SISTEMA DE TELEVISÃO INTERACTIVA PARA O MERCADO HOTELEIRO

Ricardo Miguel Gouveia Pereira

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto
2012

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: Ricardo Miguel Gouveia Pereira, Nº 1070297, 1070297@isep.ipp.pt
Orientação científica: Paula Maria Marques Moura Viana, pmv@isep.ipp.pt

Empresa: Nonius Software

Supervisão: Raul 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
25 de Novembro de 2012

Para a minha família, porque sem eles nada disto teria sido possível.

Agradecimentos

Agradeço em primeiro lugar aos meus pais pois sem a sua dedicação nada disto teria sido possível.

Agradeço também à Nonius Software pelo estágio que me proporcionou, contribuindo para o meu desenvolvimento a nível técnico, pessoal e profissional. Agradeço ao meu supervisor na empresa, Engenheiro Raul Carvalho, pela disponibilidade e pelas oportunidades que foram dadas.

Agradeço à minha orientadora de estágio, Engenheira Paula Viana por toda a dedicação e apoio prestado ao longo da elaboração da tese.

Agradeço a todos os colaboradores do departamento de investigação e desenvolvimento da Nonius, em especial ao Nuno Mota, Luís Pimenta e ao Constantino Antunes pelos conhecimentos transmitidos e pelo bom ambiente de trabalho.

Por fim quero agradecer a todos os meus amigos, em especial ao Tiago Moreira pois estagiamos no mesmo local, o que ajudou bastante no processo de adaptação.

Resumo

A Nonius Software é uma empresa nacional de engenharia na área de telecomunicações, que se dedica ao desenvolvimento de soluções para a gestão de sistemas informáticos e de entretenimento, tendo como finalidade o mercado mundial hoteleiro e hospitalar.

A solução de TV interactiva da Nonius oferece uma experiência única ao hóspede, ao disponibilizar várias opções de entretenimento e acesso a conteúdos de elevada qualidade e interesse. O hóspede tem acesso a canais de TV, aluguer de filmes, Internet, jogos, informações, promoções e compras na TV.

O objectivo principal desta dissertação foi implementar alguns serviços de entretenimento numa televisão LG Pro: Centric. Este equipamento tem como principal vantagem o facto de conter a *set-top-box* inserida dentro da própria televisão.

Em termos arquitectónicos, o sistema Nonius TV tem dois elementos fundamentais: o *backend*, responsável pelo processamento e tratamento da informação centralizada e o *frontend* instalado nos dispositivos com os quais o hóspede contacta directamente. Uma parte significativa do trabalho desenvolvido centrou-se na implementação de funcionalidades no *backend*. Foram, no entanto, também desenvolvidas algumas funcionalidades nos serviços de *frontend*.

Para o cumprimento dos objectivos estabelecidos, foi utilizada a tecnologia FLASH, tendo como linguagem de programação a segunda versão do ActionScript. Relativamente ao desenvolvimento de *backend* são utilizados o PHP e o JavaScript.

Palavras-Chave

LG Pro: Centric, *Set-Top-Box*, FLASH, ActionScript, *Backend*, *Frontend*

Abstract

Nonius Software is a national company of engineering in the area of telecommunications, dedicated to the development of solutions for the management of computer systems and entertainment, having as its purpose the hotel and hospital market.

Nonius TV offers a unique experience to the guest by offering multiple options for entertainment and access to high-quality content. The guest has access to TV channels, video on demand, Internet, games, information, promotions, and purchases on TV.

The main objective of this thesis was to develop some entertainment services on the television LG Pro: Centric. This equipment has as main advantage the fact that contains the set-top-box inserted inside the TV itself.

In architectural terms, the system Nonius TV has two fundamental elements: the backend, responsible for the processing and treatment of centralized information and the front-end installed in the devices with which the guest contacts directly. A significant part of the work was focused on the implementation of features on backend. However the developed of some features in the front-end services was also considered as part of the objectives.

To fulfill the objectives, the FLASH technology was used, having as programming language the second version of ActionScript. Regarding the developments in the backend, PHP and JavaScript were used to implement the required features.

Keywords

LG Pro: Centric, *Set-Top-Box*, FLASH, ActionScript, *Backend*, *Frontend*

Índice

Índice de Figuras

Índice de Tabelas

Índice de Código

Acrónimos

API – Application Programming Interface

AS – ActionScript

B2B – Business-to-Business

BE – Backend

CSS – Cascading Style Sheets

DE – Database resyinc end

DS – Database resyinc start

DVB-S – Digital Video Broadcasting-Satellite

DVB-T – Digital Video Broadcasting-Terrestrial

FE – Frontend

FIAS – Fidelio Interface Application Specification

GC – Guest Change

GI – Guest Check in

GO – Guest Check out

HD – High Definition

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

IP – Internet Protocol

IPTV – Internet Protocol Television

LA – Link Alive

LD – Link Description

LE – Link End

LS – Link Start

LSO – Local Shared Object

MAC – Media Access Control

MMS – Microsoft Media Services

OSD – On-Screen Display

PA – Posting Answer

PDO – Hypertext Preprocessor Data Objects

PHP – Hypertext Preprocessor

PMS – Property Management System

PNG – Portable Network Graphics

PS – Posting Simple

RTSP – Real Time Streaming Protocol

SMTP – Simple Mail Transfer Protocol

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

STB – Set-Top-Box

TCP – Transmission Control Protocol

TV – Television

UDDI – Universal Description Discovery and Integration

UDP – User Datagram Protocol

URL – Uniform Resource Locator

VoD – Video on Demand

W3C – World Wide Web Consortium

WA – WakeUp Answer

WC – WakeUp Clear

WGS – Wireless Gest Server Aplicance

WR – WakeUp Request

WSDL – Web Services Description Language

XB – Guest Bill Balance

XC – Remote Check Out Request

XML – Extensible Markup Language

XR – Guest Bill Request

1. INTRODUÇÃO

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

Faz-se uma contextualização do trabalho, apresentam-se os objectivos que se pretendem atingir, e apresenta-se a organização do relatório.

1.1. LOCAL DO ESTÁGIO

O trabalho apresentado nesta dissertação foi elaborado nas instalações da Nonius Software, localizada no Parque da Ciência e Tecnologia da Maia, enquadrado no departamento de Investigação e Desenvolvimento (I&D).

A Nonius Software é uma empresa nacional de engenharia na área de telecomunicações, que se dedica ao desenvolvimento de soluções para a gestão de sistemas informáticos e de entretenimento.

Actualmente, a Nonius Software oferece ao mercado soluções avançadas de acesso à Internet e o sistema Nonius TV, uma solução avançada de entretenimento baseada nas tecnologias *Internet Protocol Television* (IPTV) e *Video On Demand* (VoD), tendo como finalidade o mercado mundial hoteleiro e hospitalar.

1.2. OBJECTIVOS

O grande objectivo deste projecto é disponibilizar serviços num cenário em que a *set-top-box* (STB) se encontra dentro da própria TV, ao contrário do cenário já existente na Nonius em que a STB é um dispositivo independente ligado à TV.

Em termos arquitectónicos, o sistema Nonius TV tem dois elementos fundamentais: o *backend* (BE), responsável pelo processamento e tratamento da informação centralizada e o *frontend* (FE) instalado nos dispositivos com os quais o cliente contacta directamente. Uma parte significativa do trabalho desenvolvido centrou-se na implementação de funcionalidades no BE. Foram, no entanto, também desenvolvidas algumas funcionalidades nos serviços de FE.

Os serviços que foram definidos como objectivo deste projecto são:

- Video on Demand (FE);
- Serviço de Quartos (BE);
- Desportos e Lazer (BE);
- *Setup Wizard* (FE);
- *Express Checkout* (BE);
- *Survey Threshold* (BE);
- Mensagem de Boas Vindas (FE & BE);
- Mensagem de Despedida (FE & BE);
- Eventos do Hotel (FE & BE).

1.3. ORGANIZAÇÃO DO RELATÓRIO

Este relatório encontra-se dividido em 6 capítulos.

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

No segundo capítulo apresenta-se algumas tecnologias de suporte aos desenvolvimentos, fazendo-se uma comparação entre HTML5 e Flash, e apresentam-se alguns protocolos essenciais para melhor compreensão de alguns aspectos do projecto.

No terceiro capítulo, faz-se um levantamento de todas as soluções de televisões interactivas oferecidas pela Nonius, assim como uma pequena abordagem a outros produtos.

O quarto capítulo debruça-se sobre a televisão LG Pro: Centric, equipamento utilizado neste projecto, referindo-se a tecnologia e linguagem suportadas e a sua API (*Application Programming Interface*). É também apresentada de forma resumida a interface gráfica e funcional da Nonius TV.

No quinto capítulo aborda-se o desenvolvimento do *software* e são apresentados os resultados.

Por último, no sexto capítulo, finaliza-se o relatório apresentando-se as conclusões e perspectivas do trabalho futuro para optimização do projecto.

2. TECNOLOGIAS E PROTOCOLO

Neste capítulo são abordadas as tecnologias associadas ao projecto. O capítulo está dividido em três secções: a primeira incide sobre a comparação entre as tecnologias FLASH e HTML5; a segunda secção contém uma breve explicação sobre os serviços web (*web services*), tecnologia utilizada no âmbito do trabalho; por último é feita uma abordagem a um protocolo destinado ao ramo hoteleiro no qual o projecto se enquadra.

2.1. FLASH vs HTML5

Nesta secção, faz-se uma comparação entre a tecnologia FLASH e o HTML5 com especial incidência nas técnicas de multimédia.

O conjunto da quinta versão do HTML (*HyperText Markup Language)* com a tecnologia CSS3 *(Cascading Style Sheets)* e algumas APIs (*Application Program Interface*) do JavaScript formam a tecnologia HTML5 que permite criar aplicações complexas [1].

Quanto à tecnologia FLASH a linguagem utilizada é o ActionScript (AS), estando actualmente na sua terceira versão.

Uma das vantagens do HTML5 sob o FLASH é a especificação de *tags* multimédia de áudio e de vídeo. Estas *tags* são totalmente programadas em JavaScript, oferecendo uma grande facilidade de uso assim como uma grande flexibilidade de acesso aos elementos multimédia através da programação. O FLASH obriga à utilização de *third party plugins* para efectuar a reprodução dos conteúdos, o que faz com que seja desvantajoso pois nem todos os *browsers* contêm os mesmos *plugins*. Isto contribui para que seja a tecnologia preferida para equipamentos móveis pois permite aplicações muito mais rápidas.

O facto de o HTML5 ainda estar numa fase de desenvolvimento faz com que a linguagem ainda não esteja estável e sujeita a mudanças. Outro aspecto negativo é o facto de nem todos os *browsers* suportarem a tecnologia, apesar de os browsers mais utilizados (Firefox, Internet Explorer, Google Chrome, Opera, Safari) já o suportarem [2].

Tal como foi referido anteriormente, o HTML5 permite a inclusão de vídeos para reproduzir nativamente no *browser*. Sendo assim, para cada *browser* reproduzir o respectivo vídeo é necessário que inclua *codecs* de vídeo para cada um dos formatos a suportar. Desde que a organização W3C (*World Wide Web Consortium*) decidiu não especificar o formato de vídeo, cada *browser* pode escolher o formato que pretende suportar. Os tipos de formato de vídeo existentes a serem suportados nos *browsers* são o Ogg Theora, Mp4 e *Web formats* [2] [3].

A Tabela 1, faz uma comparação entre os *browsers* mais utilizados no que diz respeito aos formatos de vídeo que suportam.

Tabela 1 Comparação entre os diferentes *browsers* no tipo de formato de vídeo suportado

Browser	MP4	WebM	Ogg
Internet Explorer 9 [4]	Sim	Sim	Não
Firefox 16.0 [5]	Não	Sim	Sim
Google Chrome 22	Não	Sim	Sim
Apple Safari 5.1.7	Sim	Não	Não
Opera 12	Não	Sim	Sim

No caso do Internet Explorer 9, o formato WebM só é suportado se o *codec* VP8 estiver instalado. Quanto ao Google Chrome, o formato Mp4 chegou a ser suportado mas foi removido [4].

Com o aparecimento da *tag* `<vídeo>` *do* HTML5, torna-se possível inserir vídeos em páginas web de uma forma muito mais simples, semelhante à inserção de imagens em documentos HTML, onde o próprio browser fornece as funções de reprodução sem a necessidade de *plugins* como no FLASH.

Ou seja, para reproduzir um vídeo em HTML5 é apenas necessário o Código 1.

```
<html>
  <head>
  </head>
  <body>
    <video id="video_with_controls" width="320" controls
autobuffer>
       <source
src="http://playground.html5rocks.com/samples/html5_misc/chr
ome_japan.webm" type='video/webm' />
      Your browser does not support the video tag
    </video>
    <button id="play">Play</button>
    <button id="pause">Pause</button>
    <script>
      var movie =
document.getElementById('video_with_controls');
      document.getElementById('play').addEventListener('clic
k', function() { movie.play(); }, false);
      document.getElementById('pause').addEventListener('cli
ck', function() { movie.pause(); }, false);
    </script>
  </body>
</html>
```

Código 1 Exemplo de um reprodutor de vídeo em HTML5

O resultado deste código é um reprodutor de vídeo bastante simples, tal como se verifica na Figura 1, com as funcionalidades de *play*, *pause* e *mute*.

Figura 1 Resultado de um reprodutor de vídeo utilizando HTML5

O parâmetro *controls* adiciona controlos de vídeo tais como *play*, *pause* e volume. Apesar de não ser obrigatório, é correcto inserir os atributos relativos à largura e altura. Definindo esses atributos antecipadamente, o espaço para o vídeo é reservado enquanto a página é

carregada. Caso contrário, o *browser* não saberá o tamanho do vídeo e não irá reservar o espaço adequado para ele, o que faz com que o *layout* da página mude durante o seu carregamento. A inserção de texto entre as *tags* de vídeo é aconselhável para casos em que o browser não suporte a *tag*. Permite também inserir múltiplos elementos fonte (`<source>`). O *browser* irá utilizar o primeiro formato a ser reconhecido.

Foram adicionados dois botões do lado direito do reprodutor para mostrar a interligação entre o HTML e o JavaScript. Ou seja, foi adicionado um *listener* a cada botão, que a partir do momento que este detecta um *click* executa a função que a *tag* vídeo associa a cada botão do reprodutor.

A *tag* vídeo possui também um atributo denominado *poster* que fornece um endereço de um arquivo de imagem e que pode ser apresentada enquanto não há dados de vídeo. A imagem dada pelo atributo *poster*, o *poster frame*, destina-se a ser uma espécie de cartaz representativo do vídeo, ou seja, dá ao utilizador uma ideia de como o vídeo é. Esta técnica é utilizada em páginas *web* como o Youtube [1].

Para ser recriado este tipo de reprodutor de vídeo em ActionScript é necessário criar as imagens que representariam as funcionalidades de *play* e *pause*, e associar às mesmas um evento que fosse activado aquando do clique.

Tal como pode ser visto no Código 2 é inicializada uma variável do tipo `NetStream`. A classe `NetStream` fornece métodos e propriedades para reproduzir vídeos FLASH do sistema local ou de um endereço HTTP (*Hypertext Transfer Protocol*). É também utilizado um objecto do tipo `NetStream` para transmitir um vídeo através de um objecto `NetConnection`. A classe `NetConnection` fornece os meios para reproduzir as *streams* FLV (formato de vídeo FLASH).

Apenas é mostrada a funcionalidade de *play*, existindo outras que permitem a pausa ou a paragem do vídeo [6].

É chamado então o método `play` da classe `NetStream` passando-lhe como argumento o URL (*Uniform Resource Locator*) do vídeo.

```
private var ns:NetStream;
private var video:Video;
private var vídeo_enabled:Boolean = false;
public function VideoPlayer()
{
```

```
   var nc:NetConnection;
   video = new Video();
   nc = new NetConnection();
   nc.connect(null);
   ns = new NetStream(nc);
   video.attachVideo(ns);
}

public function play():Void
{
var video_sample_str:String=
"http://www.helpexamples.com/flash/video/cuepoints.flv";
ns.play(video_sample_str);
}
```

Código 2 Exemplo da reprodução de um vídeo em AS

Comparando o Código 1 e o Código 2, é possível verificar-se que o código para a criação do reprodutor em FLASH é mais complexo do que em HTML5.

Convém também referir que existem ferramentas da Adobe que facilitam o desenvolvimento de aplicações em FLASH. O facto de o HTML5 não necessitar de ferramentas de desenvolvimento representa uma vantagem adicional.

Para manter a robustez nas transmissões de vídeo, é utilizado como medida de precaução o FLASH. Por exemplo, a funcionalidade de *fullscreen* não é possível obter apenas com código HTML e alguns *browsers* ainda não suportam o uso de JavaScript para iniciar o modo *fullscreen*. Contudo em *browsers* como o Firefox, o Chrome e o Safari já suportam esta funcionalidade, tal como se pode verificar na Tabela 2.

Tabela 2 Comparação entre os diferentes *browsers* relativamente ao suporte de *fullscreen*

Browser	Fullscreen
Internet Explorer 9	-
FireFox 16.0	Disponível
Google Chrome 22	Disponível
Opera 12	-
Safari 5.1.7	Disponível

Tal como para o vídeo, o HTML5 prevê uma *tag* dedicada ao áudio, não sendo necessários *third party plugins* para oferecer o elemento `<audio>`. O código necessário para reproduzir áudio é bastante semelhante ao do vídeo, tal como o Código 3 demonstra.

```
<audio controls="controls">
  <source src="song.ogg" type="audio/ogg" />
  <source src="song.mp3" type="audio/mpeg" />
  Your browser does not support the audio element.
</audio>
```

Código 3 Exemplo da reprodução de um ficheiro de áudio utilizando HTML5

Uma das novas funcionalidades do HTML5 é a possibilidade da utilização do elemento *canvas*. Este elemento permite desenhar gráficos numa página Web, incluindo formas geométricas, alteração dos gradientes de cor, inclusão de imagens, etc. [1]

Para utilizar o elemento `canvas` apenas é necessário definir o `id`, a largura e altura do elemento tal como se verifica no Código 4.

```
<canvas id="rectangle" width="200" height="100"></canvas>
```

Código 4 Exemplo da criação de um elemento *canvas* em HTML5

Obviamente, que apenas isto não irá fazer os desenhos esperados, pois o elemento `canvas` não possui técnicas de desenho por si só. Sendo assim, todo o conteúdo de desenho deverá estar inserido num ficheiro JavaScript, semelhante ao do Código 5.

```
<script type="text/javascript">
var c=document.getElementById("rectangle");
var ctx=c.getContext("2d");
ctx.fillStyle="#FF0000";
ctx.fillRect(0,0,130,75);
</script>
```

Código 5 Exemplo da criação de um rectângulo vermelho

Tornou-se bastante mais simples fazer desenhos ou alterar imagens, pois utilizando JavaScript, basta apenas obter o `id` definido no elemento `canvas` e utilizá-lo para proceder assim ao desenho gráfico, tendo acesso a todas as propriedades desse objecto. O resultado do código será um rectângulo vermelho, tal como consta na Figura 2.

Figura 2 Resultado de um exemplo simples utilizando a propriedade *canvas* do HTML5

No FLASH é possível armazenar dados localmente no *browser* do utilizador, utilizando os *local shared object* (LSO), também conhecidos como FLASH *cookies*. São ficheiros de dados que podem ser criados no computador pelas páginas *web* que foram visitadas. *Shared Objects* são frequentemente utilizados para melhorar a experiência de navegação web. Uma página *web* pode gravar um *cookie* no computador de forma a que quando for visitada novamente esse *cookie* será carregado proporcionando uma experiência mais personalizada. Este conceito é bastante utilizado quando se acede a páginas *web* que necessitam de login e se pretende guardar essa informação para um próximo acesso [7].

Os LSO que estão armazenados no computador do utilizador, pelas páginas *web* visitadas, são apenas lidos pelas páginas que os criaram, tal como a Figura 3 ilustra.

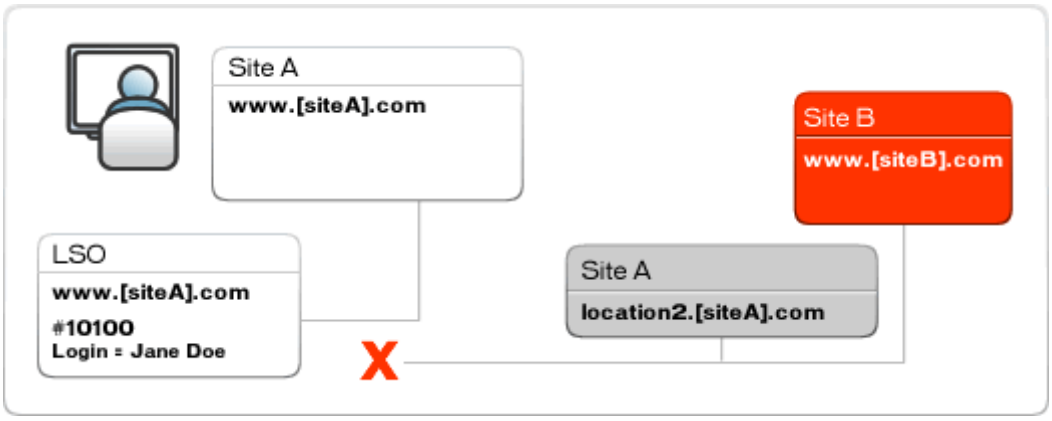

Figura 3 Apresentação do funcionamento dos *local shared objects* [7]

O HTML5 também permite armazenar dados localmente no *browser* do utilizador. Os dados não estão incluídos em cada pedido do servidor, sendo apenas utilizados quando solicitados. É também possível armazenar grandes quantidades de dados sem afectar o desempenho da página web. Os dados são armazenados em pares chave/valor e a página web só pode aceder aos dados armazenados por si [8].

