NONIUS.TV NA SMART TV LG PRO:CENTRIC

Tiago Manuel Marques da Silva Moreira

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: Tiago Manuel Marques da Silva Moreira, Nº 1070317, 1070317@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
20 de Novembro de 2012

Dedico este trabalho aos meus pais, porque sem eles nada disto teria sido possível.

Agradecimentos

Venho desde já agradecer à Nonius Software por me ter proporcionado este estágio, oferecendo-me a capacidade de aprender ainda mais, permitindo o desenvolvimento da minha capacidade técnica em diversos campos, mas também do meu relacionamento interpessoal. Agradeço ao meu supervisor na empresa, Eng.º Raul Carvalho, pela disponibilidade e ajuda em vários aspectos particulares do meu trabalho.

Quero agradecer muito à equipa deste projecto: ao Nuno Mota que enveredou comigo desde o início da Pro:Centric, ao Luís Pimenta que me deu um apoio tremendo nas horas de maior dificuldade e ao meu grande amigo Ricardo Pereira por todos os bons momentos proporcionados. Um muito obrigado também ao resto da equipa do departamento de Investigação e Desenvolvimento que me ajudaram a integrar da melhor forma na empresa.

Agradeço profundamente à minha orientadora, a Prof. Dr.ª Paula Viana, por todo o apoio prestado, pela simpatia e por toda a paciência que teve no decorrer deste estágio. Quero também agradecer-lhe todos os conselhos e sugestões dados e todo o tempo que disponibilizou para a realização desta tese, mesmo esse sendo muito limitado.

Quero agradecer também a todos os docentes do Mestrado em Engenharia Electrotécnica e de Computadores – Telecomunicações, pelos conhecimentos transmitidos durante os dois anos de Mestrado. Um especial obrigado ao Instituto Superior de Engenharia do Porto e aos meus respectivos colegas e amigos por todos os bons momentos que me proporcionaram ao longo destes últimos cinco anos, fazendo-me crescer tanto a nível académico como pessoal.

Resumo

A Smart TV é um equipamento novo e em evolução que incorpora um computador e acesso à Internet em ecrãs de grande qualidade. Permite a implementação de serviços interactivos, acesso à Internet e televisão. À medida que a tecnologia melhora, muitos equipamentos estão a tornar-se tão capazes quanto os computadores normais quando se trata de navegação na *web* e até mesmo vídeo na Internet (*Video-on-Demand* e *streaming* de vídeo).

O projecto de estágio "NONIUS.TV na Smart TV LG Pro:Centric" foi desenvolvido na empresa Nonius Software que está inserida no ramo das telecomunicações. Uma das suas áreas de actividade está relacionada com o desenvolvimento de plataformas de entretenimento para o mercado hoteleiro, combinando diversos serviços e funcionalidades a pensar no hóspede.

Este projecto teve como finalidade implementar alguns dos serviços e funcionalidades já existentes em plataformas que usam uma *Set-Top Box* da Nonius Software, numa Smart TV, aproveitando também para inovar e criar novos serviços. Nesse conjunto está incluída a implementação de uma Caixa de Mensagens, Serviço de Quartos, Serviço de Desporto e Lazer, Serviços Informativos, um cliente RTSP, um despertador, um sistema de mudança de idioma e outras pequenas funcionalidades desenvolvidas ao longo de toda a aplicação.

Esta dissertação apresenta um estudo sobre as tecnologias Smart TV existentes no mercado, assim como as vantagens e desvantagens da sua utilização para este projecto. Após uma análise de requisitos de forma a estruturar e desenhar os serviços e funcionalidades a serem criados para a aplicação, implementou-se um conjunto de serviços, usando a linguagem de programação ActionScript 2.0, que permitiram à empresa disponibilizar um novo produto baseado na televisão Pro:Centric da LG.

Palavras-Chave

Smart, TV, internet, VoD, *streaming*, NONIUS.TV, ActionScript

Abstract

The Smart TV is a new and evolving equipment that incorporates a computer and Internet access within high-quality screens. It allows the implementation of interactive services, Internet access and television. As technology improves, many devices are becoming as capable as normal computers when it comes to web browsing and even Internet video (Video-on-Demand and video streaming).

The internship project "NONIUS.TV na Smart TV LG Pro: Centric" was developed in Nonius Software company that is embedded in the telecommunications field. One of its areas of activity is the development of entertainment platforms for the hospitality market, imposing several services and features thinking about the guest.

This project had as its main purpose the implementation of some of the existing features and services on platforms that use a Set-Top Box from Nonius Software, in a Smart TV, also taking advantage to innovate and create new services. In this group we can include the implementation of an Inbox, Room Service, Sport and Leisure Service, Information Services, an RTSP client, an alarm clock, a system of language change and other small features developed throughout the application.

This dissertation presents a study on the Smart TV technologies available on the market, and discusses the advantages and disadvantages of their use for this project. After a requirements analysis to structure and design the services and features to be created for the application, a set of services were implemented, using the ActionScript 2.0 programming language, which allowed the company to offer a new product based on the Pro:Centric TV from LG.

