionam-se por exemplo, as reproduções reais das emissões

dos canais de TV e rádio (simulados com vídeos e áudio demonstrativos), bem como as actualizações reais aos menus dinâmicos, como são os casos do estado do tempo e das notícias (simulados através de *screenshots* previamente obtidos).

Além disso, existem algumas considerações a ter em conta para aperfeiçoamentos futuros. Desde logo, destaca-se o desenvolvimento integral do menu de VOD, permitindo a inclusão de novos conteúdos (categorias/filmes), bem como a introdução da funcionalidade de controlo parental. Uma outra implementação interessante, seria haver a possibilidade de seleccionar o tema a simular, sem a necessidade de fechar a aplicação simuladora e proceder à respectiva alteração dos ficheiros do tema. Noutro contexto, relacionado com a aplicação de criação de temas, NiVo Composer, tem-se como trabalho futuro a conclusão do seu desenvolvimento, assim como, toda a sua validação.

Na vertente pessoal, este estágio permitiu desenvolver competências e adquirir experiência relevante em vários níveis. Desde logo, tem-se a aprendizagem e integração em ambiente empresarial, associado ao desenvolvimento de *software*. Além disso, é também importante realçar a aquisição de novos conhecimentos a nível científico, como o caso da aprendizagem de uma nova linguagem de programação, o ActionScript, bem como a obtenção de experiência na aplicação de conceitos, como por exemplo, o XML.

Neste sentido, o estágio revelou-se uma mais-valia futura no âmbito de novos projectos, tendo-me potenciado novas capacidades para encarar com mais confiança o mundo do trabalho.

# *Referências Documentais*


[1] *Nonius Software*. (s.d.). Obtido em 10 de Junho de 2011, de Nonius Software: http://www.noniussoftware.com

[2] *Sistema NiVo*. (s.d.). Obtido em 10 de Junho de 2011, de Nonius Software: http://www.noniussoftware.com/web2/index.php?option=com_content&view=article&id=237&Itemid=135&lang=pt

[3] *NiVo Datasheet.* (s.d.). Obtido em 10 de Junho de 2011, de Nonius Software: http://www.noniussoftware.com/pdfs/3012-BR-PT-10-100104%20NIVO%20ADVANCED%20ENTERTAINMENT%20SYSTEM.pdf

[4] *Brochura NiVo.* (s.d.). Obtido em 10 de Junho de 2011, de Nonius Software: http://www.noniussoftware.com/pdfs/3097-BR-PT-10-110224%20NIVO-SOLUCAO_DE_ENTRETENIMENTO_MULTIMEDIA_E_ACESSO_A_INTERNET.pdf

[5] *Hospitality Multimedia Solutions*. (2008). Obtido em 10 de Junho de 2011, de Neos Interactive: http://www.neosinteractive.com

[6] *Neos Interactive Ltd. - General Information.* (15 de Junho de 2010). Obtido em 10 de Junho de 2011, de Neos Interactive: http://www.neosinteractive.com/user_downloads/neos.pdf

[7] *Digital Entertainment Solution*. (s.d.). Obtido em 10 de Junho de 2011, de eona: http://www.eona.com/en/des.php

[8] *Digital Entertainment Solution - Presentation.* (s.d.). Obtido em 10 de Junho de 2011, de Freegate: http://freegate.net.au/EONA-09-DES-EN.pdf

[9] *Netris IPTV Solution*. (s.d.). Obtido em 10 de Junho de 2011, de Netris: http://www.netris.ru/en/sectors/digital-television/iptv-solution.html

[10] *Netris IPTV Solution - Presentation.* (s.d.). Obtido em 10 de Junho de 2011, de Netris: http://www.netris.ru/downloads/ads/Netris_IPTV_Solution.pdf

[11] Mook, C. (2007). *Essential ActionScript 3.0* (1ª Edição ed.). (S. Weiss, Ed.) O'Reilly Media, Inc.

[12] Shupe, R., & Rosser, Z. (2010). *Learning ActionScript 3.0* (2ª Edição ed.). (M. Treseler, Ed.) O'Reilly Media, Inc.

[13] *Beginner's Guide to FlashDevelop - Basixx*. (s.d.). Obtido em 10 de Junho de 2011, de Activetuts+: http://active.tutsplus.com/tutorials/beginners-guide-to-flashdevelop-intro-basix

[14] *FlashDevelop Wikidocs*. (s.d.). Obtido em 10 de Junho de 2011, de FlashDevelop: http://www.flashdevelop.org/wikidocs

[15] *Republic of code Tutorials*. (s.d.). Obtido em 10 de Junho de 2011, de Republic of code: http://www.republicofcode.com/tutorials/flash

[16] *Adobe AIR and Adobe Flash Player Technologies*. (s.d.). Obtido em 10 de Junho de 2011, de Adobe: http://labs.adobe.com/technologies/flashplatformruntimes