No que diz respeito a plataformas móveis, a Adobe decidiu retirar o FLASH da loja Google Play, pois irá adoptar o HTML5 para esse tipo de plataformas [9].

2.2. SERVIÇOS WEB

A presente secção introduz o conceito de serviços *web* (*web services*) que serão utilizados no desenvolvimento do projecto.

A tecnologia *web service* é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas e linguagens diferentes sejam compatíveis. Os serviços *web* são componentes que permitem às aplicações enviar e receber dados em formato XML (Extensible Markup Language), independentemente da linguagem de programação utilizada.

Esta tecnologia foi criada para construir aplicações que funcionam como serviços na internet. É a tecnologia ideal para a comunicação entre sistemas, sendo muito usada em aplicações B2B (*Business-to-Business*). A comunicação entre os serviços é assim normalizada possibilitando a independência da plataforma e da linguagem de programação. Esta tecnologia inclui todos os sistemas que prestam serviços e que trocam mensagens no formato SOAP (*Simple Object Access Protocol*) e/ou possuem interfaces descritas em WSDL (*Web Services Description Language*), tal como é demonstrado na Figura 4 [10].

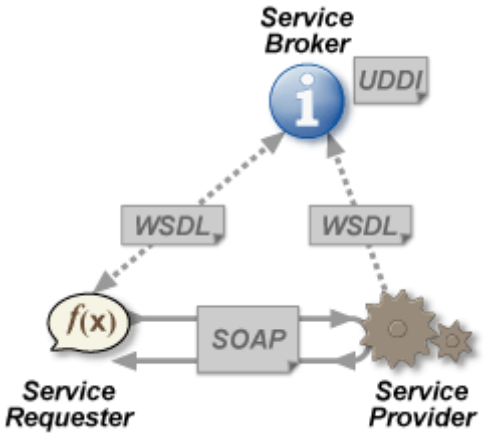

Figura 4 Arquitectura dos serviços *web* [11]

A Figura 5 apresenta a arquitectura dos serviços *web* em camadas com algumas dessas tecnologias [12].

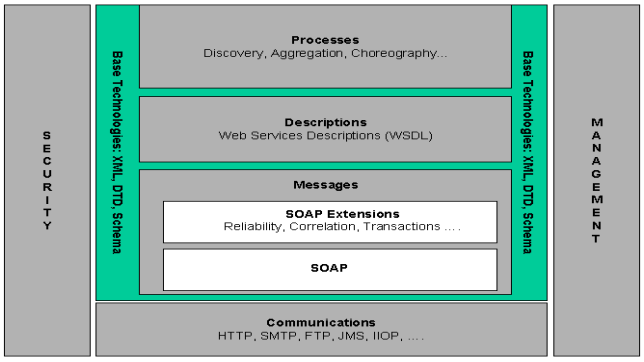

Figura 5 Arquitectura dos serviços *web* em camadas [12]

As secções seguintes apresentam as tecnologias mais importantes para o desenvolvimento e utilização dos serviços *web*.

2.2.1. *EXTENSIBLE MARKUP LANGUAGE* (XML)

A linguagem de anotação XML foi adoptada para a representação e estruturação de dados, a descrição dos serviços e o conteúdo das mensagens trocadas entre aplicações (cliente e serviço *web*). Na qualidade de metalinguagem de anotação, permite definir novos vocabulários XML, ou seja, novos conjuntos de anotações destinados a caracterizar e estruturar os dados. Estes vocabulários XML, quando são adoptados para a representação de dados, permitem que, de forma automática, se criem, por um lado, documentos válidos e bem formados para armazenamento de informação, e, por outro, se extraiam e transformem os dados neles contidos para outros formatos de representação.

No caso dos serviços *web*, a linguagem XML desempenha um papel crucial, pois é utilizada para a descrição dos serviços e para a troca de informação.

2.2.2. *SIMPLE OBJECT ACCESS PROTOCOL* (SOAP)

O protocolo SOAP permite a invocação remota de um método e para tal necessita especificar o endereço do componente, o nome do método e os argumentos para esse método. Estes dados são formatados em XML com determinadas regras e enviados normalmente por HTTP (*HyperText Transfer Protocol*) para esse componente.

SOAP é um protocolo projectado para invocar aplicações remotas através de RPC (*Remote Procedure Call*) ou trocas de mensagens, num ambiente independente da plataforma ou linguagem de programação. Desta forma, pretende-se garantir a interoperabilidade e

intercomunicação entre diferentes sistemas, através da utilização do XML e de mecanismos de transporte, tal como o HTTP.

O facto de não ser necessário um conhecimento exaustivo deste protocolo para o desenvolvimento de serviços *web* é bastante vantajoso, pois a maioria das linguagens de programação já contém classes implementadas deste protocolo.

A mensagem SOAP é um ficheiro XML que contém os seguintes elementos:

- **Envelope**: identifica o ficheiro XML como sendo uma mensagem SOAP;
- ***Header***: contém as informações de cabeçalho;
- ***Body***: contém as informações dos pedidos e das respostas;
- ***Fault***: contém os erros e informações do estado [13].

2.2.3. *WEB SERVICES DESCRIPTION LANGUAGES* (WSDL)

O WSDL visa normalizar as descrições das funcionalidades oferecidas pelo serviço *web* de forma independente da plataforma ou linguagem. Possui basicamente três finalidades: expor os métodos que determinado serviço disponibilizará; possibilitar a localização de determinado serviço e explicar o que o serviço faz [10] [14].

2.2.4. *UNIVERSAL DESCRIPTION DISCOVERY PROTOCOL* (UDDI)

O UDDI é um protocolo desenvolvido para a organização e registo dos serviços web. Permite a descrição, descoberta e integração universal de aplicações de Internet. É um directório de armazenamento de informação acerca de serviços web disponíveis, assim como um directório de interfaces de serviços *web* descritos em WSDL. A troca de mensagens é efectuada através do protocolo SOAP [10].

A Figura 6 ilustra a forma como todas estas tecnologias se relacionam e que pode ser resumida nos seguintes passos:

1 – Como o cliente pode não ter conhecimento de qual o serviço *web* que quer invocar, o primeiro passo é encontrar o serviço que atende aos requisitos do cliente. Por exemplo, o cliente quer localizar um serviço web que lhe retorne a conversão de euros para dólares. Isto é feito acedendo ao registo UDDI.

2 – O registo UDDI vai responder, dizendo quais os servidores que podem fornecer o serviço pretendido.

3 – Como já se sabe a localização do serviço web, o próximo passo será invocá-lo. Mas como o cliente não sabe qual o método que deve invocar, é necessário perguntar ao serviço *web* para se descrever.

4 – O serviço *web* vai responder utilizando a tecnologia WSDL.

5 – Após ter conhecimento da sua localização e de como o invocar, utiliza-se o protocolo SOAP para fazer o pedido.

6 – Por fim o serviço *web* irá responder com o resultado ou mensagem de erro.

Figura 6 Invocação de um serviço web

2.3. CONCEITOS RELACIONADOS COM O RAMO HOTELEIRO

Esta secção tem como objectivo esclarecer alguns conceitos relacionados com o ramo hoteleiro que são importantes para contextualizar o desenvolvimento do projecto.

2.3.1. *PROPERTY MANAGEMENT SYSTEMS* (PMS)

O *software* PMS (*Property Management Systems*) é utilizado maioritariamente no ramo da hotelaria para automatizar as operações de um hotel, facilitando o controlo de propriedades. Trata do processamento de reservas, *check-ins* e *check-outs*.

O *check-in* é o primeiro procedimento ao qual o cliente será submetido e onde fornece os seus dados, tais como o seu nome, nacionalidade, cartão de crédito, etc. Sem o check-in feito, o cliente não consegue aceder a nenhum quarto do hotel, pois é neste procedimento que lhe é atribuído um.

O *check-out* é o último procedimento ao qual o cliente é submetido e que identifica a sua saída. Neste procedimento é feita uma análise das instalações utilizadas pelo cliente, assim

como uma análise ao relatório de despesas do mesmo. Depois de o cliente efectuar o pagamento das suas despesas, o quarto em questão é marcado como disponível novamente.

O PMS permite também uma ligação com as áreas de contabilidade de *front office*, marketing, bem como interface com outros sistemas, tais como telefones, restaurantes, actividades do hotel, serviços de quarto ou *video on demand* [15].

2.3.2. FIDELIO INTERFACE APPLICATION SPECIFICATION (FIAS)

O protocolo FIAS foi criado em 1997 pela Micros, sendo o protocolo mais utilizado no ramo hoteleiro [16]. O objectivo do protocolo FIAS é definir uma norma para formatos de registo e fluxos de dados a ser utilizado para comunicações de dados entre um *Micros-Fidelio Property Management System* e outros sistemas de computador do hotel.

As mensagens enviadas ou recebidas são identificadas de imediato quanto à sua função através dos seus dois primeiros caracteres que representam o seu tipo. Isto permite que o campo seja facilmente reconhecido. Como exemplo, na mensagem do Código 6:

```
GI|RN110|GNRicardo
```

Código 6 Exemplo de um registo de *check in*

o código GI representa o tipo de registo, que neste caso seria um *check-in*. Os elementos RN e GN representam o número do quarto e o nome do hóspede respectivamente.

Existem diversos tipos de registos, estando eles divididos por tipo. Existem os registos de comunicação e controlo de ligação, sincronização da base de dados, dados dos clientes, extensão de dados do cliente, localizadores, informações do quarto, despertador, taxações, etc [17].

2.3.2.1. REGISTOS DE COMUNICAÇÃO E CONTROLO DE LIGAÇÃO

Existem 4 registos de comunicação e controlo de ligação: *Link Start* (LS), *Link Alive* (LA), *Link End* (LE) e *Link Description* (LD).

Este tipo de registos são utilizados para controlar o estado da ligação. Isto significa que se um LS é recebido a partir do PMS, a descrição da ligação (LD) e o LA devem ser retransmitidos. O LA é fornecido como um meio de verificar se a ligação ainda está activa.

Se o PMS enviar um LE, o outro sistema deve armazenar todos os registos não descartáveis (ex.: taxação) até que receba a comunicação seguinte.

As mensagens LS, LA e LE são constituídas pelos dois caracteres que identificam o tipo de registo, a data (DA) (YYMMDD) e a hora (TI) (HMS). Quanto ao LD é igual aos anteriores, apenas com mais um campo que se destina à identificação da versão do outro sistema [17].

As mensagens que ilustram a utilização deste tipo de registo encontram-se no Código 7.

```
LS|DA120512|TI112301|
LA|DA120512|TI112301|
LE|DA120512|TI112301|
LD|DA120512|TI112301|V#1.01|
```

Código 7 Exemplos de mensagens de registos de comunicação e controlo de ligação

2.3.2.2. REGISTOS DE DADOS DOS HÓSPEDES

Existem três registos relacionados com os dados dos hóspedes, *Guest Check-in* (GI), *Guest Check-out* (GO) e *Guest Change* (GC).

Estes registos são utilizados para transmitir dados relativos aos hóspedes, ou seja, qualquer informação necessária para definir ou actualizar os dados dos hóspedes será incluída nestes registos. Os registos podem conter campos de dados semelhantes, mas o tipo de registo especifica que acções devem ser executadas.

A mensagem do Código 8, representa como é transmitido o registo de *Check-in*.

```
GI|RN271|G#123|GNGuest, Mr.|GLEN|
```

Código 8 Exemplo de uma mensagem de *check in*

Onde, RN corresponde ao número do quarto, G# ao número do cliente, GN ao nome do cliente e GL à sua nacionalidade.

De forma semelhante, se demonstra a transmissão do registo de *Check-out* no Código 9.

```
GO|RN271|G#123
```

Código 9 Exemplo de uma mensagem de *check out*

Para ser feito o *check-out* não é obrigatório enviar nenhum dado do cliente, como por exemplo o nome, já que a ideia do registo *check-out* é para colocar o quarto como disponível apagando os dados do último cliente.

Se for necessário fazer uma mudança de quarto, nome ou nacionalidade é utilizado o registo GC, tal como se pode ver na mensagem do Código 10,

```
GC|RN123|GNRicardo|GLPT|RO271
```

Código 10 Exemplo de uma mensagem de mudança de quarto

onde o RN corresponde ao novo número do quarto, os campos GN e GL já estão actualizados com a mudança e o campo RO corresponde ao antigo quarto [17].

2.3.2.3. REGISTOS DE SINCRONIZAÇÃO DE BASE DE DADOS

Existem três registos de sincronização de base de dados, *Database resyinc request* (DR), *Database resyinc start* (DS) e *Database resyinc end* (DE).

Estes registos são utilizados para solicitar uma inicialização ou actualização da base de dados do sistema. Como o PMS pode misturar os registos da base de dados com os registos em tempo real, os registos DS e DE asseguram que o outro sistema conhece o seu pedido e que este foi recebido correctamente e que todas as informações da actualização da base de dados foram enviadas [17].

As mensagens do Código 11 são exemplos do registo de sincronização de base de dados.

```
DR|DA120512|TI123412|
DS|DA120512|TI123416|
GI|RN1001|G#123|GSN|
GO|RN11|G#133|
DE|DA120512|TI123512|
```

Código 11 Exemplo de uma sincronização de base de dados

O elemento GS aparece pela primeira vez e representa se é um quarto individual ou não. Caso seja enviado Y representa um quarto individual.

2.3.2.4. REGISTOS DE DADOS ESPECÍFICOS DOS CLIENTES

Existem vários registos para este tipo de acção, mas apenas irão ser mencionados os que foram utilizados no âmbito do trabalho desenvolvido. *Guest Bill Request* (XR), *Guest Bill Balance* (XB) e *Remote Check-Out Request* (XC).

Estes registos fornecem um mecanismo para pedir e enviar informação específica dos clientes.

O registo XR permite ao cliente aceder à sua conta, ou seja ao relatório de todas as suas despesas. Sendo assim, no registo apenas basta enviar o número do quarto e do cliente, como é exemplificado na mensagem do Código 12.

```
XR|RN110|G#123|
```

Código 12 Exemplo de uma mensagem de pedido de conta

Quanto ao registo XB consiste na resposta do registo XR, sendo utilizado para saber o balanço de despesas do cliente. A construção deste registo é em tudo semelhante à do caso anterior apenas com o acréscimo do campo BA que corresponde ao total de despesas. A mensagem do Código 13 ilustra a utilização deste registo, em que 1200 no campo BA corresponde á quantia total até ao momento que o cliente fez em despesas.

```
XB|RN110|G#123|BA1200|
```

Código 13 Exemplo de uma mensagem de resposta ao pedido de conta

Por fim, o registo XC corresponde a um *check-out* realizado de forma remota, ou seja, se um cliente necessitar de terminar a sua estadia no hotel e não tiver tempo de o fazer da forma convencional.

Sendo assim, é enviada uma mensagem igual à anterior mas do tipo XC, tal como se pode verificar nas mensagens do Código 14, e irá resultar uma resposta positiva ou negativa [17].

```
XC|RN110|G#123|BA1200|
XC|RN110|G#123|ASOK|BA1200
```

Código 14 Exemplo de uma mensagem de express *check-out*

2.3.2.5. REGISTOS DE ALARME

Existem três registos de alarme**,** *Wakeup Request* (WR), *Wakeup Answer* (WA) e Wakeup Clear (WC).

Se o cliente desejar ser acordado a uma hora definida, pelo hotel ou por algum equipamento que o hotel forneça, este é o tipo de registos que será utilizado.

Ao definir a hora da *wakeup call* será utilizado o registo WR, tal como a mensagem do Código 15 seguinte apresenta:

```
WR|RN2781|DA121031|TI070000|
```

Código 15 Exemplo de um registo de definição do alarme

Neste exemplo, o pedido foi feito, para o dia 31 de Outubro de 2012 às 7 horas da manhã.

A resposta à *wakeup call* definida anteriormente, caso fosse possível, seria a demonstrada no Código 16. Caso não fosse possível, o campo AS iria ser alterado conforme o tipo de perturbação que ocorreu [17].

```
WA|RN2781|DA001031|TI070000|ASOK|
```

Código 16 Exemplo da resposta ao registo WR

Por fim, para o hóspede apagar todos os pedidos de *Wakeup Call* que fez é enviada a mensagem presente no Código 17.

```
WC|RN2781|DA|TI|
```

Código 17 Exemplo de uma mensagem que apaga todos os registos de alarme

2.3.2.6. REGISTOS DE TAXAÇÃO

Existem vários registos de taxação mas apenas serão explicados os dois utilizados neste projecto, *Posting Simple* (PS) e *Posting Answer* (PA).

A forma simples deste tipo de registos (PS) é utilizada em sistemas que fazem as taxações sem ter de contactar o cliente (ou seja, telefone, TV, etc). Estes sistemas geralmente usam os registos GI/GO para garantir que o sistema num determinado quarto deve ser activado ou não.

Este registo pode ser constituído por vários campos, sendo estes seleccionados de acordo com a taxação efectuada.

No Código 18, é enviado o registo da taxação de uma chamada telefónica internacional.

```
PS|RN2781|TA10|DA000915|TI123545|P#1729|DD004989920920|
PCI|CTInternational|PTC|
```

Código 18 Exemplo de uma mensagem de *Posting Simple*

O campo que corresponde à quantia total a pagar é o TA, sendo neste caso o valor de 10 Euros. O número da taxação corresponde ao P# enquanto que o número para qual o cliente realizou a chamada é passado no campo DD. Quanto aos campos CT e PT correspondem ao tipo de chamada telefónica (nacional, internacional) e tipo de taxação. Sempre que um registo deste tipo é enviado, é necessário obter uma resposta, que é detectada quando é recebido o registo *de Posting Answer* (PA), tal como o Código 19 demonstra.

```
PA|RN2781|ASOK|P#1729|DA000915|TI123545|
```

Código 19 Exemplo da mensagem de resposta á mensagem de PS enviada

Para além dos campos já conhecidos, são também enviados o número do pedido de taxação e uma reposta (AS) se tudo foi feito com sucesso ou não [17].

2.3.3. NiVo3PFO

O NiVo3PFO é uma aplicação Java que é responsável pela comunicação entre o *backend* e os PMS. Quando o serviço é executado, a comunicação com os PMS é estabelecida e é criado um *pool* de *socket*s para as portas 3000 e 4000 para comunicação com set-top-boxes (STBs) e outros produtos oferecidos pela Nonius. Além disso, no segmento principal são criados os objectos responsáveis pela conexão com a base de dados e o objecto que mapeia um IP com o número sequencial.

Se a comunicação for efectuada pela porta 3000, esta irá tratar as mensagens/registos enviados pela STB, como por exemplo da caixa de mensagens, pedidos para ver o seu relatório de despesas ou mesmo a compra de produtos e *check-ins/check-outs*.

Caso seja um produto de gestão de acesso à Internet, deverá se conectar à porta 4000. É utilizada para fazer controlos de ligação (LA/LS/LE) e actualizações da base de dados.

Para explicar como se processam os pedidos e respostas de todo este sistema, utilizaremos o pedido de taxação de um hóspede. Para garantir que o NiVo3PFO envia a mensagem PA para o cliente correcto, é necessário depois de receber uma mensagem do tipo PS, associar o número do pedido de taxação com o endereço IP (*Internet Protocol*) do cliente que envia a mensagem, tal como se pode ver na Figura 7.

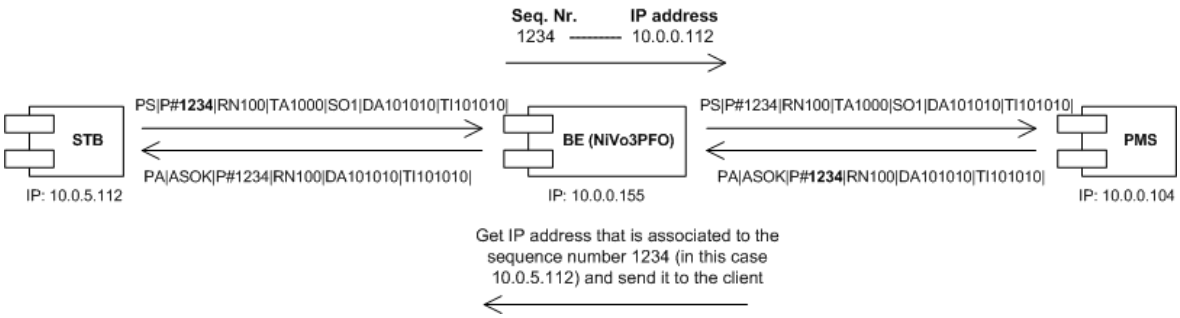

Figura 7 Solução para resolver o problema de transmissão de mensagens [18]

Caso contrário, o NiVo3PFO não sabe para onde enviar a mensagem PA e pode provocar com que esta se perca e/ou seja enviada para uma outra STB que não a que realizou o pedido, tal como a Figura 8 demonstra [18].