Keywords

Smart, TV, Internet, VoD, *streaming*, NONIUS.TV, ActionScript

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

3DTV	–	3D Television
AIT	–	Application Information Table
API	–	Application Programming Interface
APP	–	Application Software
AS1	–	ActionScript 1.0
AS2	–	ActionScript 2.0
AS3	–	ActionScript 3.0
CCTV	–	Closed-Circuit Television
CDTV	–	Conventional Definition Television
DC/OC	–	Data Carousel / Object Carousel
DTV	–	Digital Television
DVB-C	–	Digital Video Broadcasting - Cable
DVB-GEM	–	Digital Video Broadcasting - Globally Executable MHP
DVB-MHP	–	Digital Video Broadcasting – Multimedia Home Platform
DVB-S	–	Digital Video Broadcasting - Satellite
DVB-T	–	Digital Video Broadcasting - Terrestrial
DVD	–	Digital Video Disk
GEM	–	Globally Executable MHP

GUI – Graphical User Interface

HCAP – Hotel Common Application Platform

HD – High Definition

HTTP – Hypertext Transfer Protocol

I&D – Investigação e Desenvolvimento

IPTV – Internet Protocol Television

LCD – Liquid Crystal Display

LED – Light Emitting Diode

MAC – Media Access Control

MHP – Multimedia Home Platform

MPEG – Moving Picture Experts Group

MSC – Message Request Chart

OCAP – Open Cable Application Plataform

OOP – Object Oriented Programing

OSD – On-Screen Display

PC – Personal Computer

PHP – PHP: Hypertext Preprocessor

PMS – Property Management System

RAM – Random Access Memory

RSS – Rich Site Summary

RTP – Real-time Transport Protocol

RTSP – Real Time Streaming Protocol

SOAP – Simple Object Access Protocol

SSH – Secure Shell

STB – Set-Top Box

SWF – Shock Wave Flash

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

USB – Universal Serial Bus

VOD – Video on Demand

VPN – Virtual Private Network

XAIT – eXtended Application Information Table

XML – eXtensible Markup Language

1. INTRODUÇÃO

Neste capítulo, é feita uma exposição geral do projecto de estágio realizado no âmbito da unidade curricular Tese/Dissertação do 2º ano do Mestrado em Engenharia Electrotécnica e de Computadores, área de especialização em Telecomunicações.

Inicialmente é apresentada a empresa onde decorreu o estágio, bem como os objectivos definidos para o mesmo. Por fim, apresenta-se a organização do relatório.

1.1. LOCAL DO ESTÁGIO

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

A Nonius Software é uma empresa nacional de telecomunicações, com capital 100% português, que se dedica ao desenvolvimento de soluções para a gestão de sistemas informáticos, de redes de comunicação e de entretenimento. A Nonius foi fundada em Abril de 2005, começando por criar soluções de acesso à Internet para o mercado hoteleiro português. Graças à grande expansão da empresa no mercado, esta continuou a criar novos produtos e serviços direccionados para a hotelaria. Em 2009, a Nonius acrescentou ao seu portfólio de produtos, uma solução avançada de *in-room entertainment*: a TV Interactiva. Esta recorre à tecnologia IPTV (*Internet Protocol Television*), Sinalética Digital, VOD (*Video-on-Demand*), Telefonia IP, aplicações para dispositivos móveis, Internet e Televisões Inteligentes, e tem como nome NONIUS.TV. A Nonius tem a sua tecnologia

em mais de 35.000 quartos de hotel espalhados por Portugal, Espanha, resto da Europa, África e Brasil.

1.2. OBJECTIVOS

Este projecto de estágio surgiu com o intuito de adaptar todos os serviços existentes nas outras plataformas da NONIUS.TV, que dependem de uma *Set-Top Box* (STB) dedicada ligada à TV, a um cenário que traz a STB para dentro da própria televisão, contornando as limitações que tal sistema impõe (processamento, armazenamento e controlo).

A Nonius pretende seguir a tendência tecnológica do mercado televisivo tendo como fim a adaptação dos seus serviços para as Smart TVs.

O objectivo deste projecto consiste em criar uma nova plataforma que integre todos os serviços existentes desde o VOD, IPTV, Rádio, Serviços Interactivos (ex.: Serviço de Quartos, Desporto e Lazer), Serviços de Informação (ex.: notícias, meteorologia, farmácias), jogos e mensagens, na Smart TV LG Pro:Centric. Visto ser uma nova plataforma, aproveitar-se-á também para inovar criando novos serviços e tecnologias.

As principais contribuições neste trabalho foram:

- Melhoramento dos serviços já existentes, quer a nível de funcionalidades quer a nível de interface gráfica (GUI);
- Optimização da interface gráfica da plataforma tornando-a mais apelativa e mais *user friendly* em relação às que já existiam;
- Criação de novos serviços de forma a enriquecer o portefólio de aplicações;
- Criação de um cliente RTSP (*Real Time Streaming Protocol*) para *streaming* de vídeo;

Neste contexto, é de realçar a importância do projecto para a Nonius Software, pois será a primeira plataforma da empresa que não irá necessitar de uma STB. Factor inovador importante que implica novos desafios, tendo em vista grandes aplicações futuras.