# Anexo A. Estrutura dos temas do sistema NiVo

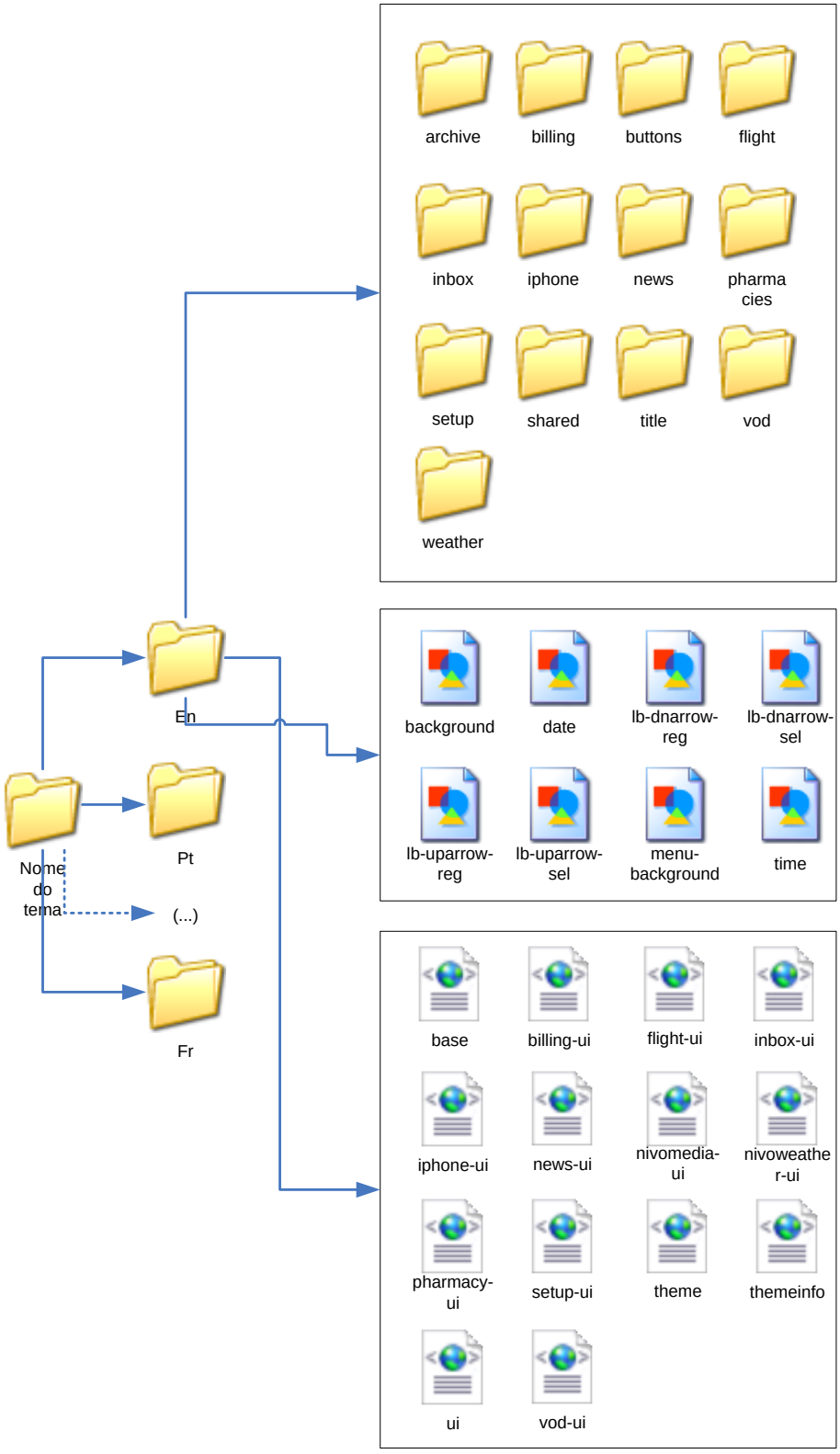


**Figura 40 Estrutura dos temas**

# Anexo B. Descrição do ficheiro ActionScript (Main.as)

- **PROCESSAMENTO DOS VÁRIOS XML**

Tal como se mencionou na secção 4.1.2, para a criação do simulador, os ficheiros XML necessários, pertencentes aos temas do sistema NiVo, são: o "base.xml", o qual contém as características para o *background*, para a data e para a hora, incluindo os caminhos para as respectivas imagens; o "nivomedia-ui.xml", onde se encontram os caminhos para os cabeçalhos dos menus de TV e de rádio; e o "theme.xml", sendo este o que contém os caminhos para todos os restantes cabeçalhos dos menus, bem como para todos os ícones (*on* e *off*), que como já foi dito, não são apresentados com a estrutura hierárquica dos menus a exibir. Além destes, tem-se portanto associado aos ficheiros do simulador NiVo, o "nivoTheme.xml", onde é estabelecida essa estrutura necessária.