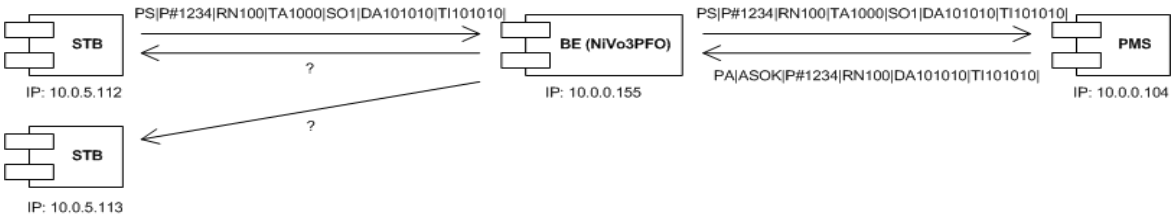

Figura 8 Problema com a transmissão de mensagens [18]

2.3.4. PROPERTY MANAGEMENT SYSTEM SERVER

O PMS *server* é responsável pela facturação de um hotel, suportando todas as principais interfaces de gestão, Fidelio, NewHotel, EasyLinkHotel, Brilliant, Sihot e Biologica.

A sua integração com os sistemas de televisão interactiva da Nonius é feito através do protocolo *standard* FIAS que opera sobre a camada transporte TCP/IP, sendo utilizado pelas várias interfaces PMS [19].

3. NONIUS TV

A solução de TV interactiva da Nonius oferece uma experiência única ao hóspede ao disponibilizar várias opções de entretenimento e acesso a conteúdos de elevada qualidade e interesse. O hóspede tem acesso a canais de TV, aluguer de filmes, Internet, jogos, informações, promoções e compras na TV. Esta solução é adaptada à imagem do Hotel sendo uma óptima ferramenta para comunicar com o hóspede e promover os serviços do Hotel, ao mesmo tempo que reduz custos de operação [19].

3.1. NONIUS TV: ARQUITECTURA

A arquitectura do sistema NONIUS TV distingue três camadas distintas: a *NiVo Service Layer*, a *Control and Conectivity Layer* e a *Access Room Layer*.

Tal como a Figura 9 apresenta, a primeira camada, a *NiVo Service Layer*, incorpora todos os serviços oferecidos pelo sistema NONIUS TV, ou seja, IPTV, VOD, TV, rádio, jogos, serviços para os utilizadores, navegação na internet, aplicações Web e serviços informativos.

A camada intermédia, a *Control and Conectivity Layer*, corresponde a um centro de dados com todos os componentes de *hardware* e software necessários para implementar o sistema NONIUS TV.

Por fim, a última camada, pode identificar-se como *Access Room Layer*, o subsistema no quarto do hotel com a STB, TV e componentes para disponibilização de Internet sem fios, proporcionando uma interface com o cliente final.

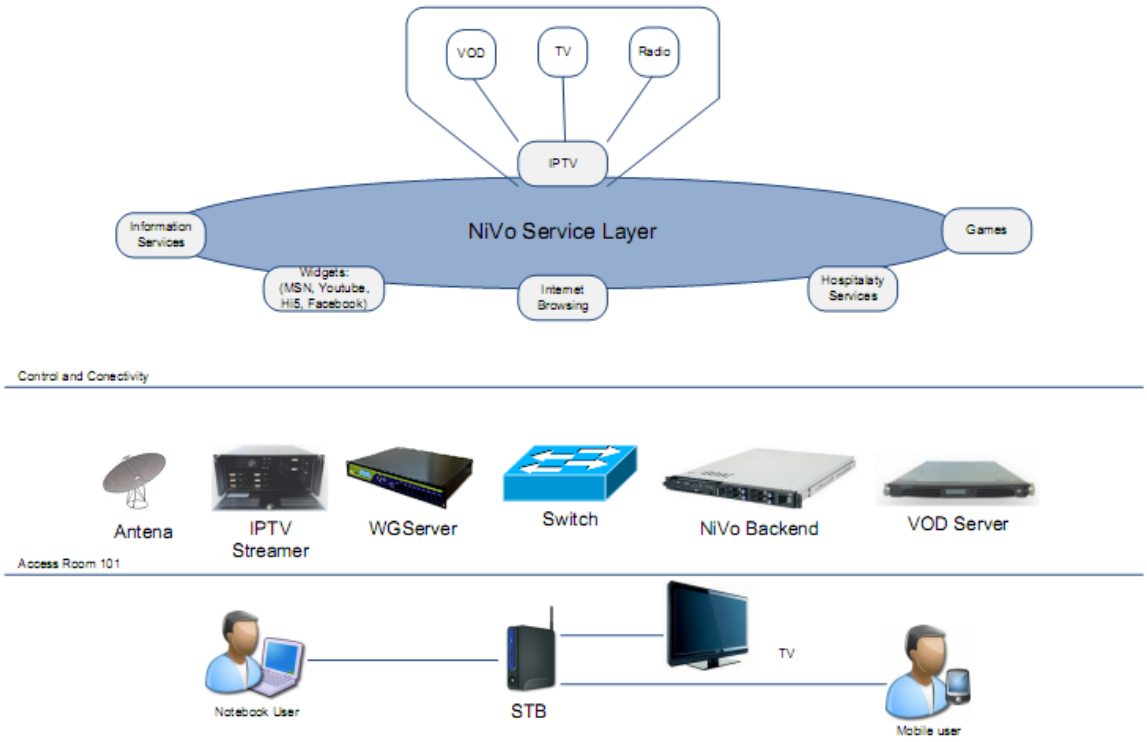

Figura 9 Arquitectura do sistema NONIUS TV [19]

3.2. NONIUS TV: COMPONENTES

O sistema Nonius TV é composto por vários agentes distintos, entre os quais, as *Set-Top-Box*, o *Backend Server*, o IPTV *Server*, *o VoD Server*, o *WGServer Apliance* e o PMS *Server*.

A Figura 10 exemplifica as interligações dos diversos componentes direccionados para a área hoteleira.

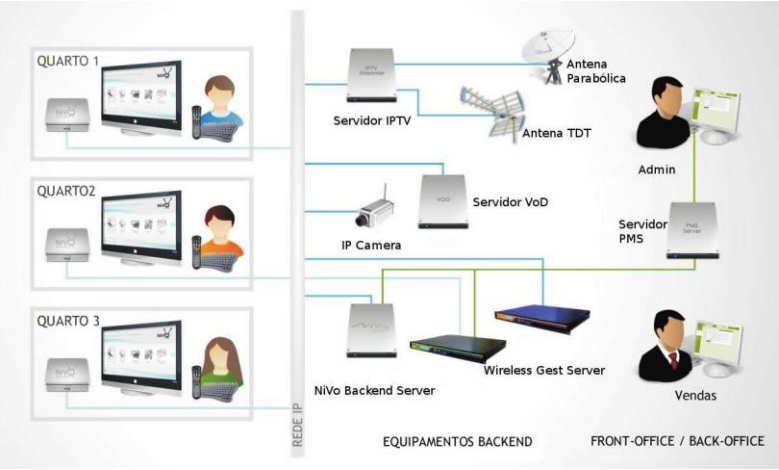

Figura 10 Componentes do sistema NONIUS TV [19]

As *Set-Top-Box* encontram-se ligadas às televisões de alta definição e ao *Backend Server*, sendo as STBs geridas pelo *Backend Server*, desde o *firmware* aos temas para interface gráfica e aos serviços NONIUS TV. O termo tema é utilizado para representar todas as imagens e ficheiros XML que permitem a personalização da interface gráfica.

Estes equipamentos têm como base um sistema Linux, o *Minimyth*, configurado pela Nonius Software para suportar vários modelos de TVs, disponibilizando os serviços NONIUS TV, personalizados ou não, para o utilizador final de acordo com o *design* da empresa, suportando navegação na Internet, jogos, música, vídeo de resoluções HD (*High Definition*) de 1080p, ponto de acesso wireless e conexão com teclado/rato.

O NONIUS TV *Backend Server* é um sistema centralizado de gestão de funcionalidade e conteúdos tendo como base um sistema Linux, o *MythTV*. Disponibiliza ferramentas de gestão, que incluem todas as interfaces Web, permitindo gerir e configurar todos os serviços e conteúdos NONIUS TV, que incluem informação sobre o tempo, agenda de voos, horários de atendimento de farmácias, canais de TV, lista de filmes ou listas de reprodução de rádio. É responsável pela gestão de todo o *firmware* e configurações das STBs, assim como de todos os serviços disponíveis ao utilizador final através das STB. Suporta todas as interfaces do NONIUS TV a correr em cada STB, inclui base de dados e interface com o sistema operativo e o *hardware*. Interage também com o PMS *Server* responsável pela facturação do hotel.

O *IPTV Server* tem como função receber e converter as emissões digitais ou analógicas de TV em formato IPTV, injectando-as na rede TCP/IP. Transmite os pacotes digitais para a

infra-estrutura NONIUS TV, convertendo os sinais de entrada a partir de DVB-S (*Digital Video Broadcasting – Satellite*), DVB-T (*Digital Video Broadcasting – Terretrial*) ou entrada analógica.

O *Video-on-Demand Server* tem como função o armazenamento e disponibilização de conteúdo VoD na infra-estrutura NONIUS TV, distribuindo fluxo de vídeo, nos formatos MPEG – 2 e MPEG – 4 HD, pela rede IP.

O *Wireless Gest Server Aplicance* (WGS) é um servidor de gestão de acesso à Internet desenvolvido pela Nonius Software baseado no ChilliSpot. Tem funções de *gateway*, suportando uma aplicação de gestão e controlo de *hotspots* com tarifação de acesso, sendo responsável por todo o acesso à Internet por parte dos utilizadores [19].

3.2.1. SOLUÇÕES DE TELEVISÃO INTERACTIVA NONIUS

A Nonius oferece três soluções no que diz respeito a televisões interactivas: LG Pro:Centric, Android TV e *Premium*.

3.2.1.1. LG PRO:CENTRIC

Esta solução será explicada de forma mais detalhada no seguimento deste documento. Nesta secção será contudo feito um breve resumo das suas características.

Em relação aos produtos que a Nonius oferece este tem a grande vantagem de não necessitar de ter uma set-top box externa. Oferece diversos serviços que incluem mensagem de boas-vindas/serviço de mensagens, *vídeo-on-demand*, compra de serviços do hotel (spa, restaurante, golf, aluguer de viaturas), serviços de quarto, integração com sistema PMS do hotel, conta na TV, *express check-out*, inquérito de satisfação, jogos, canais de TV e radio, canal corporativo/informativo e *wake up calls*.

As grandes vantagens desta solução são a facilidade de instalação, menor número de componentes activos, a fácil utilização, *zapping* em menos de 1 segundo, alta definição (1080p) e manutenção remota da TV.

3.2.1.2. SOLUÇÃO ANDROID TV

A solução de TV interactiva para hotelaria da Nonius também está disponível para plataformas Android.

O Android TV da Nonius oferece uma experiência de entretenimento e conectividade que vai além do que oferece uma *Smart* TV. O hoteleiro não necessita de actualizar as TVs LCD ou LED que já tem, para oferecer a experiência Android TV ao seu hóspede.

Mantendo as funcionalidades para hotelaria da solução Nonius, a plataforma Android TV acrescenta o desempenho e usabilidade dos jogos e aplicações do Android *Market*. Tudo isto a um baixo custo de aquisição que só é possível graças à massificação da tecnologia Android.

Para além das funcionalidades da solução LG Pro:Centric oferece também *widgets* e Internet na TV.

As características desta solução são o seu menor custo de aquisição, alto desempenho da plataforma Android, variedade de aplicações e jogos disponíveis e elevado desempenho gráfico.

3.2.1.3. SOLUÇÃO PREMIUM

A solução Premium é a mais completa das soluções de televisão interactiva que a Nonius oferece ao mercado hoteleiro. É uma solução baseada em tecnologias IPTV, sendo a única que oferece um ponto de acesso Wi-Fi em cada set-top box e integra com outros dispositivos e sistemas, como por exemplo os painéis MediaHub (oferece interfaces como HDMI, USB, AV, áudio).

Os serviços que esta solução oferece são em tudo semelhantes ao da Pro:Centric com a adição de internet *wireless* para o quarto, *widgets* (Facebook, Messenger, Wikipedia) e Internet na TV.

As suas vantagens são permitir Internet na TV, assim como, um ponto de acesso Wi-Fi integrado na set-top box e *Digital Signage* (painel informativo).

Tabela 3 Comparação entre as soluções oferecidas pela Nonius Software

Solução / Características	NONIUS TV | Premium	NONIUS TV | Android	NONIUS TV | LG Pro:Centric
TV interactiva	✓	✓	✓
Internet na TV	✓	✓	✗
Ponto de Acesso Wirless	✓	✗	✗
HD TV	720/1080p	720/1080p	720/1080p
Media Hub	Automático	Manual	✗
Set-top Box	Externa	Externa	Interna
Remote App	✓	✓	✗
Mercado / Loja de aplicações	✗	Android Market	✗

Através da Tabela 3, podemos observar resumidamente, as diferenças entre as três soluções oferecidas pela Nonius Software.

4. LG PRO: CENTRIC

Este capítulo vai-se debruçar sobre a televisão LG Pro:Centric, o modelo utilizado no âmbito do desenvolvimento da tese, referindo-se à tecnologia e linguagem suportada e à sua API (*Application Programming Interface*). Será também feita uma descrição da estrutura da interface gráfica, assim como de todos os serviços disponíveis, denominados ao longo deste relatório como *plugins*.

4.1. TECNOLOGIA E LINGUAGEM

A tecnologia utilizada nas televisões LG Pro: Centric é o FLASH que permite a criação de animações interactivas. É usual chamar apenas de FLASH aos arquivos gerados pelo Adobe Flash, ou seja, a animação em si. Esses arquivos têm a extensão ".swf" (*Shockwave Flash File*) e podem ser visualizados numa página web utilizando um *browser* que o suporta ou através do *Flash Player* que é uma aplicação distribuída gratuitamente pela Adobe. O FLASH permite ainda o desenvolvimento de aplicações complexas utilizando a linguagem ActionScript.

O ActionScript é uma linguagem de programação orientada a objectos que está actualmente na sua terceira versão, também denominada como AS 3.0. No entanto, a televisão LG Pro: Centric apenas suporta a versão 2.0.

A primeira versão do ActionScript foi lançada com a versão 5 do FLASH em 2000, tendo as actions do FLASH 4 passando a ser chamadas de ActionScript. O nome ActionScript 1.0 apenas foi adoptado após a introdução do ActionScript 2.0.

O ActionScript 2.0 surgiu em 2003 com o lançamento do FLASH MX 2004 e do Flash Player 7, onde foram adicionados mais objectos, métodos e outros elementos à linguagem. Esta versão permite o uso de uma abordagem orientada a objectos para o desenvolvimento de aplicações [20].

Em 2006 foi lançada a terceira versão, o ActionScript 3.0, juntamente com o Adobe Flex 2.0 e o Adobe Flash Player 9. Esta versão vai além dos recursos de script das suas versões anteriores. Foi desenvolvida para facilitar a criação de aplicações altamente complexas com grandes conjuntos de dados e bases de código reutilizáveis orientadas a objectos. O código do ActionScript 3.0 pode ser executado até dez vezes mais rápido do que as versões anteriores [21].

O código ActionScript pode ser desenvolvido e editado com recurso a diversos editores com suporte a ActionScript. Além dos *softwares* da Adobe, actualmente o Adobe Flash Professional C6, existem outras soluções gratuitas para o mesmo efeito. O utilizado para o desenvolvimento do projecto foi o Flash Develop. Apesar da grande vantagem de ser um ambiente de desenvolvimento *opensource*, contém outras vantagens como por exemplo o sistema de *Intellisense*. É um sistema que é capaz de completar o código automaticamente, além de lhe oferecer ajuda sobre recursos, referências e sintaxe de métodos e propriedades [22].

4.2. LG PRO: CENTRIC API

A televisão LG Pro: Centric tem uma API própria que contém duas classes: `IHcap` e `IHcapConst`.

A classe `IHcap` oferece métodos relacionados com:

- Os canais de televisão;
- Vídeo;
- Informação da televisão;
- Controlo do tempo;
- Controlo do áudio;

- Funcionalidades da televisão (desligar, reiniciar).

Quanto à classe `IHcapConst` define constantes para a classe `IHcap` tais como os códigos dos botões do comando.

4.3. MODELO DA INTERFACE GRÁFICA

O sistema Nonius TV permite ao cliente a personalização da interface gráfica dos menus. Neste projecto, irá ser demonstrado a personalização da interface gráfica utilizada pela Nonius Software.

Antes do menu principal ser apresentado, é feito o carregamento das *playlists* que contêm os canais de televisão, as rádios, do *video-on-demand* entre outras coisas. No decorrer do processo aparecem mensagens de *loading*, assim como um logotipo que pode ser personalizado.

Após o carregamento, o menu principal irá aparecer tal como a Figura 11 ilustra. Todo este menu pode ser personalizado, desde o *background* aos ícones dos *plugins*.

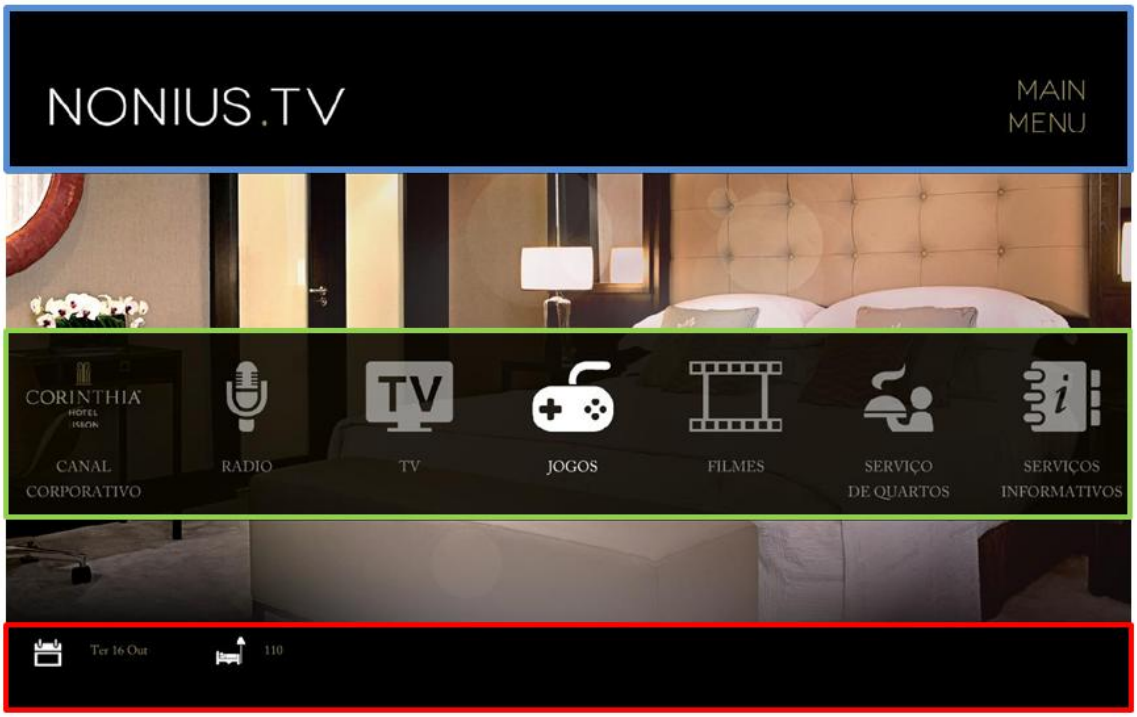

Figura 11 Menu principal

A interface gráfica do menu principal é constituída pelos seguintes elementos:

- ***Background*:** é um elemento fulcral na interface, sendo comum em todos os *plugins*, condicionando todos os restantes elementos, pois a escolha dos mesmos deverá enquadrar-se com o background escolhido. O fundo terá que ter como largura 1360 pixéis e de altura 768 pixéis. O seu formato é PNG (*Portable Network Graphics*) pois é o suportado pela televisão.
- **Cabeçalho:** está representado pela cor azul na Figura 11. As suas dimensões são 1360 pixéis de largura e 160 pixéis de altura. O cabeçalho é constituído por um logotipo no canto esquerdo, por sua vez no lado direito encontra-se o nome do menu em questão. O nome do menu, aparece de acordo com o idioma do hóspede. Ou seja, suporta multilingue. De momento suporta Alemão, Espanhol, Francês, Inglês e Português.
- **Menu de serviços:** está representado pela cor verde na Figura 11. Este elemento contém os ícones de todos os serviços disponibilizados. Alterando a visibilidade dos mesmos conforme se encontravam seleccionados ou não. As dimensões dos ícones devem ser 180 pixéis de largura e 280 pixéis de altura.
- **Rodapé:** está representado pela cor encarnada na Figura 11. Este elemento é composto no seu canto esquerdo por dois ícones e duas caixas de texto que indicam a data e o número do quarto. A dimensão dos ícones utilizados é de 40px40p. No canto direito aparecerá também um ícone que representa o número de mensagens não lidas. Existem vários ícones para esta funcionalidade tal como a Figura 12 ilustra, que serão apresentadas conforme a situação.

Figura 12 Diferentes ícones para as mensagens não lidas

Para a navegação nos menus dos respectivos serviços disponibilizados foi necessário a criação de um ícone *up*, *down*, *left* e *right* cujas dimensões serão 40px40p, tal como a Figura 13 ilustra. Quando é possível navegar para qualquer uma das direcções referidas acima, a visibilidade da seta está ao máximo, caso contrário esta não aparece pois a sua visibilidade será nula.

Figura 13 Ícones de down/up/left/right

Existem vários serviços que possuem diversas funcionalidades, ou seja, implica o uso de vários botões do comando. Nesses casos é colocada uma barra de opções no rodapé do menu, onde são apresentados todos os botões que podem ser utilizados no respectivo plugin, tal como representado na Figura 14.

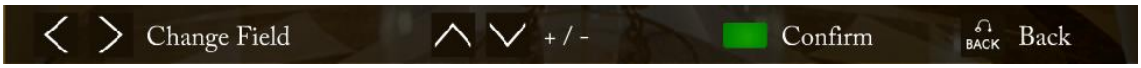

Figura 14 Barra de opções

4.4. SERVIÇOS DISPONÍVEIS

Os serviços disponibilizados pela solução de televisão interactiva LG Pro: Centric são os seguintes:

- **Canal Corporativo**: este serviço permite colocar um vídeo de apresentação do hotel ou um canal de televisão, sendo a escolha feita pelo cliente. Contém também um pequeno *feed* de notícias que será apresentado de acordo com a nacionalidade do hóspede. A Figura 15 ilustra o serviço descrito.

Figura 15 Canal corporativo

- **Rádio**: este serviço disponibiliza ao hóspede a oportunidade de escutar rádio. As *playlists* que contém os logótipos das rádios e as suas *streams* podem ser personalizadas.
- **Canais de TV**: este serviço disponibiliza ao hóspede a oportunidade de poder assistir canais de televisão. Tal como para o serviço de rádios, as *playlists* são personalizáveis. É possível criar uma *playlist* por idioma, sendo posteriormente carregada de acordo com a nacionalidade do hóspede. O menu que permite o hóspede seleccionar o canal de televisão desejado é composto por uma barra central que contém os ícones representantes de cada canal, tal como a Figura 16 demonstra.

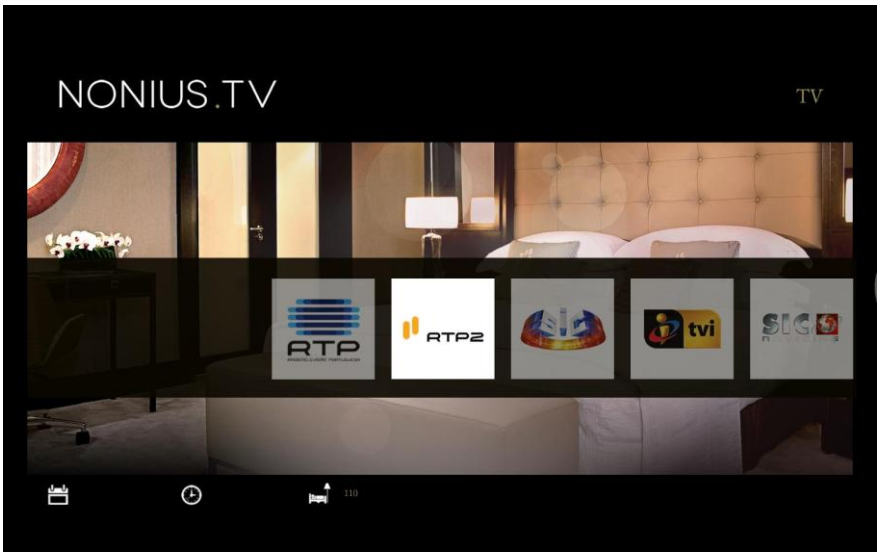

Figura 16 Menu do serviço que disponibiliza os canais de televisão

- **Jogos**: O menu deste serviço é semelhante ao da TV e do rádio. Permite ao hóspede seleccionar de uma lista de jogos qual o que pretende jogar.
- ***Video-on-Demand***: O serviço de VoD permite ao hóspede escolher dentro de vários géneros, o filme que pretende visualizar. Permite uma secção de filmes de adulto que estará bloqueada ou não, conforme a escolha do hóspede. Este serviço será explorado com mais detalhe ao longo da dissertação.
- **Serviço de Quartos:** Este serviço permite ao hóspede escolher produtos dentro de várias categorias, tais como, produtos de higiene, bebidas, etc. e efectuar a sua compra, tal como a Figura 17 ilustra.

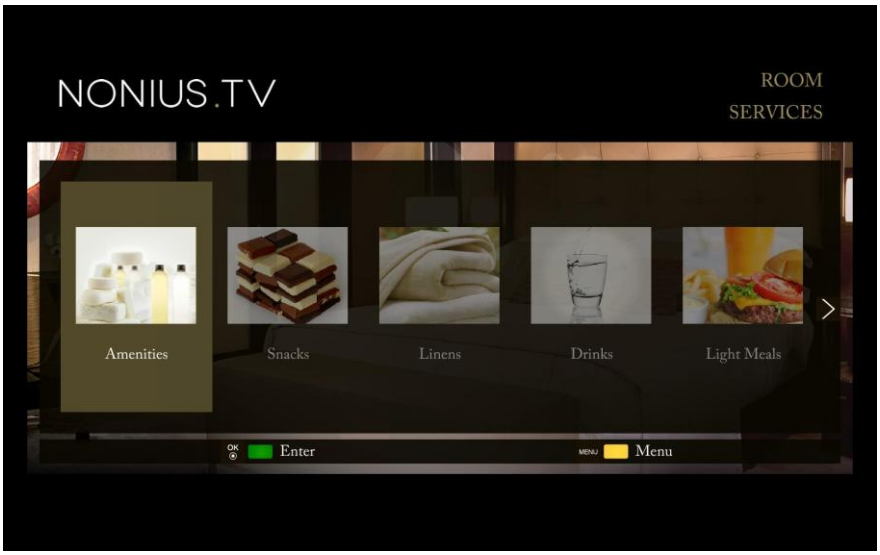

Figura 17 Serviço de quartos

- **Desportos e Lazer, Directório de Serviços, Serviço de Hóspedes:** estes serviços em questões de interface gráfica são semelhantes, apenas o conteúdo é que se altera. Oferecem serviços como SPA, aluguer de viaturas, golfe, visitas a museus, marcações em restaurantes e informações do hotel em que o hóspede pode comprar serviço ou colocar numa lista de interesses.
- **Serviços Informativos:** Disponibiliza serviços informativos tais como: voos, mapas, notícias, farmácias e meteorologia**.** A interface gráfica deste serviço está ilustrada na Figura 18.

Figura 18 Serviços informativos

- **Conta:** Este serviço permite ao hóspede ver o total das suas despesas, assim como uma lista de todas as compras efectuadas. Oferece a opção de *express checkout* que será explicada no decorrer do relatório.
- **Mudança de Idioma:** Permite ao hóspede alterar o idioma de todos os menus, tendo como opção cinco idiomas (Alemão, Espanhol, Francês, Inglês e Português);
- **Caixa de Mensagens:** Serviço que permite ao hóspede consultar as mensagens enviadas pelo hotel, tais como mensagem de boas-vindas, promoções, eventos, etc. Permite também ao hóspede eliminar as mensagens lidas.
- **Inquérito:** Este serviço gera um número de questões relacionadas com a qualidade dos serviços oferecidos pelo hotel, assim como das suas condições.
- **Alarme:** Caso o hóspede necessite de acordar a determinada hora, este é o serviço utilizado para definir a hora a que pretende ser acordado, tal como a Figura 19 ilustra. A televisão irá iniciar caso esteja desligada e um som de alarme irá começar a tocar de forma gradual.

Figura 19 Serviço de alarme

- **Eventos**: Este *plugin* permite ao hóspede saber se têm algum evento associado durante a sua estadia no hotel. Caso esteja associado, detalhes sobre o evento irão ser apresentados em forma de texto e imagens/vídeo.
- ***Setup Wizard***: Este serviço não está disponível ao hóspede, pois apenas pode ser acedido por uma sequência secreta de botões do comando que permite registar a box da televisão num determinado quarto, podendo também ver informações da televisão. Sendo bastante útil para a equipa técnica.

4.5. ESTRUTURA DOS TEMAS

O tema é constituído por directórios e ficheiros XML, tal como a Figura 20 ilustra. No que diz respeito aos directórios, grande parte está dividido por *plugin* e contém imagens e ícones necessários ao menu em questão. Existem três directórios genéricos que são utilizados por todos os menus: *buttons*, *shared* e *title*.

O primeiro directório contém os ícones dos vários menus, enquanto o directório *shared* contém todos os ícones relativos aos botões do comando. No terceiro directório estão disponíveis as imagens de cabeçalho.

É igualmente importante verificar que neste tipo de estrutura não existe nenhum directório nem localização especificada para os ícones dos canais de TV, rádio e jogos. O directório correspondente a estes ícones encontra-se fora da estrutura dos temas, denominado *content*, sendo transversal a todos. Ou seja, para qualquer tema serão sempre utilizados os mesmos ícones de canais e jogos. Relativamente a imagens que não se alteram conforme o tema (imagens *default*), são colocadas no directório *commons*.

À parte dos directórios de categorias mencionados anteriormente, na própria raiz do tema encontram-se também algumas imagens. Entre estas imagens pode referir-se a imagem de *background*, as imagens de data e hora e número de quarto.

Ainda na própria raiz do tema existem alguns ficheiros XML, com várias definições, que incluem os tamanhos e fontes de texto, posições das imagens, e os caminhos para essas mesmas imagens que compõe o tema. O ficheiro "`theme.xml`" contém toda a informação relativa aos ícones dos menus, enquanto que o ficheiro "base.xml" tem a informação relativa ao menu principal, ou seja, *background* e imagens relativas à data, hora e número de quarto.

Existe também um directório, chamado *nivo_adapters*, onde estão vários ficheiros XML que contêm a informação relativa a farmácias, notícias e ao VoD, sendo a partir dai que irá ser extraída a informação para ser apresentada. Informação essa que no caso das farmácias consiste no seu nome, endereço, disponibilidade, etc.

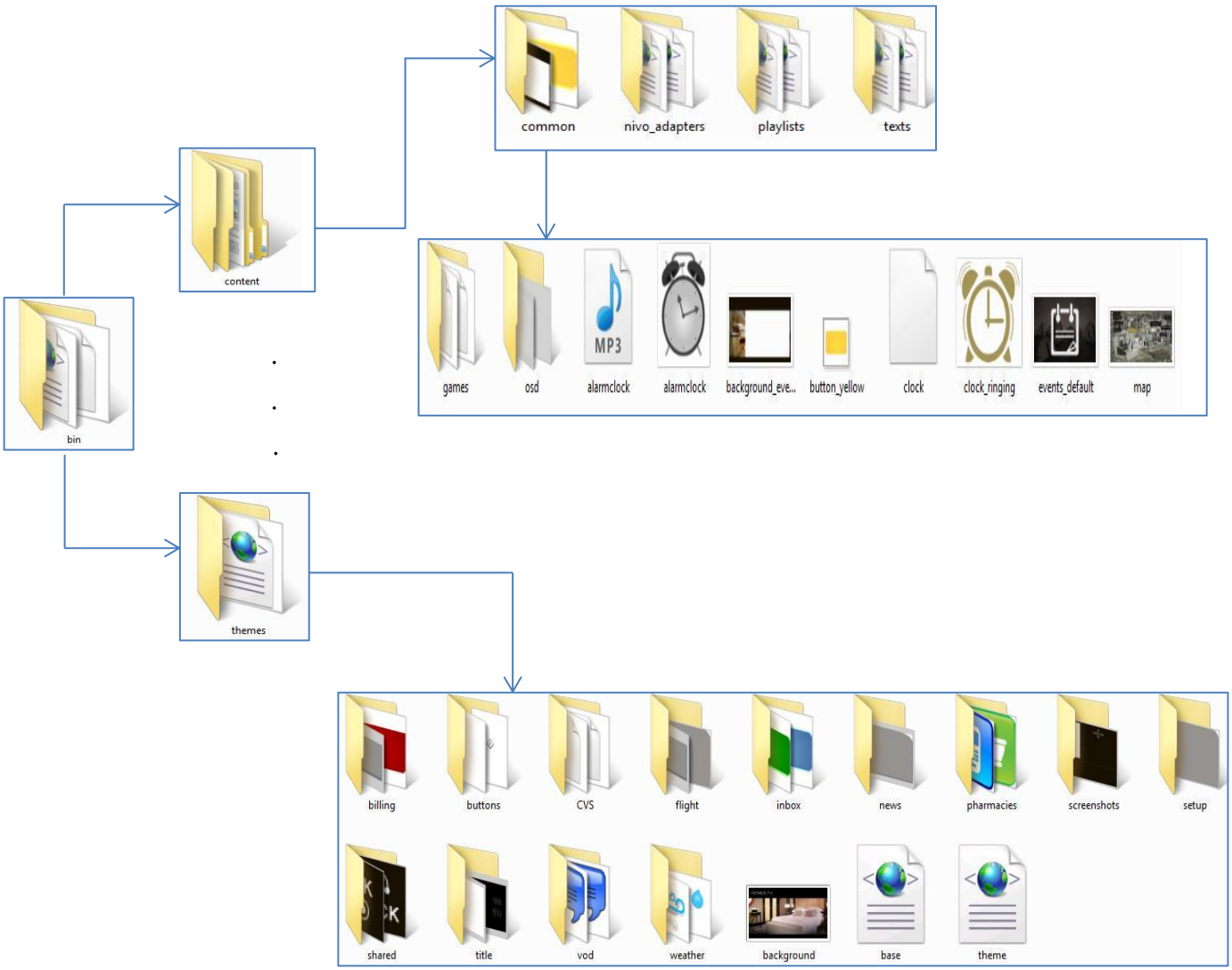

Figura 20 Estrutura dos temas

A Figura 20 ilustra a estrutura dos directórios que permitem criar a interface gráfica.

5. DESENVOLVIMENTO DO *SOFTWARE*

Neste capítulo irá ser abordado o desenvolvimento do *software*, tanto *frontend* (FE) como de *backend* (BE).

Os serviços que foram abordados no âmbito desta dissertação são os seguintes:

- Video on Demand (FE);
- Serviço de Quartos (BE);
- Desportos e Lazer (BE);
- *Setup Wizard* (FE);
- *Express Checkout* (BE);
- *Survey Threshold* (BE);
- Mensagem de Boas Vindas (FE & BE);
- Mensagem de Despedida (FE & BE);
- Eventos do Hotel (FE & BE).

Relativamente ao *frontend* a aplicação está dividida em diversas classes. A Tabela 4 apresenta as classes criadas/alteradas no desenvolvimento da aplicação.

Tabela 4 Classes criadas/modificadas e uma breve descrição do que foi implementado

Classe	Breve descrição da implementação
DateTime.as	Criação das janelas de mensagens de boas-vindas e de despedida
Main.as	*Parsing* das playlists de TV, rádio e VoD. Carregamento do tema.
NiVo_ActivitiesCategories.as	Serviço *web* que retorna todas as categorias
NiVo_ActivitiesProducts.as	Serviço *web* que retorna todos os produtos.
NiVo_ActivitiesReservation.as	Serviço *web* que submete os pedidos do hóspede
NiVo_Bill.as	Opção de *express checkout*
NiVo_Centric.as	Inicialização de todas as classes. Implementação de métodos que permitem entrada e saída em menus.
NiVo_Events.as	Tratamento dos serviços *web* relativos aos eventos
NiVo_MainMenu.as	Serviço *web* que retorna informação das mensagens de boas vindas e despedida
NiVo_RadioChannels.as	*Layout* do menu dos rádios e reprodução do rádio
NiVo_RoomServices.as	Serviço *web* que retorna as categorias dos serviços de quartos
NiVo_RoomServicesProducts.as	Serviços *web* que retornam os produtos e submetem os pedidos do hóspede
NiVo_Survey.as	Implementação do *survey threshold*
NiVo_Trailer.as	Implementação da opção trailer no serviço de *vídeo-on-demand*
NiVo_Variables.as	Definição de variáveis que são utilizadas por várias classes
NiVo_Vod.as	Implementação do *layout* do menu do VoD
NiVo_Wizard.as	Criação do *layout* e das funcionalidades do menu
PMS_Events.as	Criação de métodos relacionados com o PMS

SoapRequests.as	Criação dos métodos que permitem fazer os pedidos aos serviços *web*
TV_Controller.as	Atribuição dos métodos aos botões do comando
Utils.as	Contém métodos genéricos que podem ser utilizados por todas as classes
XML_Parser.as	*Parsing* dos ficheiros XML
Welcome_Message.as	Criação do *layout* e das funcionalidades da mensagem de boas vindas
Vod_Info_Pop_Up.as	Criação da janela que contém a informação do filme seleccionado no serviço de VoD
Vod_Code_Validator.as	Geração do código PIN e validação do mesmo
Radio_Station.as	Criação de uma janela que conterá uma imagem como *background* enquanto o hóspede escuta rádio
My_Events.as	Construção do *layout* da janela dos eventos
Express_Checkout_Pop_Up.as	Janelas de confirmação da opção de *express checkout*
Alert_Message.as	Construção do *layout* de janelas de *loading*
Last_Day_Message.as	Construção do *layout* e das funcionalidades da mensagem de último dia

O desenvolvimento realizado no *backend* debruçou-se fundamentalmente em três áreas: *Config*, *Management* e *WebServices*.

A interface *Config* permite configurar o *backend*, incluindo definições de rede, *upgrades*, licenças, etc. Está implementado em PHP (*Hypertext Preprocessor*).

A interface *Management* também implementada em PHP permite a configuração dos serviços disponibilizados na televisão e da personalização da interface gráfica.

São criados também os serviços web necessários para o funcionamento dos serviços oferecidos.

5.1. VIDEO-ON-DEMAND

Este serviço permite ao hóspede escolher dentro dos géneros disponíveis quais os filmes que pretende alugar durante a sua estadia.

Antes de qualquer desenvolvimento no *layout* do *plugin*, é necessário fazer o *parsing* do ficheiro XML que contém toda a informação dos filmes (género, nome do filme, duração, actores, custo, etc.). Na classe `XML_Parser` é criado um método denominado `parse_genre_vod` que recebe como argumentos o ficheiro XML e o idioma do hóspede. O resultado deste método será um vector com os filmes organizados por género, sendo cada posição do vector correspondente a um género diferente.

O método `parse_genre_vod` é chamado na classe `Main`, para que esta análise ao ficheiro XML seja feita no arranque da televisão.

Em relação ao *layout* do VoD, este foi realizado maioritariamente na classe `NiVo_Vod`. Em termos de *layout* é o serviço com maior complexidade.

Os *listeners* em ActionScript, acompanham o que acontece na aplicação FLASH, ficando à espera de uma acção específica. Quando o *listener* detecta a acção, o FLASH executa métodos em resposta ao respectivo evento [20]. Cada serviço contém um *listener* próprio e são criados na classe `NiVo_Centric`.

A classe `NiVo_VoD` contém dois métodos bastante úteis relativos ao *layout*. Um deles é o método `show`, em que são chamados todos os métodos relativos à construção do *layout*. Enquanto que o outro método, denominado `hide`, corresponde a chamadas de métodos que permitem esconder/destruir o *layout*, quando por exemplo é pretendido regressar ao menu principal.

Relativamente à construção do cabeçalho do *plugin* é utilizado o método `create_VOD_header` onde é feito o carregamento da imagem escolhida para ocupar a região de cabeçalho, assim como a criação de uma caixa de texto no canto superior direito que apresenta o nome do serviço no idioma do hóspede.

Para a construção da secção que contêm os géneros disponíveis, tal como a Figura 21 ilustra, criaram-se os seguintes métodos:

- **`create_genre_bar`**: Neste método serão invocados os métodos `create_bar_with_border`, `create_bar` e `create_genre` permitindo assim a construção das barras de géneros.
- **`create_bar_with_border`**: Método criado na classe `Utils` que permite a construção de um rectângulo. Permite escolher qual a dimensão do rectângulo, a sua

cor e a cor da linha de borda. Nesta situação é utilizado para criar a secção onde as barras de género serão inseridas.

- **`create_bar`**: Método criado na classe `Utils`, semelhante ao anterior com a diferença que não existe linha de borda. Utilizado para criar uma barra que indica qual o género que se encontra seleccionado.
- **`create_genre`**: O número de vezes que este método é chamado depende do número de géneros disponíveis. Cria uma caixa de texto com o nome do género e actualiza a posição vertical na qual o próximo género será inserido. Como podem existir filmes de teor mais adulto, é necessário existir um controlo parental, ou seja, por omissão estes géneros aparecem bloqueados. A verificação se estes géneros se encontram bloqueados é feito neste método, colocando a uma cor diferente o nome do género caso se encontrem bloqueados.