Começando por abordar o processamento do ficheiro "base.xml", surgiu inicialmente um entrave, uma vez que este não possuía a definição de um espaço de nomes utilizado, não permitindo assim ao programa realizar a sua leitura. A solução passou pela criação directa deste mesmo XML, no próprio código ActionScript, obviamente já com a correcta definição do espaço de nomes (xmlns:size="localhost"), como se pode verificar no excerto de código apresentado de seguida, não se utilizando directamente os ficheiros "base.xml" dos pacotes dos temas.

```
function processXML_base():void
{
 var myXML_base:XML =
       <mythuitheme xmlns:size="localhost">

       <window name="backgroundwindow">

             <imagetype name="backimg">
             <filename>background.png</filename>
             </imagetype>
(…)
       </window>

       </mythuitheme>

 my_background_base = myXML_base.window.(@name ==
"backgroundwindow").imagetype.(@name ==
"backimg").filename;
```

```
 my_datetime_base = myXML_base.window.(@name ==
"backgroundwindow").imagetype.filename;

}
```

Em primeiro lugar, criou-se uma instância da classe *XML()*, à qual se associou o XML pretendido. De seguida, utilizando-se variáveis do tipo XMLList (objectos que podem representar um ou mais elementos XML) previamente definidas ("my_background_base" e "my_datetime_base"), extrai-se e processa-se as informações úteis deste ficheiro, ou seja, as localizações das imagens de *background*, data e hora.

Passando para outro ficheiro a ser processado, tem-se o "theme.xml". Neste foi igualmente necessário ter-se em consideração alguns aspectos especiais. Aqui, a situação em causa relaciona-se com o facto de os ficheiros "theme.xml" dos temas do NiVo conterem repetições de *tags*, exactamente com o mesmo nome e conteúdo, tal como se apresenta de seguida num excerto de código exemplificativo.

```
<image mode="GAMES">title/title_GAMES.png</image>
<image
mode="INFO_SERVICES">title/title_INFO_SERVICES.png</image>
<image
mode="INFO_FLIGHT">title/title_INFO_FLIGHT.png</image>
<image mode="MAIN">title/title_MAIN.png</image>
<image mode="GAMES">title/title_GAMES.png</image>
<image
mode="INFO_FLIGHT">title/title_INFO_FLIGHT.png</image>
<image
mode="INFO_SERVICES">title/title_INFO_SERVICES.png</image>
<image mode="MAIN">title/title_MAIN.png</image>
```

Para contornar esta situação, no código ActionScript, após ser feito o carregamento deste XML, elimina-se uma das repetições para cada caso existente. Na prática, define-se um XMLList para cada repetição, contendo assim dois elementos, dos quais um será eliminado, utilizando-se para isso o operador *delete*. Só depois, e utilizando-se outras variáveis do tipo XMLList previamente definidas para o efeito, extrai-se e processa-se as informações pretendidas deste ficheiro, ou seja, as localizações dos cabeçalhos dos menus, bem como dos respectivos ícones *on* e *off*. No excerto de código seguinte apresenta-se como este processo é realizado.

```
var myXML_theme:XML;
var myXML_themeLoader:URLLoader = new URLLoader();

myXML_themeLoader.load(new URLRequest("theme.xml"));
myXML_themeLoader.addEventListener(Event.COMPLETE,
processXML_theme);
```

```
function processXML_theme(e:Event):void
{
	myXML_theme = new XML(e.target.data);

	//ELIMINAR REPETIÇÕES DE NÓS NO XML theme.xml
	var myList:XMLList =
myXML_theme.titles.image.(@mode == "GAMES");
	var myList1:XMLList =
myXML_theme.titles.image.(@mode == "INFO_FLIGHT");
	var myList2:XMLList =
myXML_theme.titles.image.(@mode == "INFO_SERVICES");
	var myList3:XMLList =
myXML_theme.titles.image.(@mode == "MAIN");

(…)

	delete myList[1];
	delete myList1[1];
	delete myList2[1];
	delete myList3[1];

(…)

	my_images_theme = myXML_theme.buttondef;
	my_header_theme = myXML_theme.titles;
}
```

No que diz respeito ao "nivomedia-ui.xml", o seu processamento é executado de forma normal. Extraem-se as informações pretendidas para duas variáveis do tipo XMLList, tendo realce para o simulador apenas as localizações das imagens para os cabeçalhos dos menus de TV e de rádio, como se mostra de seguida no excerto de código.

```
var myXML_nivomedia:XML;
var myXML_nivomediaLoader:URLLoader = new URLLoader();
myXML_nivomediaLoader.load(new URLRequest("nivomedia-
ui.xml"));
 myXML_nivomediaLoader.addEventListener(Event.COMPLETE,
processXML_nivomedia);

function processXML_nivomedia(e:Event):void
{
	myXML_nivomedia = new XML(e.target.data);

	my_header_tv = myXML_nivomedia.window.(@name ==
"tv_gallery").container.(@name == "text").image.(@name ==
"title").filename;

	my_header_radio = myXML_nivomedia.window.(@name ==
"radio_gallery").container.(@name == "text").image.(@name
== "title").filename;

}
```