Figura 21 Barra de géneros

O número de géneros apresentados é limitado pois pode existir um número elevado deles. Sendo assim é apenas possível mostrar nove géneros de cada vez.

De modo a permitir a deslocação ascendente e descendente na secção de géneros, utilizaram-se os seguintes métodos:

- **`Down`**: Permite a deslocação vertical descendente. Este método é atribuído ao botão *down* do comando na classe `TV_Controller` utilizando a API da televisão. O Código 22 apresentado no Anexo B ilustra esta implementação.
- **`Slide_up`**: Caso o hóspede esteja com a barra de selecção no último género apresentado e existam mais géneros, ao pressionar para baixo no comando é necessário que ocorra um deslizamento, ou seja, o primeiro género apresentado será

escondido e os restantes géneros recuaram uma posição, mostrando assim na última posição outro género.

- **`Up`**: Permite a deslocação vertical ascendente. Este método é atribuído ao botão *up* do comando na classe `TV_Controller` utilizando a API da televisão.
- **`Slide_down`**: Este método é semelhante ao `slide_up` apenas o sentido do deslocamento é o contrário. Ou seja, se o hóspede se encontrar na situação final descrita no método `slide_up` ao pressionar para cima no comando, é necessário que ocorra um deslocamento, voltando a esconder o último género e os restantes avançaram uma posição, voltando a mostrar o género que tinha sido escondido da primeira posição.

Relativamente à secção que contém as capas dos filmes, esta depende da secção de géneros pois apresenta as capas dos filmes de acordo com o género seleccionado. Para a construção da secção das capas dos filmes, foram criados os seguintes métodos:

- **`create_image_bar`**: é invocado o método `create_bar` da classe `Utils` para criar a secção onde as capas dos filmes serão apresentadas. Para todos os filmes de cada género irá ser criado um `movieclip` onde será carregada a imagem do filme. Todos os `movieclips` serão armazenados num vector e organizados por género. Este método é invocado no método `show`.
- **`Show_movies`**: é necessário limitar o número de capas de filmes apresentadas (cinco capas de cada vez). Aqui serão colocados os `movieclips` criados no método anterior relativos ao género seleccionado, com a sua visibilidade activa e aplicado um `resize` nas capas, pois a capa seleccionada tem que ser maior que as restantes. Este método é chamado no método `show`, `up` e `down`.

A Figura 22 apresenta o resultado final da secção que contém as capas dos filmes.

Figura 22 Apresentação das capas dos filmes

Relativamente ao deslocamento nesta secção, é bastante semelhante ao referido na secção dos géneros com a diferença que o deslocamento é horizontal. Foram criados os seguintes métodos:

- **`Left`**: Permite o deslocamento horizontal para a esquerda. Este método é chamado na classe `TV_Controller`, sempre que o botão com a seta para a esquerda do comando for pressionado.
- **`Slide_left`**;
- **`Right`**: Permite o deslocamento horizontal para a direita. Este método é chamado na classe `TV_Controller`, sempre que o botão com a seta para a direita do comando for pressionado.
- **`Slide_right.`**

É necessário existir uma secção que contém informações do filme seleccionado, tais como o nome, duração, descrição e preço. Para esta secção, os métodos implementados foram os seguintes:

- **`Create_info_bar`**: é invocado o método `create_bar` da classe `Utils` para criar a secção onde a informação do filme será apresentada. Toda a informação será guardada em vectores organizados por género.
- **`Text_define`**: São criadas as caixas de texto que contêm a informação do filme seleccionado. Este método será chamado nos métodos `show`, `right`, `left`, `up` e `down`. Pois sempre que ocorrer uma deslocação seja no género ou nos filmes a informação irá se alterar.

O resultado final da construção do *layout* é apresentado na Figura 23.

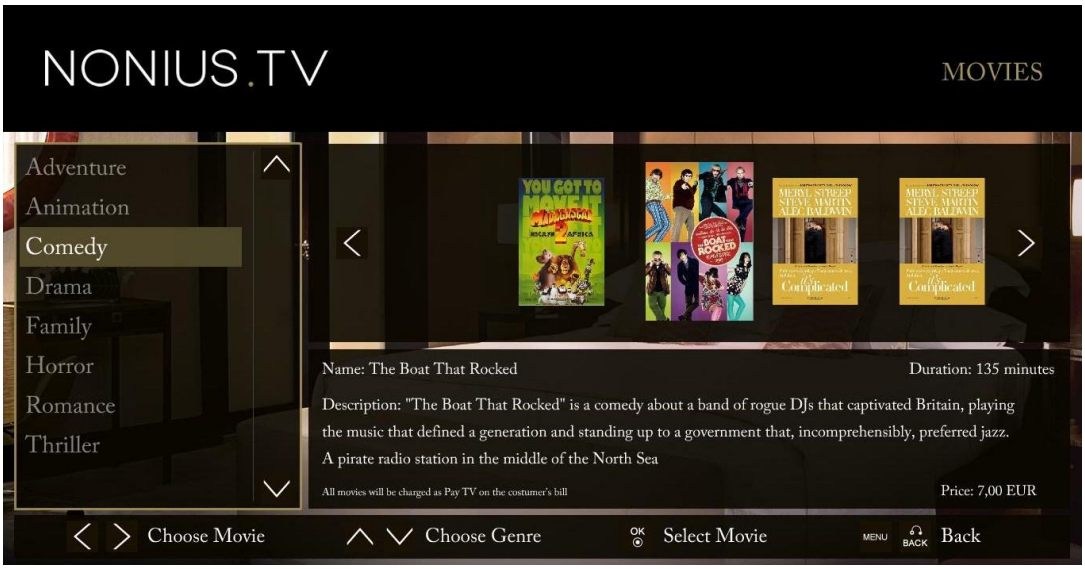

Figura 23 Resultado final do serviço de *Video on Demand*

Após o hóspede verificar todos os filmes disponíveis, ao pressionar no botão Ok provocará o aparecimento de uma janela (*pop up*) que terá novamente a capa do filme escolhido, informação um pouco mais detalhada do filme e três botões. Esses botões são representados por três cores, verde, azul e vermelho. Esta janela foi criada numa outra classe denominada `Vod_Info_Pop_Up` que contém os seguintes métodos:

- **`vod_info_pop_up`**: São recebidas todas as propriedades necessárias para a criação da janela e o vector que contém toda a informação do filme. Assim como a atribuição de funcionalidades aos três botões, pois a classe possui o seu próprio *listener*, não sendo necessário definir na classe `Tv_Controller`. Este método é invocado quando o botão Ok é pressionado.
- **`Show`**: Todo o processo de criação e apresentação da janela é realizado neste método.
- **`Destroy`**: Destrói a janela de informação.

Relativamente aos botões inseridos na janela, as suas funcionalidades são as seguintes:

- **Vermelho**: Destrói a janela e volta ao menu base do serviço;
- **Azul**: Permite ao hóspede visualizar um *trailer* do filme escolhido;
- **Verde**: Ao ser pressionado, é feita uma verificação se o filme foi comprado anteriormente. Caso seja a primeira vez irá disparar uma nova janela onde pedirá ao hóspede a inserção de um código.

A verificação se o filme foi comprado anteriormente é feita utilizando o método `vod_payment` na classe `NiVo_Vod`. Este método receberá um ficheiro XML e será feita a análise do mesmo. Existem três tipos de resposta possíveis:

- **Ok**: Significa que o código foi inserido correctamente e avança para a reprodução do filme. É também guardado no ficheiro `purchases.php` o id do filme comprado.
- **No_PIN**: O filme já foi comprado anteriormente e começará imediatamente a reprodução do filme;
- **Error**: Caso seja este o tipo de resposta, irá aparecer uma janela onde explicará ao hóspede o problema. Existem três tipos de erros: o hóspede estar *check-out*, problemas no *stream* do vídeo ou a não existência de rede.

No caso de nenhuma destas respostas ser obtida, então é chamado o método da classe `Vod_code_validator` que irá apresentar uma janela onde pedirá ao hóspede para inserir um código. Para a geração deste código foi criado o método `randRange` na classe `Utils` que permite gerar códigos entre 1000 e 9999.

Na classe `Vod_code_validator` é atribuído a todos os botões numéricos um método que permite acrescentar a uma *string* todos os algarismos pressionados. Após o hóspede inserir os quatro algarismos do código, ao pressionar Ok irá ser invocado um método que verifica se o código inserido está correcto. Se o código estiver incorrecto então um novo código será gerado, caso contrário a janela irá ser destruída e é invocado novamente o método `vod_payment`.

O resultado do aparecimento da janela de informação e de inserção de código pode ser visualizado na Figura 24.

Figura 24 Ecrã das janelas de informação e inserção de código

Na interface existente do *Backend* o hotel poderá definir o tipo de pagamento para o sistema de *Video on Demand*. Existe a opção de *Per Movie* e *Per Stay*. Na primeira opção, o hóspede irá pagar apenas os filmes que vir e terá acesso aos mesmos durante 24 horas, enquanto que na segunda opção, o hóspede pagará apenas uma quantia e terá acesso a todos os filmes até o seu *check- out* ser realizado.

A caixa de texto relativa ao preço do filme deverá ser actualizada conforme o tipo de pagamento escolhido pelo hotel. A lógica deste processo é apresentada na Figura 25.

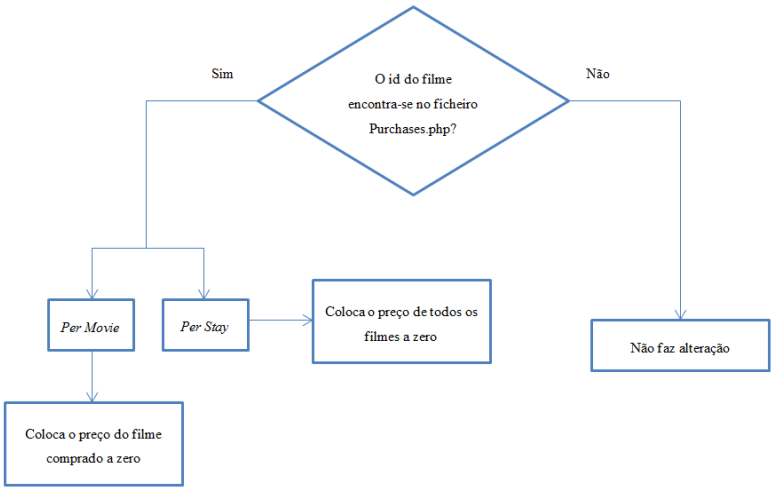

Figura 25 Fluxograma representativo do tipo de pagamento do sistema de VoD

Ao longo da reprodução do filme, o hóspede tem ao seu dispor algumas funcionalidades, tais como *play*, *pause*, *fast forward*, *fast rewind* e activação de legendas. Para que o hóspede tenha noção de qual a funcionalidade que seleccionou, no canto superior direito será apresentada durante quatro segundos, uma imagem associada à funcionalidade escolhida. As imagens estão ilustradas na Tabela 5.

Tabela 5 Imagens utilizadas para a demonstração das funcionalidades do sistema de reprodução de vídeo

Play	Pause	Fast Forward	Fast Rewind	Legenda on	Legenda off
					

A criação destas imagens é feita no método `button_pressed` e é chamado em todos os métodos da classe `RTSP_Client` que permitem que essas funcionalidades aconteçam.

No canto superior esquerdo, tal como a Figura 26 ilustra, de forma semelhante aos ícones aparece um *timestamp* que mostra o tempo actual do filme sempre que alguma dessas funcionalidades de vídeo é executada.

Figura 26 Timestamp

5.2. SERVIÇO DE QUARTOS

O *plugin* serviço de quartos permite ao hóspede fazer algumas compras que lhe serão entregues no seu quarto, tais como produtos de higiene, alimentos, bebidas, etc.

Para que o serviço de quartos tenha todas as suas funcionalidades foi necessário fazer alterações no *backend*.

O serviço de quartos está dividido em três fases:

- **Categorias:** Primeiramente são apresentadas ao hóspede todas as categorias disponibilizadas pelo hotel relativamente ao serviço de quartos.
- **Produtos:** Quando o hóspede selecciona uma categoria, posteriormente são apresentados todos os produtos existentes relativos à categoria seleccionada. O hóspede pode escolher a quantidade e poderá adicionar mais produtos à sua lista de reservas.
- **Reserva:** No momento em que o hóspede decide que não pretende seleccionar mais produtos, é apresentada uma tabela que contém todos os produtos seleccionados. Se o hóspede pressionar o botão verde, todos os seus pedidos serão submetidos e adicionados às suas despesas.

Tal como foi referido na secção relativa aos serviços *web*, este é constituído por vários elementos, tendo sido desenvolvido na linguagem PHP utilizando um grupo de classes que permitem criar e consumir serviços *web* em SOAP (`nuSoap`).

De forma a obter uma melhor organização, o desenvolvimento do código dos serviços *web* encontra-se dividido em cinco ficheiros:

- **`registers.php`**: São feitos os registos dos serviços *web*.

- **`complex_types.php`**: Neste ficheiro são criados objectos que podem conter mais que uma propriedade.
- **`methods.php`**: São desenvolvidos os serviços.
- **`functions.php`**: Utiliza-se este ficheiro para criar métodos auxiliares que serão utilizados no desenvolvimento dos serviços no ficheiro `methods.php`.
- **`index.php`**: Faz o *include* da classe `nuSoap` e do ficheiro `functions.php`. É feita a configuração do WSDL, assim como o *include* dos restantes ficheiros (métodos, registos e tipos complexos). São também processados os pedidos para tentar invocar os serviços.

Tendo em conta as três fases do serviço de quartos, e para que estas ocorram sem problemas são necessários três serviços *web*:

- **`getRoomServicesCategories`**: Um dos primeiros passos no desenvolvimento de um serviço web é o seu registo. Sendo assim, no ficheiro `registers.php` é colocado o Código 20, em que o primeiro argumento corresponde ao nome do serviço, o segundo ao nome e tipo de argumento de entrada e o último está relacionado com o tipo de retorno do serviço.

```
$server->register(
   'getRoomServicesCategories',
   array('language' => ' xsd:string'),
   array('return' => 'tns:RoomCategories')
);
```

Código 20 Registo de um serviço

O método criado para listar todas as categorias tem como argumento de entrada o idioma do hóspede, ou seja, quando o pedido for feito terá que ser enviado do lado do cliente esse parâmetro. Toda a informação relacionada com as categorias, produtos e submissão de pedidos é armazenada na base de dados, tendo sido necessário criar todas as tabelas necessárias e relacioná-las da melhor forma. A estrutura da base de dados relativa ao serviço de quartos é apresentada na Figura 51 do Anexo A.

As tabelas criadas são preenchidas através da interface do *backend* tal como ilustrado nas Figuras 27 e 28, sendo inseridos todos os detalhes relativos às categorias e produtos.

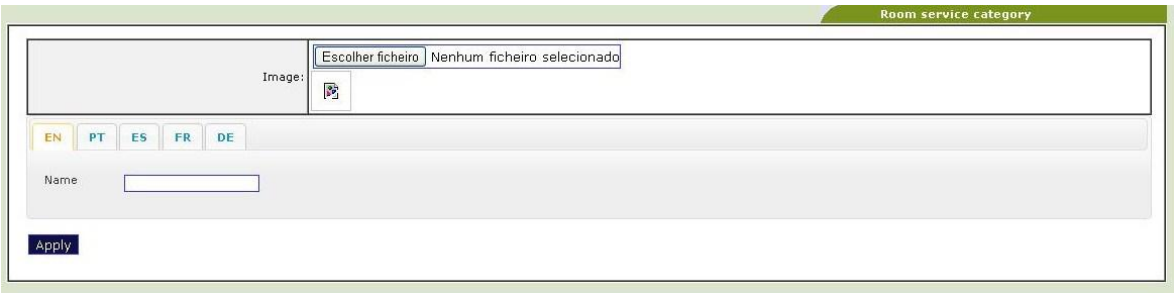

Figura 27 Criação de uma categoria do serviço de quartos através da interface do *backend*

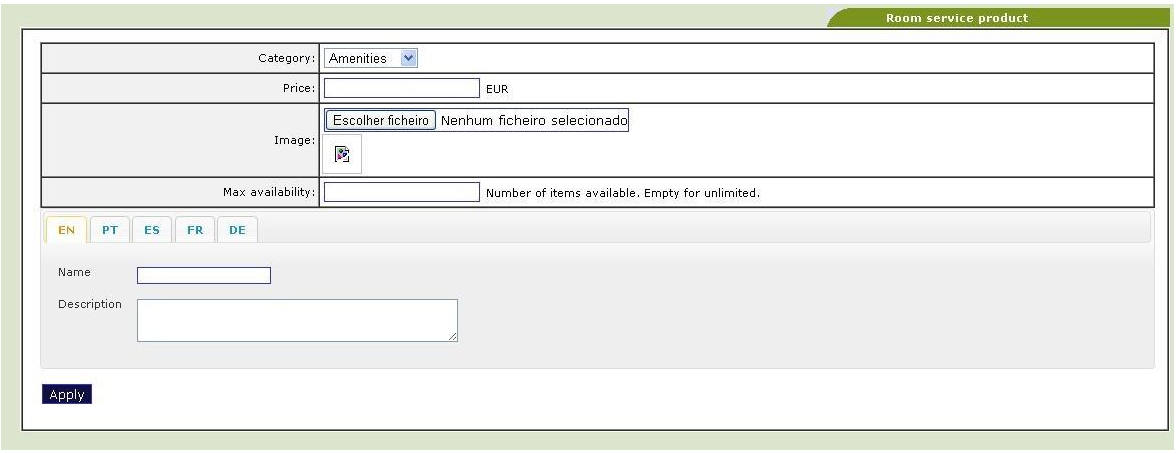

Figura 28 Criação de um produto do serviço de quartos através da interface do *backend*

Para obter todas as componentes que constituem a categoria (id, nome, imagem, idioma) é necessário construir uma *query* SQL (*Structured Query Language*) que permita retornar todos esses parâmetros da base de dados. A *query* deve retornar as componentes das categorias de acordo com o idioma do hóspede. Para preparar e executar a *query* SQL, são utilizados métodos da classe `PDO` (*PHP Data Objects*). Quando a *query* é executada, é retornado um vector que contém o conjunto de resultados.

O resultado deste serviço não é do tipo primário, ou seja é necessário construir o tipo complexo no ficheiro `complexTypes`. É desenvolvida a estrutura de dados de cada categoria e posteriormente armazenada num vector. Este vector conterá a estrutura de dados de todas as categorias existentes. O processo de criação dos tipos complexos é apresentado no Código 23 no Anexo C.

- **`getRoomServicesProducts`**: O desenvolvimento deste serviço é bastante semelhante ao anterior. A grande diferença encontra-se na *query* de selecção, pois de acordo com o idioma recebido, é feita a selecção dos vários componentes que

constituem o produto (id, id da categoria, imagem, preço, quantidade, nome, descrição e idioma). O registo do serviço é realizado da mesma forma.

- **`setRoomServicesOrders`**: Serviço correspondente à submissão de pedidos e à sua taxação. Este método tem como argumentos de entrada um ficheiro XML que contém todos os pedidos relativos ao serviço de quartos e o idioma. É necessário saber o número do quarto pois é a ele que será associada a compra dos produtos. O ficheiro XML apenas contém o id do produto e a quantidade requerida pelo hóspede.

É criado um método no ficheiro `functions.php` que irá fazer a submissão do pedido tendo como argumento o vector que contém a informação sobre o produto submetido (id, quantidade, nome, preço e descrição), o número do quarto e o idioma. Este método irá ser chamado e o seu retorno (booleano) é guardado numa variável. Antes de o pedido ser submetido, passa por uma serie de condições, tais como, se a quantidade pretendida é superior a zero ou se existe a quantidade necessária do produto em *stock*. Se alguma destas condições falhar, uma mensagem de erro aparecerá ao hóspede. Caso contrário, a taxação continua o seu processo sendo adicionado na tabela respectiva os pedidos. É construída e enviada para o PMS uma mensagem do tipo PS. Não existe problema se o processo de compra falhar, pois o pedido já foi adicionado à base de dados e contém uma *flag* que indica que os pedidos foram entregues ao PMS. Para o processo de compra concluir correctamente, o PMS deverá responder ao PS enviado com uma mensagem do tipo PA em que o parâmetro AS deverá ser OK.

Do lado do cliente, é criada a classe `SoapRequests` onde são construídos os métodos que permitem fazer os pedidos. A Figura 29 ilustra em que classes são feitos os pedidos dos serviços.