Por último, no que diz respeito ao processamento de ficheiros XML, tem-se o "nivoTheme.xml". Obtém-se assim toda a estrutura a ser apresentada no simulador, bem como as localizações dos *screenshots* para os menus que não possuem submenus, bem como dos vídeos e dos jogos. De seguida, tem-se um excerto do código responsável por este processamento.

```
var myXML:XML;
var myXMLLoader:URLLoader = new URLLoader();
myXMLLoader.load(new URLRequest("nivoTheme.xml"));
myXMLLoader.addEventListener(Event.COMPLETE, processXML);

function processXML(e:Event):void
{
	myXML = new XML(e.target.data);

	my_images = myXML.buttons.buttondef;
	my_total = my_images.length();
	my_presentation = myXML.buttons.buttondef.(@name ==
"PRESENTATION").fullimage;
	my_pharmacy = myXML.buttons.buttondef.(@name ==
"INFO_SERVICES").buttoninfo.(@name ==
"PHARMACIES").fullimage;

(…)

	my_total_games = my_images.(@name ==
"GAMES").buttongame.length();
	my_total_channels = my_images.(@name ==
"TV").list.image.length();
	my_total_radio = my_images.(@name ==
"RADIO").list.image.length();

(…)

}
```

Como se pode verificar, além da definição da estrutura dos menus, obtém-se também a quantidade de ícones associados a cada um deles, utilizando-se para isso o método *length()* da classe XML. Por exemplo, a variável "my_total", do tipo *Number*, diz respeito ao número de ícones existentes para o menu principal.

- **CARREGAMENTO DAS VARIÁVEIS DO FICHEIRO DE TEXTO**

Em relação ao carregamento das variáveis a utilizar no programa, tem-se o próximo excerto de código, como exemplo deste processo.

Como se pode verificar, é igualmente criada uma instância da classe *URLLoader*, utilizando-se o método *load()* para carregar o ficheiro de texto "variables.txt". Quando o

processo de carregamento é concluído, as variáveis que se pretendia obter são atribuídas às que foram definidas no código ActionScript com o objectivo de as receber. Neste caso, para mais fácil percepção do conteúdo associado a cada uma, definiram-se as variáveis contidas no ficheiro de texto exactamente com o mesmo nome das variáveis associadas, definidas no ficheiro ActionScript.

```
var loader:URLLoader = new URLLoader();
loader.dataFormat = URLLoaderDataFormat.VARIABLES;
loader.addEventListener(Event.COMPLETE, loading);
loader.load(new URLRequest("variables.txt"));

function loading (event:Event):void {
	velDeslizamento = loader.data.velDeslizamento;
	espacoEntreIcon = loader.data.espacoEntreIcon;
	espacoEntreIcon4 = loader.data.espacoEntreIcon4;
	espacoEntreChannels =
loader.data.espacoEntreChannels;
	espacoEntreChannels4 =
loader.data.espacoEntreChannels4;

(…)
	my_size = loader.data.my_size;
	my_bold = loader.data.my_bold;
	my_color = loader.data.my_color;

(…)

}
```

- **CRIAÇÃO DE *MOVIECLIPS* PARA COMPONENTES GRÁFICOS**

Para a criação de todo o aspecto gráfico do simulador, utilizam-se vários *MovieClips*. Obviamente que, em primeiro lugar, encontra-se a criação de um *MovieClip* para receber a imagem de *background* do tema, passando-se de seguida pelos casos dos cabeçalhos para os menus, bem como a criação dos próprios contentores de ícones para os respectivos menus a apresentar.

Como se verificará a seguir, utiliza-se a classe *Loader* para carregar os ficheiros externos de imagens, em tempo de execução do simulador Flash. Para tal, em cada caso, cria-se uma instância da classe e, de seguida, utiliza-se o seu método *load()* para carregar o ficheiro externo. Para se apresentar as imagens no ecrã, será também necessária, posteriormente, a utilização do método *addChild()*.

Começando por demonstrar a utilização da imagem de *background*, encontra-se no próximo excerto de código o método responsável pela criação e posicionamento do

*MovieClip* que irá conter a respectiva imagem. Como facilmente se analisa, este *MovieClip* é adicionado ao *stage*, sendo os valores das suas coordenadas proveniente das variáveis obtidas do ficheiro de texto externo.

```
function createBackgroundContainer():void
{
	background_mc = new MovieClip();
	background_mc.x = my_background_x;
	background_mc.y = my_background_y;
	stage.addChild(background_mc);
}
```

De seguida, tem-se a utilização da classe *Loader* para carregar a imagem de *background*. Em primeiro lugar, verifica-se se o XMLList ("my_background_base"), correspondente à localização da respectiva imagem, obtida pelo processamento do ficheiro "base.xml", contém ou não algum elemento. Posteriormente, regista-se o evento *Event.COMPLETE*, para adicionar a imagem ao *MovieClip* criado, o "background_mc", através do método *addChild()*, apenas quando a imagem for totalmente carregada.

```
function callBackground():void
{
	if (my_background_base.length() > 0)
	{
		var background_url = my_background_base;
		var background_loader:Loader = new Loader();
		background_loader.load(new
URLRequest(background_url));


 background_loader.contentLoaderInfo.addEventListener
(Event.COMPLETE, backgroundLoaded);
	}
}

function backgroundLoaded(e:Event):void
{
	var my_background:Loader = Loader(e.target.loader);
	background_mc.addChild(my_background);
}
```

Passada a demonstração da adição do *background* do tema, tem-se agora um caso de criação do cabeçalho para um menu. Neste caso, e visto ser realizado da mesma forma para todas as categorias de menus, aqui aborda-se apenas a forma de adição do cabeçalho do menu principal.

No excerto de código seguinte, verifica-se que, tal como no caso anterior de adição da imagem de *background*, para os cabeçalhos cria-se igualmente um *MovieClip* para conter a respectiva imagem. Da mesma forma, realiza-se o posicionamento do *MovieClip*, sendo

que também é adicionado ao *stage*. No entanto, no caso dos cabeçalhos, realiza-se ainda a definição da propriedade *visible* do respectivo *MovieClip*, com o valor *true* ou *false*. Neste caso, tratando-se do cabeçalho do menu principal, o seu valor inicial será *true*, visto ser o primeiro a ser apresentado. Por outro lado, a inicialização dos restantes submenus será com o valor *false*. Só posteriormente, mediante as interacções com o simulador, estes valores serão alterados de acordo com o caso que se verificar.

```
function createHeaderContainer():void
{
	header_mc = new MovieClip();
	header_mc.x = my_header_x;
	header_mc.y = my_header_y;
	stage.addChild(header_mc);
	header_mc.visible = true;
}
```

Depois disto, e similarmente ao que se realizou para a adição da imagem de *background*, utiliza-se a classe *Loader* para carregar a imagem do cabeçalho, como se verifica no excerto de código seguinte. De igual forma, verifica-se se o XMLList ("my_header_theme"), correspondente à localização da imagem em causa, contém ou não algum elemento. No entanto, neste caso, esta imagem está associada ao processamento do ficheiro "theme.xml", onde a referida variável XMLList inicialmente apenas acede ao elemento genérico "titles", do XML. Sendo assim, utilizando as propriedades da classe XML em ActionScript, acede-se particularmente ao elemento correspondente à localização do URL da imagem do cabeçalho do menu principal. Só depois se passa para a adição da respectiva imagem ao *MovieClip* criado, o "header_mc", através do método *addChild()*.

```
function callHeader():void
{
	if (my_header_theme.image.(@mode ==
"MAIN").length() > 0)
	{
		var header_url = my_header_theme.image.(@mode
== "MAIN");
		var header_loader:Loader = new Loader();
				header_loader.load(new
URLRequest(header_url));

 header_loader.contentLoaderInfo.addEventListener(Event.C
OMPLETE, headerLoader);
	}
}

function headerLoader(e:Event):void
{
	var my_header:Loader = Loader(e.target.loader);
	header_mc.addChild(my_header);
```

```
}
```

Por último, no caso dos contentores de ícones associados aos vários menus a apresentar, obviamente que o processo de desenvolvimento é também transversal a todos eles, sendo que aqui apenas será descrito o caso do menu principal.

Como se irá verificar, este processo não é tão linear como os de criação do *background* e dos cabeçalhos, havendo mais algumas variáveis e condicionantes. Desde logo, é necessário ter-se em consideração a quantidade de ícones que será adicionada ao contentor, para assim se definir a sua posição no *stage*. Esta situação está associada ao facto de estarmos na presença de *sliding menus*, ou seja, menus que rodam infinitamente, dando a ilusão de um anel, onde o ícone seleccionado será sempre o central. Sendo assim, só é permitida a visualização de um número ímpar de ícones no ecrã. Por exemplo, caso o menu possua cinco ou mais ícones para apresentar, o contentor terá uma determinada posição e mostrará sempre cinco ícones visíveis. Por outro lado, caso o menu tenha associados apenas três ou quatro ícones, o contentor terá uma posição diferente (mais central), e serão sempre visíveis apenas três ícones.