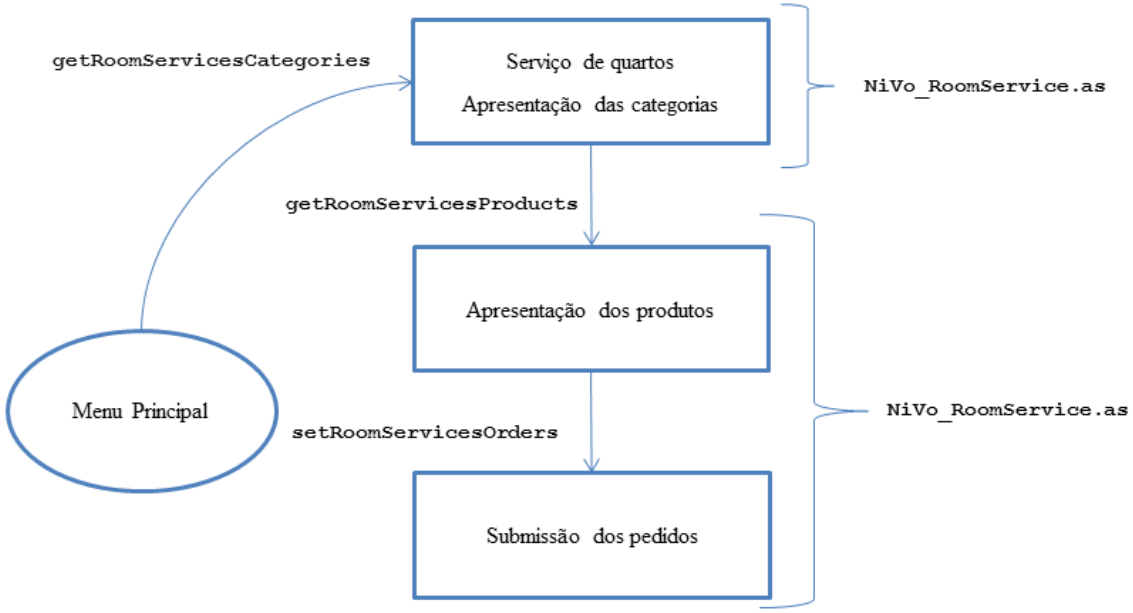

Figura 29 Lógica associada ao pedido de serviços *web* no serviço de quartos

5.3. DESPORTOS E LAZER

Este *plugin* é em tudo semelhante ao dos serviços de quarto, tanto em questões de *layout* como em serviços *web*.

São necessários três serviços *web* que irão listar as categorias, os produtos e posteriormente a submissão dos pedidos.

Neste plugin não existem apenas produtos. Contêm também locais, monumentos, edifícios que o hóspede pode seleccionar como interesse. Continua a existir produtos que é necessário efectuar a compra, como aluguer de automóveis, bicicletas, tratamentos num SPA, fazer uma reserva no restaurante.

Tal como se pode verificar na Figura 52 do Anexo A, a estrutura da base de dados é bastante mais complexa do que a de serviços de quartos, pois existe uma maior descrição de produtos. A grande diferença ocorre na existência de uma tabela denominada `super_categories`. Isto significa que em vez de um *plugin* apenas denominado desportos e lazer, podemos criar mais dois: directório de hóspedes e serviço de hóspedes.

Uma das super categorias é designada de directório de serviços, onde se encontram informações do hotel, emergências, pontos turísticos. A outra é o serviço de hóspedes onde estão todos os serviços oferecidos pelo hotel, como por exemplo, spa, restaurantes, actividades.

Os métodos dos serviços web são construídos da mesma forma, apenas com a diferença que entrará como argumento em todos eles, uma variável denominada `type` que dirá qual a super categoria escolhida e a partir dai saberá de quais tabelas ir buscar ou adicionar a informação. O valor da variável `type` é enviado pelo cliente. Se o valor desta variável for vazio, será relacionada ao *plugin* desportos e lazer. Apenas um aparte, a tabela `sport_and_leisure_produtcs` contém um campo denominado *action*. É este campo que irá ditar se é um produto de reserva ou de interesse. A inserção nas tabelas é feita através da interface do *backend*, tal como as Figuras 30 e 31 apresentam.

Figura 30 Criação de uma categoria do serviço de hóspedes através da interface do *backend*

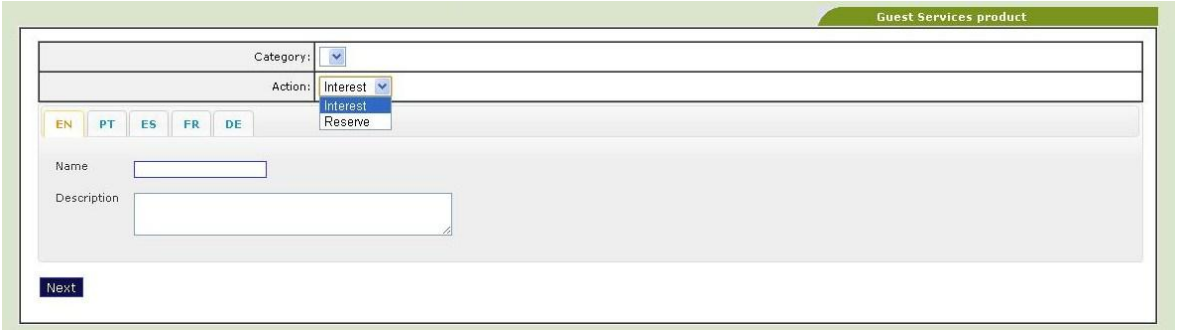

Figura 31 Criação de um produto do serviço de hóspedes através da interface do *backend*

A taxação dos produtos é feita da mesma forma que o serviço de quartos. O pedido do serviço de taxação é feito na classe `NiVo_ActivitiesReservation`.

Foi necessário criar um novo serviço web que permite seleccionar todas as imagens relativas a um produto. Essas imagens estão no caso dos desportos e lazer, na tabela

`sports_and_leisure_product_media`. Este serviço retorna um vector de imagens. O pedido deste serviço é feito na classe `NiVo_ActivitiesProducts`, quando o hóspede estiver a escolher os produtos e pretender que lhe sejam apresentadas mais imagens do produto.

5.4. *SETUP WIZARD*

Este serviço não se encontra disponível ao hóspede. É utilizado pelos técnicos para fazer o registo da *box*, seleccionando a classe e o número do quarto que serão associados à *box*. Contém também uma secção que apresenta informações sobre a televisão tais como, o seu IP, a classe seleccionada, o número de quarto, o modelo da TV e a versão de *firmware*. Este serviço apenas aparece no arranque da televisão, quando a classe seleccionada ainda se encontra em *default*. Outra forma de aceder a este serviço é através de uma combinação de botões num limitado espaço de tempo.

Para a criação do *layout* foi necessário criar uma classe denominada `NiVo_Wizard` onde são construídos todos os métodos necessários para obter as funcionalidades desejadas.

Esta classe inclui os seguintes métodos:

- **`Settings`**: Neste método são criadas as caixas de texto onde serão escritas a classe e o número do quarto. São também criadas duas secções utilizando o método `create_bar` da classe `Utils` onde aparecerá a classe e o número de quarto seleccionados. São realizados pedidos a dois serviços que informem qual a classe e o número de quarto seleccionados.
- **`Pop_up_class`**: Quando a selecção se encontra na classe, ao ser pressionado OK no comando, irá gerar uma *pop up* que apresentará todas as classes disponibilizadas, podendo assim escolher a classe desejada. As classes apresentadas resultam de um pedido a um serviço que retorna todas as classes existentes.
- **`Choose_class`**: A classe que tiver sido seleccionada na *pop up*, passará a ser apresentada na secção destinada à classe escolhida na base do plugin.
- **`Choose_roomnumber`**: É atribuído aos botões numéricos do comando um método que permite escrever na secção destinada ao número de quarto seleccionado.
- **`Pop_up_setup`**: Quando a selecção se encontra na *string Setup*, ao ser pressionado OK no comando, irá gerar uma *pop up* que apresentará alguma informação sobre a televisão.

- **`Content_setup`**: Neste método são criadas as caixas de texto que contêm todas as informações. Para obter informações da televisão, tais como o MAC, o seu modelo e a versão de *firmware*, utilizou-se a API da televisão.
- **`Show_instructions`**: É criada uma secção que conterá instruções de como utilizar este *plugin*.

Todas as deslocações possíveis neste serviço são verticais podendo ser ascendentes ou descendentes. Os métodos criados para que esses deslocamentos sejam possíveis são semelhantes aos do serviço de *vídeo on demand*:

- **`Up`**: permite a deslocação vertical ascendente.
- **`Down`**: permite a deslocação vertical descendente.
- **`Class_up`**: permite a deslocação vertical ascendente na janela que contém as classes disponíveis.
- **`Class_down`**: permite a deslocação vertical descendente na janela que contém as classes disponíveis.
- **`Setup_up`**: permite a deslocação vertical ascendente na janela que contém informações da televisão.
- **`Setup_down`**: permite a deslocação vertical descendente na janela que contém informações da televisão.

Quando a selecção se encontra na *string Save* e é pressionado o botão OK do comando, faz com que todas as definições sejam guardadas, ou seja, são chamados os serviços `setClass` e `setRoom` que guardam qual a classe e quarto que foram definidos. Posteriormente, a televisão irá reiniciar com as definições efectuadas.

O resultado do plugin está ilustrado na Figura 32.

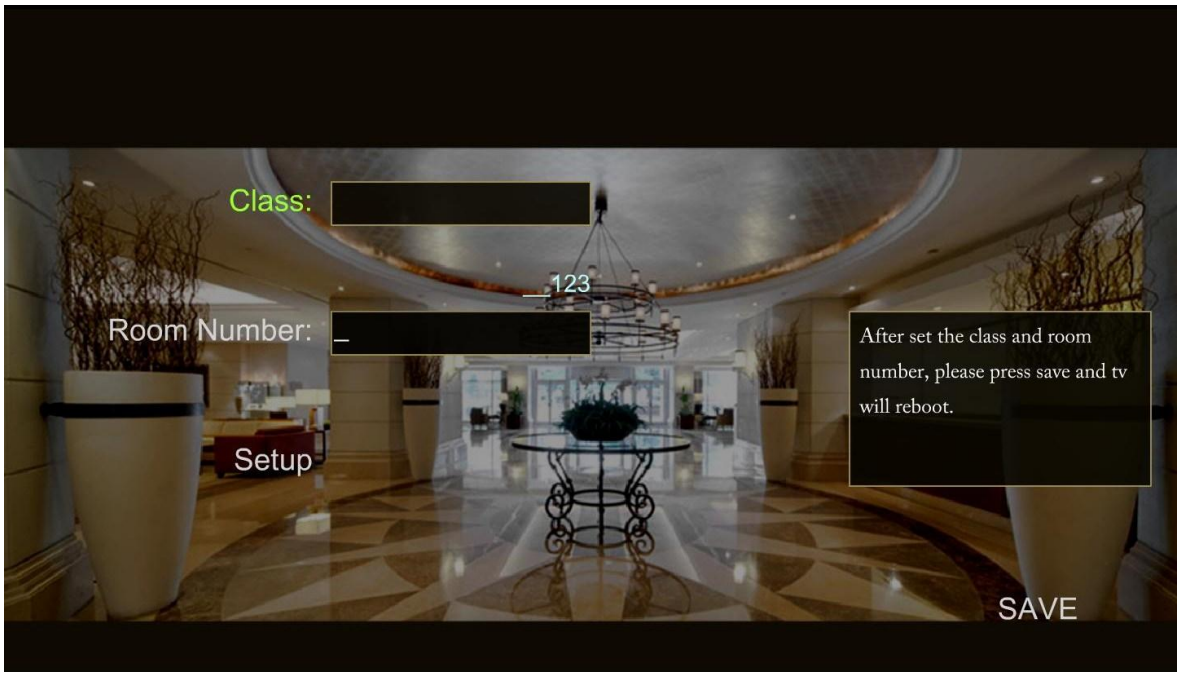

Figura 32 Serviço *setup wizard*

5.5. RÁDIO

Ao entrar neste plugin, uma lista de ícones de rádio irá aparecer ao hóspede, podendo seleccionar qual a rádio que pretende ouvir. Ao seleccionar a rádio, o menu que contém todos os ícones de rádio vai desaparecer e uma imagem do tamanho da resolução da televisão irá aparecer, imagem esta que pode ser escolhida na interface do *backend*. Aparecendo também uns OSD (*On -Screen Display*) a informar o nome da rádio.

O *layout* deste *plugin* é bastante simples, sendo bastante semelhante ao do menu principal. Não será explicado ao pormenor pois utiliza métodos de criação de *layout* já explicados no serviço de *video on demand*. Toda a informação relativa às rádios é recebida de um ficheiro XML e o seu *parsing* é realizado no arranque da televisão.

Para criar os ícones, foi necessário guardar num vector todas as imagens recebidas pelo *parsing* do XML criando um `movieclip` para cada uma e fazendo o carregamento da mesma.

A deslocação no menu apenas pode ser feito na horizontal e a sua explicação é em tudo semelhante aos casos já descritos. Se a selecção estiver no primeiro ícone, o deslocamento para a esquerda não é permitido. No caso de ser o último ícone não permite o deslocamento para a direita.

O resultado do serviço de rádio pode ser verificado na Figura 33.

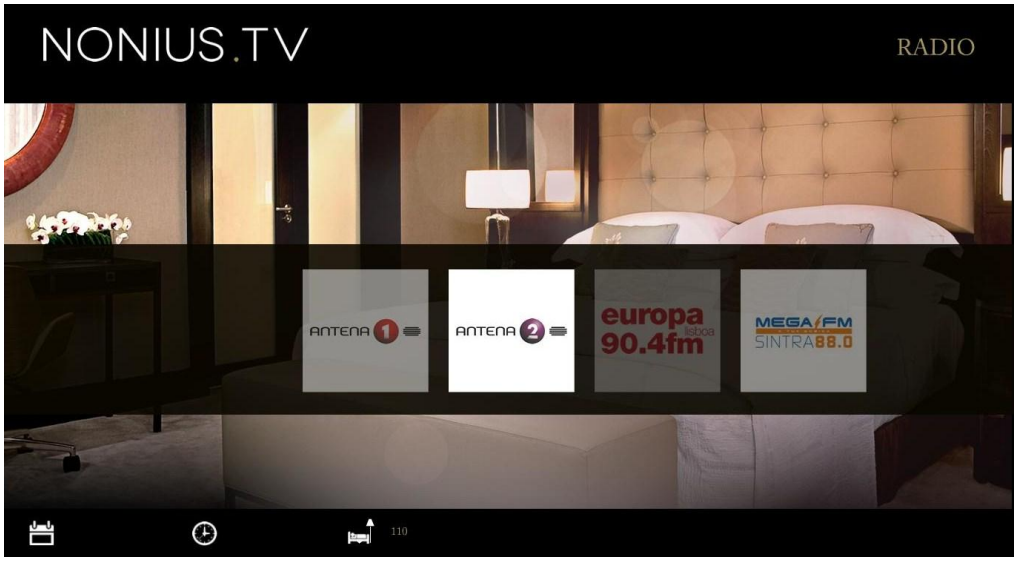

Figura 33 Demonstração do serviço de rádio

Quando o hóspede seleccionar alguma das rádios, será invocado o método `enter_radio` da classe `NiVo_RadioChannels` que chama o método `hide` que esconde tudo, ou seja coloca os `movieclip` e as caixas de texto com visibilidade zero. Chamará também o método que irá criar um `movieclip` que colocará uma imagem de fundo enquanto o rádio está em reprodução. Esta imagem pode ser alterada na interface do *backend*.

Para construir e mostrar o OSD que contém o nome da rádio que está a ser reproduzida, foram criados os seguintes métodos:

- **`createOSDContainer`**: é definido o local onde irá aparecer o OSD e é criada uma variável que guarda todas as propriedades do texto, tais como, a fonte, o tamanho e a cor. É criado também o `movieclip` que irá conter o fundo do OSD (rectângulo cinzento) e a caixa de texto onde será escrito o nome da rádio.
- **`show_osd`**: é chamado sempre que o hóspede entrar em uma rádio. Coloca a visibilidade do `movieclip` e da caixa de texto no máximo, sendo usado um método da API do FLASH que permite definir um intervalo que certo método é executado, tendo sido escolhido um intervalo de quatro segundos.
- **`hide_osd`**: quando é atingido o limite do intervalo de tempo, o `movieclip` e a caixa de texto serão destruídos.

Para a reprodução do áudio da rádio, foi criado um método designado `request_channel` que é chamado no método `enter_radio` e que permite o pedido de reprodução da *stream*.

O pedido de reprodução da *stream* irá ser tratado de forma diferente de acordo com o seu tipo. Se a *stream* for UDP (*User Datagram Protocol*) são definidas algumas propriedades do canal, como o IP e o porto, e que utilizando o método da API da televisão `request_channel_change` permite a reprodução da mesma. Caso a *stream* seja MMS (*Microsoft Media Services*), é utilizada a classe `NetStream` da API do FLASH.

É possível utilizar um objecto `NetStream` para transmitir áudio através de um objecto `NetConnection`, sendo chamado o método `play` da classe `NetStream`, e passando como argumento o IP da *stream*.

Se a *stream* for UDP, será utilizado o método `stop_current_channel` que pertence á API da televisão. No caso de ser MMS será apenas chamado o método `stop` da classe `SOUND`.

5.6. EXPRESS CHECKOUT

No serviço que permite ao hóspede consultar a sua conta, existe uma opção denominada *express checkout*.

O *express checkout* é utilizado quando o hóspede pretende efectuar o seu *checkout* mas não quer perder tempo na recepção. Não evita a ida à recepção mas acelera o processo, pois valida os consumos do hóspede.

Relativamente ao *layout*, foi adicionada uma barra no final do *plugin* que contém uma referência ao botão do comando que permite a opção de *express checkout*, assim como uma caixa de texto que apresenta a respectiva legenda. O resultado do serviço está ilustrado na Figura 34.

Figura 34 Serviço de conta com a opção de *express checkout*

Quando o hóspede pressiona o botão verde, uma janela de confirmação é apresentada. Para a construção da janela é utilizada a classe `pop_up_ExpressCheckout` que contém um método com o mesmo nome, onde recebe como argumentos os caminhos das imagens dos botões, assim como propriedades de texto e dimensionamento da janela. Para apresentar a janela, é chamado o método `show` na classe `NiVo_Bill`. A janela contém um texto que explica ao hóspede a consequência de pressionar os botões apresentados e a sua despesa total, tal como a Figura 35 demonstra.

Figura 35 Janela de confirmação do *express checkout*

Se o hóspede pressionar o botão vermelho, volta para o menu base do serviço. Caso o botão pressionado seja o verde, é feito um pedido ao serviço *web* que permite o processo de *express checkout*. Se ocorrer algum erro no processo, uma janela semelhante à da Figura 35 será apresentada a explicar a razão do problema. Caso contrário, é apresentada uma janela com uma mensagem a informar o hóspede que o processo de *express checkout* ocorreu com êxito. A conta do hóspede é limpa utilizando o método `power_off` da API da televisão que permite desligar a mesma.

Em relação ao serviço *web*, é criado um método denominado `performExpressCheckout` que recebe como argumento o número do quarto que é necessário para a construção da mensagem que será enviada para o PMS. A mensagem é construída de acordo com o que já foi referido na secção relativa ao protocolo FIAS, sendo utilizado um registo do tipo XC onde apenas é necessário adicionar os parâmetros que correspondem ao número do quarto, ao qual se pretende executar o *express checkout* e ao valor total de despesas. Para que o processo termine com sucesso, o PMS deverá responder uma mensagem igual à enviada, com a adição do parâmetro `ASOK`.

A Figura 36 apresenta a lógica do processo de *express checkout*.

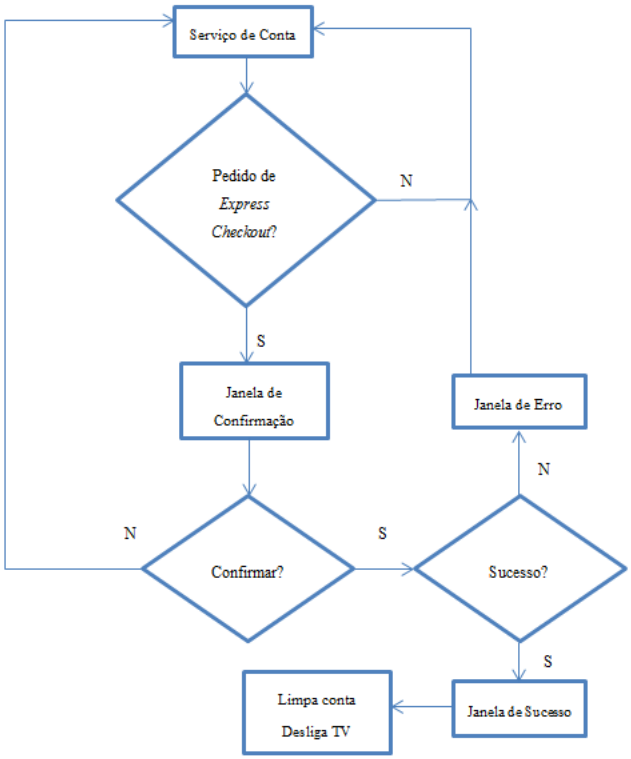

Figura 36 Fluxograma demonstrativo do processo de *express checkout*

5.7. SURVEY THRESHOLD

O *survey* exibe ao hóspede um inquérito com uma série de perguntas, tal como se pode verificar na Figura 37. As respostas a essas perguntas podem ser de vários formatos (sim/não, estrelas, múltipla escolha e resposta aberta). O serviço não foi desenvolvido no âmbito do projecto, contudo foi necessário adicionar a funcionalidade de *threshold*. Ou seja, o hotel pode definir um limiar para cada pergunta que exige uma resposta de estrelas ou sim/não. Caso a resposta à pergunta seja igual ou inferior a esse limiar, fará com que seja enviado um email ao hotel.