Relacionado com o contentor de ícones do menu principal, apresenta-se no excerto de código seguinte parte do método *createContainer()*, onde se encontram as características mencionadas anteriormente.

```
function createContainer():void
{
	container_mc = new MovieClip();

	if (my_total >= 5)
	{
		container_mc.x = my_x;
		container_mc.y = my_y;
	}

	else if (my_total == 4 || my_total == 3)
	{
		container_mc.x = my_x4;
		container_mc.y = my_y4;
	}

(…)
	stage.addChild(container_mc);
(…)
}
```

Além disso, ainda no método *createContainer()*, são criadas máscaras para limitar a visibilidade do contentor dos ícones, mediante a quantidade de ícones que se pretende

manter visíveis no ecrã. Sabendo-se as dimensões de cada ícone, em altura e largura, cria-se uma máscara com o formato rectangular desejado, ficando assim bem definido o contentor do menu. Este processo encontra-se exposto no excerto de código fornecido de seguida.

```
function createContainer():void
{

(…)
if (my_total >= 5)
{
       var maskWindow:MovieClip = new MovieClip();
       maskWindow.graphics.beginFill(0);

 maskWindow.graphics.drawRect(my_mainmenu_mask_x,
my_mainmenu_mask_y, my_mainmenu_mask_width,
my_mainmenu_mask_height);

       addChild(maskWindow);
       container_mc.mask = maskWindow;
       mc1Act = true;
}

else if (my_total == 4 || my_total == 3)
{
       var maskWindow:MovieClip = new MovieClip();
       maskWindow.graphics.beginFill(0);

 maskWindow.graphics.drawRect(my_mainmenu_mask_x4,
my_mainmenu_mask_y4, my_mainmenu_mask_width4,
my_mainmenu_mask_height4);

       addChild(maskWindow);
       container_mc.mask = maskWindow;
       mc1Act = true;
}

(…)
}
```

Tendo-se já criado o contentor necessário, procede-se ao carregamento dos ícones associados ao menu, utilizando-se para isso a classe *Loader*. Neste caso, visto que os URL de todos os ícones pertencentes ao tema estão definidos no "theme.xml", seleccionam-se apenas os elementos que tenham o mesmo atributo que os do "nivoTheme.xml" no nível da hierarquia correspondente ao menu principal. À medida que vão sendo carregados, são adicionados ao vector "images", que conterá todos os ícones.

```
function callIcones():void
{
       for (var i:Number = 0; i < my_total; i++)
       {
```

```
            vect_images[i] = i;

            var icone_url = my_images_theme.(@name ==
(my_images[i].@name)).image;
            var icone_loader:Loader = new Loader();
            icone_loader.load(new URLRequest(icone_url));
            images.push(icone_loader);

 icone_loader.contentLoaderInfo.addEventListener(Event.CO
MPLETE, iconeLoaded);

      }
}
```

Posteriormente, quando se conclui o carregamento, o evento definido irá accionar a função "iconeLoaded". Aqui utiliza-se a classe *Bitmap* para representar as imagens que foram carregadas com a classe *Loader*. Cada um dos *bitmaps* é adicionado ordenadamente ao contentor ("container_mc"), tendo-se em conta os valores da largura dos ícones e do espaçamento entre eles, sendo que estes valores foram obtidos através do ficheiro de texto "variables.txt".

Tal como no processo de criação do contentor, no excerto de código seguinte apresentam-se as situações em que se possui cinco ou mais ícones para adicionar ao contentor, bem como quando se possui apenas três ou quatro ícones. Neste caso, a diferença reside no espaçamento que é dado entre os ícones.

```
function iconeLoaded(e:Event):void
{
      if (my_total >= 5)
      {
            for (var i:int = 0; i < images.length; i++)
            {
                  var bm:Bitmap = new Bitmap();
                  bm = Bitmap(images[i].content);
                  bm.smoothing = true;
                  bm.x = i * (my_icone_width +
espacoEntreIcon);
                  container_mc.addChild(bm);
            }
      }
      else if (my_total == 4 || my_total == 3)
      {
            for (var i:int = 0; i < images.length; i++)
            {
                  var bm:Bitmap = new Bitmap();
                  bm = Bitmap(images[i].content);
                  bm.smoothing = true;
                  bm.x = i * (my_icone_width +
espacoEntreIcon4);
                  container_mc.addChild(bm);
```

```
                }
        }

(…)
}
```

Até este ponto analisou-se todo o processo de criação de um contentor de ícones para um determinado menu. No entanto, surge agora um aspecto importante, relacionado com o processo adoptado para criar o efeito de um menu infinito (*sliding menu*). A solução passou pela criação de dois contentores de ícones iguais, para cada menu que se pretenda implementar, havendo portanto também a necessidade de se criar os métodos *createContainer2()* e *callIcones2()*. Desta forma, é possível posicionar-se os dois contentores de ícones sempre lado a lado, "jogando" com as suas posições, e tendo em conta a utilização de uma classe específica (*TweenLite*) com a finalidade de criar o efeito de deslizamento. Todo este processo será abordado mais à frente.