Todos os ficheiros criados/modificados no directório *Management* corresponde ao desenvolvimento da interface do *backend*.

No ficheiro `SurveySettings` que se encontra no directório *Management*, foi necessário adicionar ao formulário que permite adicionar novas questões no inquérito, uma opção que permita definir qual o limiar desejado para o envio do email caso este seja atingido.

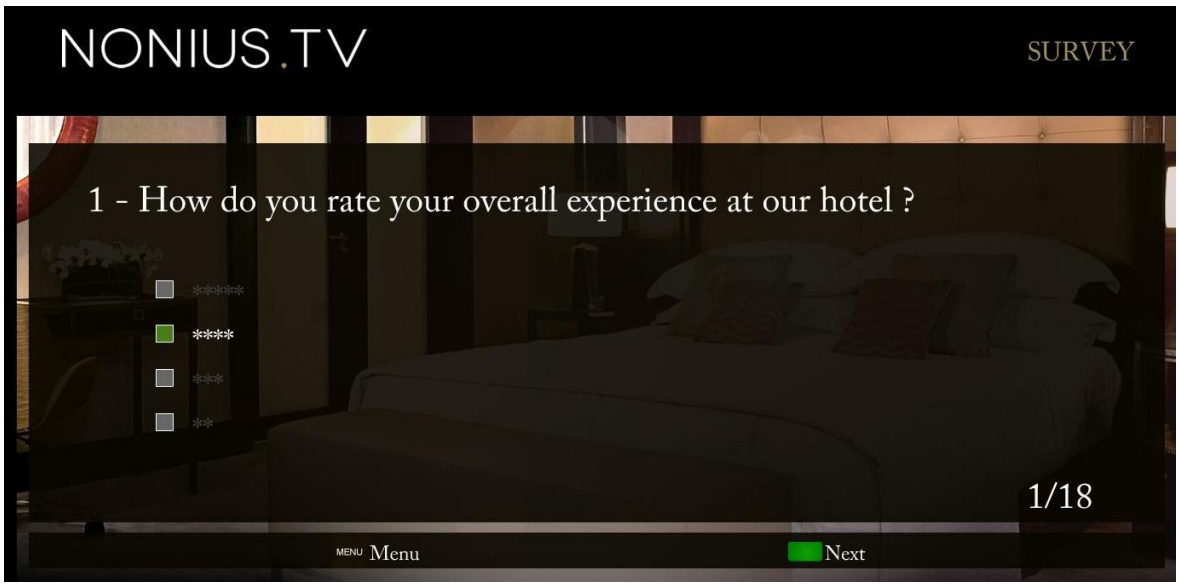

Figura 37 Inquérito

Foi colocada também uma *checkbox* que ao ser seleccionada significa que o hotel pretende utilizar esta funcionalidade. Para que o hotel possa escolher o limiar das questões é criada uma *dropbox list* apenas para as respostas de estrelas e sim/não. No caso de ser uma resposta de estrelas iria aparecer de 2 a 5 estrelas como opções para ser seleccionado o limiar. E como é óbvio, no caso das perguntas de sim/não aparecerá as opções das mesmas.

O resultado da interface do *backend*, com a adição deste novo parâmetro é demonstrado Figura 38.

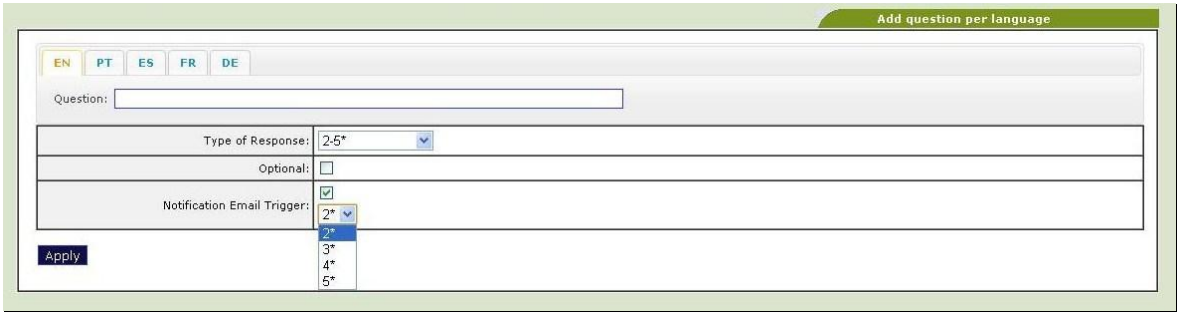

Figura 38 Formulário que permite a criação do inquérito com a opção de *threshold*

Na tabela da base de dados que armazena todas as questões submetidas, foi adicionada uma nova coluna que contém qual o valor do limiar.

Para o envio com sucesso do email é necessário fazer a sua configuração. É utilizado o protocolo SMTP (*Simple Mail Transfer Protocol*), baseado em texto simples onde um ou mais destinatários são especificados e posteriormente será enviada a mensagem. Alguns dos parâmetros necessários para a configuração do email são o IP, a porta do servidor do email, tipo de encriptação caso o servidor necessite, o nome de utilizador e a password.

Para que o hotel consiga configurar o conteúdo do email enviado, desenvolveu-se um formulário na interface do *backend* que permite configurar o remetente, o destinatário, o assunto e o corpo da mensagem, tal como a Figura 39 demonstra. Para isso, são utilizadas caixas de texto para cada um dos parâmetros e ao ser clicado o botão de aplicar, todas os parâmetros serão guardados na base de dados.

Figura 39 Configuração do email para a funcionalidade de threshold

É utilizado o serviço *web* que submete todas as respostas dadas pelo hóspede criando uma condição que caso seja a última resposta submetida pelo hóspede, será feita uma análise

das mesmas. Sendo assim, é feita uma análise apenas das questões do tipo sim/não e estrelas em que se alguma das respostas for igual ou inferior ao limiar estabelecido no *backend*, fará com que um email configurado seja enviado para o destinatário escolhido. Para a criação e envio do email é utilizada a classe `NiVo_Mail` que contém o método `getMail` que acede à base de dados e selecciona as configurações efectuadas. Tal como o Código apresenta, é utilizado o método `send` para o envio do email e recebe como argumentos todos os parâmetros que foram seleccionados com o método `getMail`.

```
$mail= new NivoMail();
$mail_content = CommentsSettings::getMail();

$mail-
>send($mail_content['destination_email_threshold'],$mail_con
tent['from_email_threshold'],$mail_content['from_name_thresh
old'], mail_content['subject_threshold'],
$mail_content['body_threshold']);
```

Código 21 Excerto de código relativo ao envio do email

5.8. MENSAGEM DE BOAS VINDAS

Quando o hóspede faz o seu *check in* ao ligar a televisão pela primeira vez no seu quarto, uma mensagem de boas vindas aparecerá contendo um *slideshow* de imagens escolhidas pelo hotel.

Para a construção da mensagem, foi necessário criar um menu na interface do *backend* que permite ao hotel escrever a mensagem que pretende que seja apresentada ao hóspede, assim como as imagens que aparecerão no *slideshow*. Foi criado um formulário no ficheiro `setWelcomeMessage` (no directório Management) que contém:

- Uma *checkbox* que informa se a funcionalidade de mensagem de boas vindas está activa ou não;
- Uma *dropbox list* que terá todos os idiomas disponíveis;
- Uma caixa de texto que contém a mensagem de boas vindas, e muda de acordo com o idioma seleccionado anteriormente;
- Duas *dropbox list*: a primeira lista todas as classes disponíveis, para que possa ser seleccionada a classe pretendida. A segunda permite a escolha do tempo entre duas imagens no *slideshow*.

O resultado deste formulário encontra-se ilustrado na Figura 40.

Figura 40 Interface do *backend* que permite a configuração da mensagem de boas vindas

Ao pressionar o botão *Apply* todas as informações do formulário são guardadas na base de dados. Serão guardadas todas as mensagens correspondentes a cada idioma, mesmo que estejam em branco.

Para que o hotel consiga realizar o *upload* das imagens para que sejam apresentadas no *slideshow*, foi necessário a criação de um segundo formulário.

Foi então utilizado um método já existente da classe `parameter` denominado `FileField` que criará o mecanismo necessário para se obter o sistema de *upload* de imagens. No método é passado como argumento o nome, o id, o nome do botão de *upload*, o tamanho máximo do ficheiro e uma pequena descrição.

As imagens que forem adicionadas serão apresentadas na interface do *backend* com recurso a uma tabela. Tal como a Figura 41 ilustra, na primeira coluna da tabela aparecerão as imagens enquanto que na segunda será apresentada uma *checkbox* que caso seleccionada, ao ser pressionado o botão *Delete Logo* a imagem é eliminada.

Figura 41 Formulário de *upload* de imagens para o *slideshow* da mensagem de boas vindas

Quando o botão *upload* for pressionado, são verificadas três condições:

- Se o número de imagens é inferior a dez;
- Se o formato da imagem é PNG;

- Se a resolução é 980x550 pixéis;

Caso estas condições se verifiquem, a imagem será armazenada num directório criado apenas para a mensagem de boas vindas denominado `welcome_message`. E actualizará um ficheiro XML criado especialmente para a mensagem de boas vindas que contém o URL de todas as imagens adicionadas. Este ficheiro XML encontra-se no directório `nivo_adapters`, que contém outros ficheiros XML que fornecem informações relativas às farmácias, meteorologia, voos e notícias.

Para passar a informação da mensagem de boas vindas para a televisão, é aproveitado o serviço web já existente, `getRoomInfo`, que envia dados como o número do quarto, o nome do hóspede, o número da reserva, o tipo de moeda, o idioma, data de chegada e saída e agora enviará também a mensagem de boas vindas, definida na interface do *backend*.

O pedido ao serviço `getRoomInfo` é criado na classe `Soap_Requests` e é invocado na classe `MainMenu` de dois em dois minutos.

Na resposta do serviço web `getRoomInfo`, é feito o *parse* do ficheiro XML que contém os endereços das imagens e armazena num vector que será enviado no método da mensagem de boas vindas, assim como o conteúdo da mensagem.

Em relação ao *layout*, foi criada a classe `Welcome_Message` dedicada a esta funcionalidade, que contém um método que irá criar uma janela do tamanho da resolução da televisão. Nesse método entrarão como argumentos as dimensões da janela, propriedades do texto (cor, tamanho, fonte, negrito), o conteúdo da mensagem de boas vindas recebido pelo serviço web `getRoomInfo` e o vector que contém as imagens que irão permitir o *slideshow*.

A janela da mensagem de boas vindas é constituída por um cabeçalho, tal como é utilizado nos *plugins*, no lado esquerdo estará a mensagem de boas vindas, enquanto que no lado direito com uma dimensão de 980x550 é apresentado o *slideshow* de imagens.

No método `show` da classe `Welcome_Message`, é feita uma análise do vector que contém os URLs das imagens. Caso o vector:

- Esteja vazio é apresentada uma imagem *Default* existente na *box*.
- Contenha apenas uma imagem, não necessita de *slideshow*.
- Contenha duas ou mais imagens, é necessário a construção do método `slideshow`.

O método *slideshow* permite a troca de imagens. É utilizado o método `SetInterval` da API do FLASH que invoca o método `slideshow` sempre que o intervalo de tempo seleccionado na interface do *backend* termina. Caso nenhum intervalo de tempo tenha sido definido, utiliza-se o valor de cinco segundos como *default*.

São utilizados *shared objects* que guardam a informação mesmo que a televisão seja desligada. O método `welcome_message` é chamado nas seguintes condições:

- Caso ocorra um *check-in*;
- Caso a mensagem de boas vindas seja apresentada e a televisão tenha sido posteriormente desligada mas o hóspede ainda não tinha feito o "*exit*" da mensagem.

Para garantir as duas situações anteriores, são criados na classe `PMS_Events` dois *shared objects*:

- **`CURRENT_CHECKIN_STATE`**: informa sobre o estado corrente do hóspede (*check in* ou *check out*).
- **`CURRENT_WELCOME_MESSAGE_FLAG`**: informa se a mensagem de boas vindas foi cancelada ou não pelo hóspede.

Para definir o valor do *shared object*, foi criado um método na classe `Utils` denominado `set_value_from_shared_object` em que são recebidos como argumentos o nome do *shared object* e o valor que se pretende atribuir.

O resultado da janela que contém a mensagem de boas vindas está ilustrado na Figura 42.

Figura 42 Mensagem de boas vindas

5.9. MENSAGEM DE ÚLTIMO DIA

A mensagem de último dia é apresentada ao hóspede no dia anterior ao seu *checkout*. O *layout* é bastante semelhante ao da mensagem de boas vindas, com a diferença que sugere ao hóspede visitar o inquérito e o serviço que permite o *express checkout*. Para isso, são adicionados dois botões (verde e azul) no rodapé da janela que ao serem pressionados funcionam como atalho para o serviço seleccionado.

O formulário para a configuração da mensagem de despedida na interface do *backend* é bastante semelhante ao da mensagem de boas vindas. Apenas foi adicionado um parâmetro que permite ao hotel definir a partir de qual hora, no penúltimo dia de estadia do hóspede, a mensagem será apresentada. Para isso, é utilizada uma JQuery que gera uma janela, tal como a Figura 43 demonstra, onde são apresentados blocos de horas e minutos para que seja mais acessível escolher a hora.

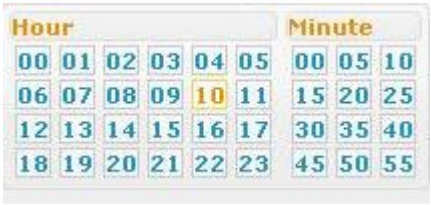

Figura 43 Janela de selecção da hora

É utilizado um sistema de *upload* de imagens diferente, sendo este um pouco mais sofisticado pois permite o arrastamento de imagens. Tem também a vantagem de ocupar um menor espaço. Apenas aceita imagens que estejam no formato PNG e com uma resolução de 980x550 pixéis.

Para a construção deste sistema de *upload*, foi criada a classe `MediaUploaderField`, na qual o método construtor recebe como argumentos o nome que se pretende atribuir ao parâmetro e o id do formulário no qual o sistema é inserido. Ao ser feito o *upload* das imagens, são criados os URLs da imagem e da *thumbnail* que permite a previsualização das imagens.

Na classe `MediaUploaderField` foram implementados os seguintes métodos:

- **`setAllowedTypes`**: Permite definir quais os formatos de imagens que são autorizados.
- **`setAllowedDimensions`**: Permite definir quais as dimensões das imagens que são autorizadas.
- **`deleteSuccessCallback`**: É utilizada quando se permite eliminar alguma das imagens inseridas.
- **`setMediaList`**: Recebe como argumento uma lista com os URLs das *thumbnails* para que possa ser feita uma previsualização das imagens inseridas.

As imagens ao serem inseridas neste sistema de *upload*, fazem com que os URLs das imagens e *thumbnails* sejam inseridos numa tabela genérica denominada `Media`. A tabela é designada genérica, pois todos os menus que utilizem este sistema irão utilizar todos a mesma tabela. É utilizado um campo escondido que contém um vector com todos os ids das imagens adicionadas, sendo actualizado sempre que alguma imagem é adicionada ou retirada. Ao pressionar o botão *Apply*, é utilizado esse vector para poder adicionar na tabela denominada `last_day_message_media` que contém os ids das imagens destinadas à mensagem de despedida.

O resultado do formulário que permite a configuração da mensagem de despedida está ilustrado na Figura 44.

Figura 44 Formulário de configuração da mensagem de despedida

É importante ter em conta que se no *check in* não for enviada a data de saída do hóspede esta mensagem não irá funcionar, pois é dependente desse factor.

Tal como na mensagem de boas vindas, é utilizado o serviço web `getRoomInfo` para enviar as configurações realizadas no *backend* (mensagem, hora e a data de saída do hóspede).

Para que no lado da box fosse possível obter os URLs das imagens adicionadas, foi necessário a criação do serviço web `getLastDayMessageImages` que faz uma selecção na tabela `Media` utilizando os ids das imagens da tabela `last_day_message_media`.

Relativamente ao *layout*, é criada a classe `LastDayMessage` que contém os mesmos métodos que a classe da mensagem de boas vindas, com a diferença que são criadas duas imagens que representam os botões que ao serem pressionados irão servir de atalho para o serviço de conta ou o inquérito.

Tal como já foi referido, o serviço `getRoomInfo` é invocado de dois em dois minutos, sendo assim, na sua resposta são criadas uma série de condições:

- Os parâmetros correspondentes à mensagem não podem ser vazios.
- Se a data corrente for igual ao penúltimo dia da estadia do hóspede e a data corrente superior ou igual à data definida no *backend*.
- O *shared object* `LAST_DAY_MESSAGE_FLAG` tem de ter o valor de *true*.

Caso estas condições se verifiquem será chamado o serviço web que retorna os URLs das imagens escolhidas no *backend*. Sendo assim na resposta do serviço `getLastDayMessageMedia` é feito o parse do ficheiro XML e são chamados os métodos da classe `Last_Day_Message` que permitem a construção e apresentação da mensagem de despedida.

Foi necessário definir um *shared object* na classe `Utils` denominado de `LAST_DAY_MESSAGE_FLAG` inicialmente definido a *false*. Caso ocorra um *check in* ou *check out*, o valor do *shared object* é colocado a false. O valor apenas é colocado a *true* nas seguintes situações:

- Caso a janela que contém a mensagem de boas vindas tenha sido cancelada pelo hóspede;
- Caso não exista mensagem de boas vindas.

5.10. EVENTOS DO HOTEL

Este serviço permite ao hotel criar eventos, onde define a sua data de início e de fim, e associar ao mesmo os quartos que pretender. O hóspede, ao tentar aceder ao serviço na televisão do seu quarto, apenas será apresentado o evento que esteja mais perto da data actual. Caso não esteja associado a nenhum evento, uma janela aparecerá a informá-lo.

Para esta funcionalidade, foi necessário trabalhar na interface do *backend*, criar os serviços web, e o *layout*.

Relativamente à interface do *backend* foi necessário construir um menu denominado *Events* que contém três submenus:

- *Create Event*: Este submenu permite ao hotel criar o evento, onde define a sua data de início e fim e associa os quartos que pretender a esse evento. Permite também adicionar imagens para *slideshow* ou uma *stream* de vídeo.

- *View Events*: Será apresentada uma tabela que contém todos os eventos existentes.

- *Delete Events*: Permite ao hotel eliminar os eventos que pretender.

O processo de criação do evento está dividido em três passos. O primeiro passo apresenta um formulário que contém:

- Três caixas de texto que permite ao hotel escrever o nome, a data de início e fim do evento.
- É utilizada uma JQuery que utiliza o método `getRooms` para obter todos os quartos e inserir cada um deles dentro de uma caixa. Podendo o hotel assim seleccionar vários quartos de uma só vez.
- É utilizada uma JQuery que cria uma caixa de texto e que na sua parte superior contém cinco abas correspondentes aos idiomas suportados. A aba ao ser clicada apresentará a caixa de texto correspondente ao idioma seleccionado. Podendo assim definir qual a mensagem de apresentação do evento.

O resultado do formulário está ilustrado na Figura 45.

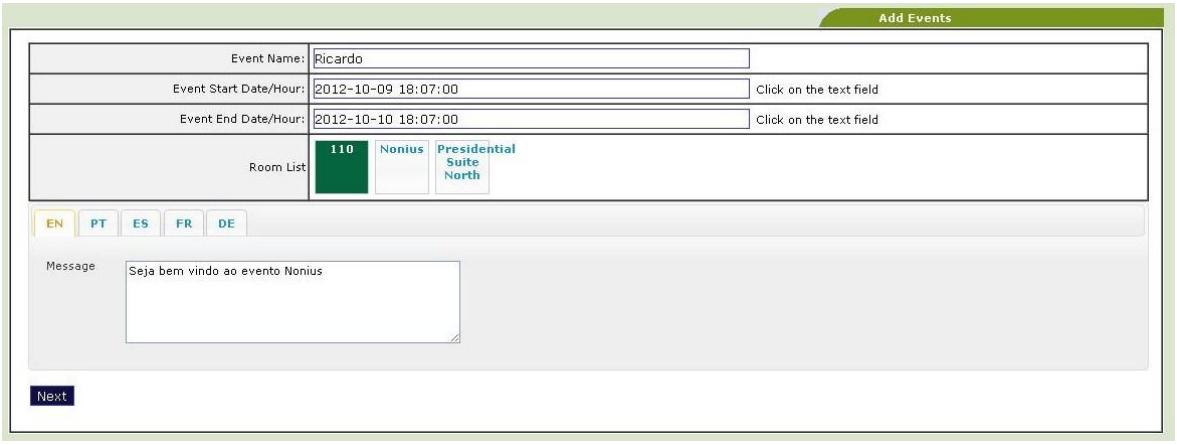

Figura 45 Formulário que permite a criação do evento

O botão *Next* ao ser pressionado, redirecciona a página para o segundo passo de criação do evento. É apresentada uma *dropbox list* que permite ao hotel escolher qual o tipo de multimédia que pretende utilizar. Se o tipo de multimédia escolhido for:

- Imagens: É feito o redireccionamento da página onde será apresentado um sistema de *upload* de imagens igual ao utilizado na mensagem de despedida.
- Vídeo: É feito o redireccionamento da página onde será apresentada uma caixa de texto onde poderá ser inserida uma *stream* UDP, tal como a Figura 46 demonstra.

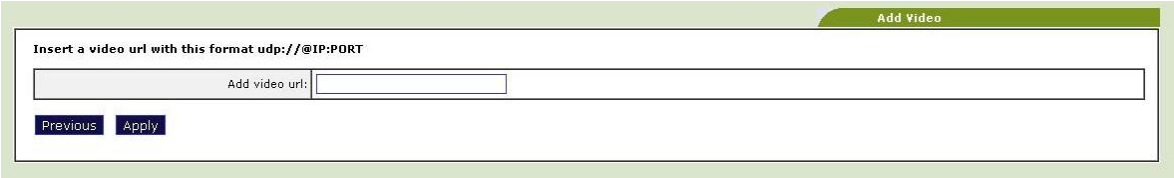

Figura 46 Adição de uma *stream* de vídeo para o evento

O botão *Apply* ao ser pressionado irá redireccionar para uma página que apresenta uma tabela com todos os eventos criados, mostrando o nome do evento, as datas de início e fim e os quartos associados ao mesmo.

Para hotéis com um elevado número de quartos e com interesse em criar vários eventos, seria complexo realizar uma pesquisa de um evento para o editar ou eliminar. Sendo assim, decidiu-se distinguir os eventos por cores, onde a cor verde corresponde aos eventos correntes, cinza claro está associado aos eventos passados e o preto para os eventos futuros. Foi também adicionada uma JQuery associada à tabela que adiciona um filtro de pesquisa. O resultado desta tabela pode ser visto na Figura 47.

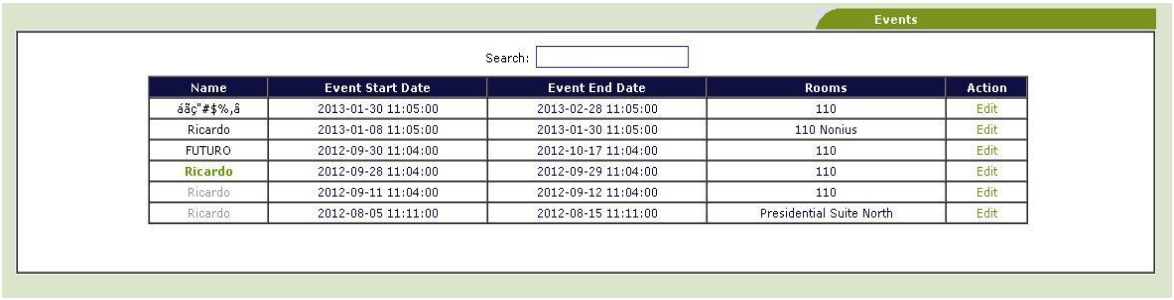

Figura 47 Tabela demonstrativa dos eventos criados

Se na tabela for clicado no *edit*, então irá ser redireccionado novamente para a página de criação de evento, mas no URL terá agora o valor do id do evento. Caso algo seja alterado, é feito um *update* na base de dados.

O submenu denominado *Delete Event* apresenta uma tabela semelhante à da Figura 47, com a diferença que na última coluna, a acção existente em vez de ser de edição é de eliminação.

Para que esta funcionalidade seja possível, são necessários três serviços web:

- **`getEventsInformation`**: retorna o conteúdo do evento como a mensagem, as datas, os quartos associados e o tipo de multimédia escolhido.

- **`getEventsImages`**: retorna os URLs de todas as imagens adicionadas na interface do *backend*.
- **`getEventsVideo`**: retorna o URL da *stream* de vídeo.

Relativamente ao *layout*, este segue exactamente a mesma linha das mensagens de boas vindas e de despedida. Para isso foram criadas duas classes:

- **`NiVo_Events`**: são chamados os serviços web necessários para a criação do serviço de eventos. São chamados os métodos da classe `MyEvents` que permitem a construção e apresentação do *plugin*.
- **`MyEvents`**: Contém os métodos necessários para a construção e apresentação da janela de eventos.

Quando o ícone do menu principal é escolhido pelo hóspede, é chamado o serviço *web* `getEventsInformation`, que retorna o conteúdo do evento. Na resposta do mesmo, se este retornar vazio, ou seja, não existir nenhum evento associado, é criada uma janela que informará o hóspede que não contém nenhum evento associado, tal como a Figura 48 ilustra. Caso contrário é invocado o serviço web correspondente ao tipo de multimédia escolhido.

Figura 48 Janela apresentada na inexistência de eventos associados

Caso o tipo de multimédia escolhido seja imagens, o processo é exactamente igual ao já referenciado no desenvolvimento das mensagens de boas vindas e despedida.

No que diz respeito ao vídeo, é construído o método `show_video` (invocado no método `show`) onde é identificado o IP e a porta da *stream* UDP recebida e é invocado o método `request_channel` que já foi referenciado no desenvolvimento do serviço de rádio. É

utilizado também o método `set_video_size` da API da televisão que permite definir a posição e o dimensionamento do vídeo.

O resultado do *layout* do serviço de eventos está apresentado na Figura 49.

Figura 49 Serviço de eventos

5.11. DETALHES

No desenvolvimento do projecto foram criados serviços web para suportar o serviço de alarme. Foram necessários três serviços web:

- **`getWakeUpCall`**: Retorna a hora que está definida no alarme.
- **`setWakeUpCall`**: Submete a hora pretendida para o alarme.
- **`deleteWakeUpCall`**: Elimina a hora submetida.

Estes serviços deixaram de ser utilizados, pois ficou decidido que o processo todo do alarme ficaria do lado da televisão.

Na interface do *backend* foi construído um parâmetro que permite ao hotel definir qual o volume máximo da televisão. A televisão recebe esse valor e utilizando o método `on_volume_change` da API da televisão, que é invocado sempre que o hóspede altera o volume da televisão e faz a verificação. A lógica deste processo está representada na Figura 50.

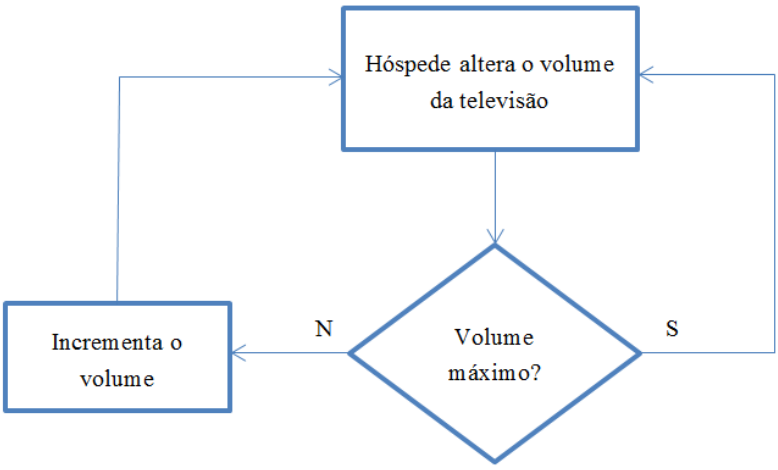

Figura 50 Demonstração da lógica associada à definição do volume máximo da televisão

6. CONCLUSÕES

Ao longo deste documento apresentou-se o desenvolvimento de serviços do produto Nonius TV para a televisão interactiva LG Pro: Centric, referindo o trabalho relativo ao *backend*, assim como de *frontend*.

Este serviço de televisão interactiva já se encontra implementado e funcional em 500 quartos do maior hotel existente em Portugal. Desta forma, pode-se afirmar que os objectivos deste projecto foram cumpridos com sucesso.

Uma das grandes vantagens do serviço de televisão interactiva LG Pro: Centric consiste no facto da *box* estar embebida dentro da própria televisão. Isto é vantajoso, principalmente no ramo hoteleiro, pois o equipamento ocupa menos espaço e garante-se uma maior segurança do material. Algumas dificuldades foram encontradas ao longo do desenvolvimento do projecto, devido ao facto da memória interna da televisão ser limitada a 40 MB. Isto faz com que se o pacote que contém o ficheiro swf e o tema se aproximar deste valor, o rendimento diminui tornando a aplicação mais lenta e podendo mesmo provocar a não inicialização da televisão. Isto obrigou a um cuidado detalhado no software desenvolvido.

Um dos maiores desafios no desenvolvimento da aplicação foi o serviço de v*ideo on demand*, pois a sua interface gráfica é bastante complexa e contém várias funcionalidades.

No que diz respeito a aspectos pessoais, este estágio permitiu-me desenvolver competências e adquirir experiência em vários níveis. Foram desenvolvidos conhecimentos a nível científico, empresarial e pessoal.

De forma a tornar o serviço ainda mais atractivo para o cliente final, novas funcionalidades podem ser incluídas. A título de exemplo pode referir-se suporte de mais idiomas, o volume do alarme aumentar conforme o tempo, alertas de expiração da licença e promoções.

Além disso, dentro de alguns meses, será lançado um novo modelo da LG Pro: Centric que permitirá o uso de HTML5. Como trabalho futuro prevê-se converter para HTML5 algumas funcionalidades do que existe em FLASH. Isto fará com que esta televisão não fique em desvantagem em relação às outras soluções de televisão interactiva oferecidas pela NONIUS pois permitirá acesso a serviços de Internet.

.

Referências Documentais

[1] HTML5 (29 de Março de 2012). Obtido em 13 de Outubro de 2012, de W3C: http://www.w3.org/TR/html5/

[2] Dive into HTML5 (s.d). Obtido em 13 de Outubro de 2012, de Mark Pilgrim: http://diveintohtml5.info/

[3] HTML5 Features – Multimedia (s.d). Obtido em 13 de Outubro de 2012, de HTML5Rocks: http://www.html5rocks.com/en/features/multimedia

[4] HTML5 - Internet Explorer 9 Guide for Developers (2012). Obtido em 13 de Outubro de 2012, de Microsoft: http://msdn.microsoft.com/en-us/ie/hh410106.aspx

[5] Media formats supported by the HTML5 audio and video elements (24 de Setembro de 2012). Obtido em 13 de Outubro de 2012, de Mozzila Developer: https://developer.mozilla.org/en-US/docs/Media_formats_supported_by_the_audio_and_video_elements

[6] ActionScript Classes (s.d). Obtido em 13 de Outubro de 2012, de Adobe Help: http://help.adobe.com/en_US/AS2LCR/Flash_10.0/help.html?content=00001394.html

[7] What are local shared objects? (2012). Obtido em 13 de Outubro de 2012, de Adobe: http://www.adobe.com/security/flashplayer/articles/lso/

[8] Web Storage (2012). Obtido em 13 de Outubro de 2012, de W3C: http://dev.w3.org/html5/webstorage/

[9] An update on Flash Player and Android (28 de Junho de 2012). Obtido em 13 de Outubro de 2012, de Adobe: http://blogs.adobe.com/flashplayer/2012/06/flash-player-and-android-update.html

[10] Maria Benedita Campos Neves Malheiro, Das aplicações aos serviços web, 2010

[11] Maria Benedita Campos Neves Malheiro, Serviços web, 2004

[12] Web Services Architecture, (2004), Obtido em 14 de Outubro de 2012, de W3C: http://www.w3.org/TR/ws-arch/#stakeholder_using

[13] Simple Object Access Protocol, (2000), Obtido em 14 de Outubro de 2012, de W3C: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383494

[14] Web Services Description Language, (2001), Obtido em 14 de Outubro de 2012, de W3C: http://www.w3.org/TR/wsdl

[15] Property Management Systems, (2012), Obtido em 14 de Outubro de 2012, de Agylysys: http://www.agilysys.com/home/Hospitality/Solutions/pms.htm

[16] Micros, Property Solutions, (2012), Obtido em 14 de Outubro de 2012, de Micros: http://www.micros-fidelio.eu/en/Solutions/Hotels-and-Resorts/Individual-hotels-and-hotel-cooperation/Property-Solutions.aspx

[17] Fidelio, Welcome to Fidelio Interface Application Specification,10 de Novembro de 2009

[18] Nonius Software, NiVo3PFO HLD, 2010

[19] Nonius Software, Sistema Nonius TV, Setembro de 2011

[20] Learning ActionScript 2.0 in FLASH, (2007), Obtido em 15 de Outubro de 2012, de Adobe: http://livedocs.adobe.com/flash/9.0/main/flash_as2_learning.pdf

[21] Introduction to ActionScript 3.0, (s.d), Obtido em 15 de Outubro de 2012, de Adobe: http://help.adobe.com/en_US/ActionScript/3.0_ProgrammingAS3/WS5b3ccc516d4fbf351e63e3d118a9b90204-8000.html

[22] Flash Develop, The open source FLASH IDE, (2012), Obtido em 15 de Outubro de 2012, de Flash Develop: http://www.flashdevelop.org/

Anexo A. Estrutura de base de dados

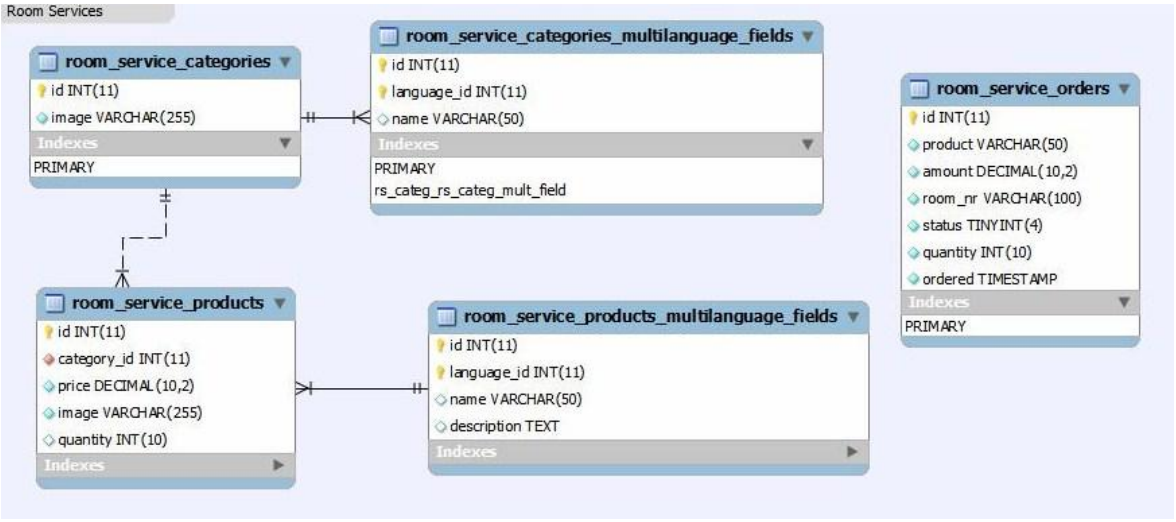

Figura 51 Estrutura de base de dados para o serviço de quartos

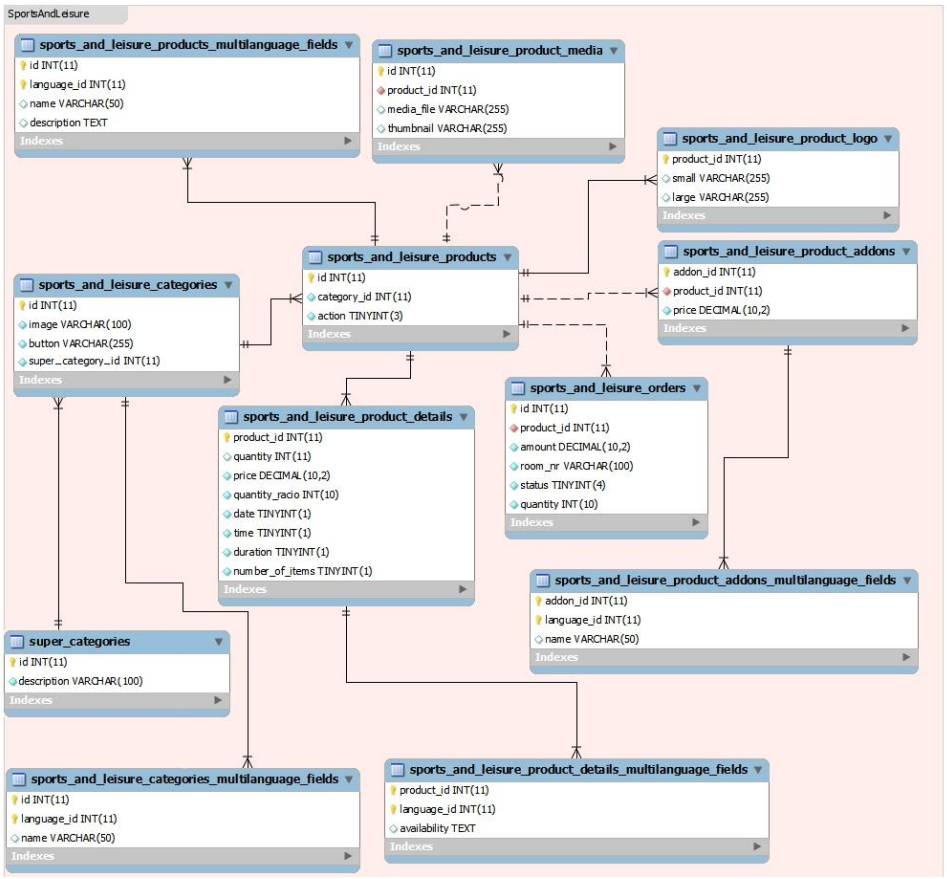

Figura 52 Estrutura de base de dados para o serviço de desportos e lazer

Anexo B. Acções do comando para criarem interactividade

A interactividade da aplicação é realizada através da utilização do comando. Para tal, associa-se um *listener* a cada serviço, que verifica qual o botão pressionado e realiza as acções definidas para esse botão. É utilizada a classe IHcapConst que define constantes aos botões do comando. Este processo é demonstrado no excerto de código seguinte:

```
//LISTENER FOR VOD
vodListener.onKeyDown = function()
{
   switch (vod_code)
   {
   case IHcapConst.KEY_RIGHT: NiVo.NiVo_vod.right();
   break;
   case IHcapConst.KEY_LEFT:  NiVo.NiVo_vod.left();
   break;
   case IHcapConst.KEY_ENTER:
   NiVo.NiVo_vod.process_enter_key(NiVo);
   break;
   case IHcapConst.KEY_CHANNEL_UP:  NiVo.channel_up();
   break;
   case IHcapConst.KEY_CHANNEL_DOWN: NiVo.channel_down();
   break;
   case IHcapConst.KEY_PAUSE: NiVo.NiVo_vod.pause();
   break;
   case IHcapConst.KEY_PLAY: NiVo.NiVo_vod.play();
   break;
   case IHcapConst.KEY_FAST_FORWARD: NiVo.NiVo_vod.seek_fw();
   break;
   case IHcapConst.KEY_REWIND: NiVo.NiVo_vod.seek_rw();
   break;
   case IHcapConst.KEY_UP: NiVo.NiVo_vod.up();
   break;
   case IHcapConst.KEY_DOWN: NiVo.NiVo_vod.down();
   break;
   case IHcapConst.KEY_STOP:
   case IHcapConst.KEY_BACK:
   case IHcapConst.KEY_EXIT:
   case IHcapConst.KEY_ESCAPE:
        if ( NiVo.NiVo_vod.vod_enabled == true)
        {
                NiVo.NiVo_vod.process_exit_key(vod_code);
                break;
        }
        NiVo.exit_vod();
   break;
   case IHcapConst.KEY_MENU:
        if (NiVo.NiVo_vod.vod_enabled)
        {
                NiVo.NiVo_vod.exit_from_movie();
                NiVo.NiVo_vod.show();
                NiVo.NiVo_vod.vod_enabled = false;
                NiVo.exit_vod();
        }
   NiVo.exit_vod();
   break;
```

```
case IHcapConst.KEY_RED:
       if ( NiVo.NiVo_vod.trailer.isEnabled())
       {
              NiVo.NiVo_vod.show_pop_up_after_trailer();
       }
break;
case IHcapConst.KEY_AD: NiVo.NiVo_vod.change_audio();
break;
```

Código 22 Processo de atribuição de métodos aos botões do comando

Anexo C. Criação de tipos complexos

Quando o resultado de um serviço *web* não é do tipo primário (int, char, etc.) é necessário construir um tipo complexo.

```
$server->wsdl->addComplexType(
   'RoomCategory',
   'complexType',
   'struct',
   'all',
   '',
   array(
          'id' => array('name'=>'id','type'=>'xsd:string'),
          'image' =>
array('name'=>'image','type'=>'xsd:string'),
          'name' =>
array('name'=>'name','type'=>'xsd:string'),
          'code' =>
array('name'=>'code','type'=>'xsd:string'),
          )
);

$server->wsdl->addComplexType(
   'RoomCategories',
   'complexType',
   'array',
   '',
   'SOAP-ENC:Array',
   array(),
   array(
          array(
                 'ref' => 'SOAP-ENC:arrayType',
                 'wsdl:arrayType' => 'tns:RoomCategory[]'
          )
   )
);
```

Código 23 Processo de construção de tipos complexos