Além disso, existe ainda mais um aspecto particular para completar o processo de criação de um menu. Como foi já referido, para cada ícone existe sempre uma imagem *on* e uma imagem *off*. Já foi também dito que o tipo de menu implementado é o chamado *sliding menu*, pelo que o ícone seleccionado (ícone *on*) será sempre o central. Sendo assim, aborda-se agora a criação do método que irá carregar todos os ícones *on* num vector, permitindo a utilização apenas do ícone pretendido. Por outras palavras, dependendo do deslizamento do menu, será sempre sobreposto o ícone *on* correspondente na posição central, ou seja, no ícone seleccionado.

Visto que o *sliding menu* é implementado com a utilização de dois contentores de ícones, é necessária também a criação de dois métodos para se obter o ícone *on* desejado. Dependendo do contentor que está a ser apresentado no ecrã, e com um ícone central que lhe pertence (contentor "activo"), poderá invocar-se um de dois métodos: o *callIconesOn()* ou o *callIconesOn2()*. Sendo estes dois métodos similares, no seguinte excerto de código apresenta-se apenas o primeiro.

```
function callIconesOn():void
{
       for (var i:Number = 0; i < my_total; i++)
       {
            var iconeOn_url = my_images_theme.(@name ==
(my_images[i].@name)).activeimage;
            var iconeOn_loader:Loader = new Loader;
            iconeOn_loader.load(new
URLRequest(iconeOn_url));
            imagesOn.push(iconeOn_loader);
```

```
        }

        imgSelect_mc.x = my_imgSelect_x;
        imgSelect_mc.y = my_imgSelect_y;
        stage.addChild(imgSelect_mc);

        imgSelect_mc.addChild(imagesOn[selectedBtn]);
}
```

Como se pode verificar, inicialmente é realizado o carregamento dos ícones *on* correspondentes ao menu em causa, sendo ordenadamente adicionados ao vector "imagesOn". De seguida, surge o *MovieClip* ao qual irá ser adicionada a imagem do ícone *on* em cada caso. Este é posicionado no *stage*, sendo as coordenadas obtidas do ficheiro de texto "variables.txt". Obviamente que estas coordenadas são definidas para que o *MovieClip* fique sobreposto ao ícone central do contentor de ícones. Para se saber qual o ícone *on* a colocar, é importante a utilização da variável "selectedBtn". Esta última é do tipo *Number*, e contém sempre o valor do ícone central na ordem do menu actual. Inicialmente, para o caso de menus com cinco ou mais ícones, a variável "selectedBtn" possui o valor dois, correspondendo ao terceiro ícone (ícone central), visto o primeiro ser considerado com o valor zero. Obviamente que, por exemplo, no caso de menus com quatro ícones, o valor inicial da variável "selectedBtn" é 1, uma vez que o ícone central corresponde ao segundo existente.

Por fim, no código apresentado de seguida, faz-se ainda alusão a um caso especial, referente aos ícones seleccionados para os menus dos canais de TV e de rádio. Estes são os únicos ícones que não possuem uma imagem para o estado *on* e outra para o estado *off*. Neste caso, a solução para dar realce ao ícone seleccionado, passa pela adição de um *background* para o respectivo ícone, o *MovieClip* "imgSelect_chann_mc". Este *background* para os ícones dos canais mantém-se fixo na posição central do respectivo menu, cujas coordenadas também são obtidas do ficheiro de texto "variables.txt". Como é lógico, este *MovieClip* apenas terá a sua propriedade *visible* definida como *true*, quando se entra num dos menus ao qual estará associado, ou seja, o menu de TV ou o menu de rádio.

```
function callChannOn():void
{
        var channOn_url = my_channels_back.image;
        var channOn_loader:Loader = new Loader();
        channOn_loader.load(new URLRequest(channOn_url));

        imgSelect_chann_mc.x = my_imgSelect_chann_x;
        imgSelect_chann_mc.y = my_imgSelect_chann_y;
        stage.addChild(imgSelect_chann_mc);
```

```
        imgSelect_chann_mc.addChild(channOn_loader);
        imgSelect_chann_mc.visible = false;
 }
```

- **ACÇÕES DO TECLADO PARA CRIAR A INTERACTIVIDADE**

A interactividade da aplicação é realizada através da utilização do teclado. Para tal, associa-se um evento de atendimento, que verifica qual a tecla pressionada e realiza as acções definidas para essa tecla. Este processo é demonstrado no excerto de código seguinte:

```
stage.addEventListener(KeyboardEvent.KEY_DOWN,
keyHandler);
function keyHandler(evt:KeyboardEvent):void
{
        stage.removeEventListener(KeyboardEvent.KEY_DOWN,
keyHandler);

        switch (evt.keyCode)
        {
              case Keyboard.LEFT:
                    (…)
                    break;
              case Keyboard.RIGHT:
                    (…)
                    break;
              case Keyboard.DOWN:
                    (…)
                    break;
              case Keyboard.UP:
                    (…)
                    break;
              case Keyboard.ENTER:
                    (…)
                    break;
              case Keyboard.ESCAPE:
                    (…)
                    break;
              default:
                    (…)
        }
}
```

Como se pode verificar, as únicas teclas às quais se associam acções, são as setas direccionais do teclado, o *enter* e o *escape*.

No que diz respeito à tecla *enter*, é utilizada para se exibir o conteúdo do ícone que for seleccionado. No caso de se estar perante a entrada num submenu, tem-se em consideração a propriedade *visible* da classe *MovieClip*, para tornar-se visíveis os ícones e imagens pretendidos para o menu em causa, tornando-se invisíveis os ícones e imagens do menu

anterior. No entanto, há casos como os de canais TV, rádios e jogos, em que é iniciada a sua reprodução aquando da acção da tecla *enter*.

A tecla *escape* é utilizada para se voltar a menus anteriores. À semelhança do que se sucede com as acções da tecla *enter*, faz-se uso da propriedade *visible* dos *MovieClips*, para tornar-se visíveis os menus anteriores, mediante o caso em que se encontra. Além disso, no caso em que se encontram em reprodução canais de TV, rádios ou mesmo jogos, a acção da tecla *escape* irá parar a sua reprodução e regressar ao menu.

Por último, em relação às direcções, utilizam-se as teclas para a direita e para a esquerda com a finalidade de percorrer-se os contentores de ícones dos menus, ao passo que as teclas para cima e para baixo são utilizadas para transitar entre categorias de filmes no menu de VOD.

De seguida, faz-se uma descrição dos procedimentos utilizados para ser possível criar-se o efeito de deslizamento necessário nos menus, sendo estes procedimentos transversais para qualquer um deles.

No excerto de código apresenta-se uma situação para quando se pressiona a tecla para a esquerda, quando se está perante o menu principal.

```
case Keyboard.LEFT:

if (container_mc.visible == true) //MAIN MENU
{
(…)
       if (mc1Act == true)
       {
       (…)
             if (selectedBtn == 2 && tamanhoVector >= 5)
             {
                   container_mc2.x = - tamanhoVector *
(my_icone_width + espacoEntreIcone);
                   myXPos2 = -tamanhoVector;
                   selectedBtn = selectedBtn - 1;
                   myXPos2 += 1;
                   myXPos += 1;
                   TweenLite.to(container_mc,
velDeslizamento, { x: myXPos * (my_icone_width +
espacoEntreIcone), ease:Strong.easeOut } );
                   TweenLite.to(container_mc2,
velDeslizamento, { x: myXPos2 * (my_icone_width +
espacoEntreIcone), ease:Strong.easeOut } );

 imgSelect_mc.removeChild(imagesOn[selectedBtn + 1]);
```

```
                    callIconesOn();
                    imgSelect_mc.visible = true;
                }
            (…)
            }
(…)
}
```

Como se observa neste excerto, tem-se inicialmente uma variável do tipo booleano ("mc1Act"), que verifica se é o primeiro contentor de ícones que se encontra activo. Isto porque, como foi referido anteriormente, o *sliding menu* é implementado com a utilização de dois contentores de ícones. A este respeito, entende-se que um contentor está activo quando o ícone central/activo actual lhe pertence.

Posteriormente, verifica-se qual o ícone que se encontra seleccionado, sendo que, neste caso, corresponde ao terceiro ícone da lista, uma vez que a variável "selectedBtn" tem o valor dois. Sendo assim, visto haver só mais dois ícones à esquerda deste, correspondentes ao contentor activo, e como o seu deslizamento será para a direita (tecla para a esquerda pressionada), deve de seguida posicionar-se o segundo contentor de forma que o seu último ícone surja na primeira posição do ecrã, aquando desse mesmo deslizamento. Desta forma, consegue-se o desejado efeito de circularidade do menu.

O referido deslizamento é obtido através da utilização da classe *TweenLite*. Esta classe tem como finalidade a criação de animações, por exemplo, em objectos ou *MovieClips*.

Neste exemplo, utiliza-se o seu método *to()*. Este método é utilizado para efectuar interpolações, correspondendo o seu primeiro argumento ao alvo da interpolação. Como se verifica no excerto de código apresentado, aplica-se o método a cada um dos contentores de ícones, sendo em cada caso o respectivo *MovieClip* o alvo da interpolação.

O segundo argumento do método diz respeito ao tempo, em segundos, que dura a interpolação. Neste caso, utiliza-se a variável "velDeslizamento", que é recebida do ficheiro de texto externo ("variables.txt"), contendo o valor zero.

Por fim, no terceiro argumento, encontra-se a posição para a qual se desloca o *MovieClip*, correspondendo à deslocação da largura de um ícone, adicionando um espaçamento.