1.3. ORGANIZAÇÃO DO RELATÓRIO

Este relatório encontra-se dividido por seis capítulos.

No primeiro capítulo, é feita uma breve introdução, onde se apresenta a empresa onde foi feito o estágio, onde é contextualizado o trabalho efectuado e onde são apresentados os objectivos e as contribuições mais importantes para este projecto.

No segundo capítulo, é feita uma análise sobre o mercado das Smart TVs em termos de tecnologias e soluções comerciais existentes, assim como as arquitecturas e perspectivas de desenvolvimento para estes sistemas.

O terceiro capítulo faz uma apresentação dos serviços existentes na NONIUS.TV, assim como da gama de produtos de televisão interactiva que a empresa tem disponível no mercado. Descreve também a arquitectura e os componentes necessários para o funcionamento de todo sistema. No fim deste capítulo é descrita a personalização do sistema da NONIUS.TV.

No quarto capítulo é apresentado o modelo de televisão LG Pro:Centric, a nível da sua API (*Application Programming Interface*) e a linguagem de programação que esta usa, fazendo também referência à plataforma de desenvolvimento utilizada.

O quinto capítulo apresenta toda a implementação efectuada para este projecto e descreve a funcionalidade dos respectivos serviços com base no que foi desenvolvido.

Por fim, o sexto capítulo descreve todas as conclusões tiradas ao longo do projecto e perspectivas de trabalho futuro para optimização do mesmo.

2. SMART TV

Neste capítulo, é feito um levantamento sobre o mercado das Smart TVs em termos de tecnologias e soluções comerciais existentes, assim como as arquitecturas e aspectos futuros para este sistema.

Começa-se por definir o conceito de Smart TV e alguns outros termos relacionados para contextualizar a área de trabalho. Depois são apresentadas as arquitecturas e soluções existentes no mercado. Por fim, são referidos alguns pontos importantes para o futuro deste sistema.

2.1. SMART TV E A CASA DIGITAL

Existem três termos que definem o conceito de Smart TV: convergente, interligada e inteligente. O termo Smart TV descreve a integração da Internet em televisões e em S*et-Top Boxes,* assim como a fusão e convergência da tecnologia entre TVs, *Smartphones*, *Tablets*, computadores e outros dispositivos de comunicação que habitualmente são usados por várias pessoas, permitindo ao utilizador um acesso a múltiplos canais de conteúdos.

A Internet é agora uma parte dominante do dia-a-dia de muitas pessoas, sendo considerado um bem essencial. Os telemóveis já evoluíram de dispositivos utilizados apenas para fazer chamadas telefónicas, para *Smartphones* – telemóveis com avançadas funcionalidades

computacionais que permitem ter acesso à Internet em qualquer lugar. A mesma tecnologia foi implementada nas televisões, daí o termo Smart TV. O foco principal das Smart TVs de hoje é a utilização interactiva dos meios de comunicação *online*, como Internet TV, *streaming* de conteúdos, redes sociais e navegação na *web*, adicionando novas características à experiência tradicional de se ver televisão. Da mesma forma que a navegação na Internet, os *widgets* da *web* e o *software* (como jogos ou aplicações) são integrados nos *Smartphones* actuais, a mesma tendência de conectividade tornou-se parte das TVs de hoje, criando uma convergência entre computadores, dispositivos móveis e a TV digital. As Smart TVs permitem aos telespectadores procurar e encontrar filmes, vídeos e fotos na Internet ou armazenados num disco rígido em casa, através da própria TV, usando um controlador remoto.

Actualmente, a Smart TV oferece a capacidade de aceder à Internet através da própria TV, mas prevê-se que venha a incluir muitas outras funcionalidades. Existe actualmente investigação e até alguns resultados muito precoces de novas tecnologias e aplicações para este tipo de mercado, e alguns deles irão ser referenciados mais à frente neste relatório [1].

Apesar de actualmente já se ouvir falar das Smart TVs, a realidade é que os modelos actuais existentes no mercado ainda não mostram a solução de TV totalmente convergente e social que irá realmente mudar a televisão, transformando uma indústria que tem visto uma mudança constante nos últimos anos – desde a introdução das tecnologias plasma e LCD, televisão sobre IP (IPTV), serviços de vídeo a pedido (*Video on Demand*), a televisão 3D (3DTV), integração com sistemas de vídeo vigilância e assistência médica.

Embora existam muitos modelos de dispositivos que se afirmam como "interligados" e "inteligentes", ainda não existe a convergência realmente necessária aos utilizadores. A solução de uma televisão totalmente convergente passa por combinar todos os dispositivos de comunicação numa única plataforma. A televisão será um dispositivo completamente multimédia, combinando todos os dispositivos usados pelas pessoas diariamente fornecendo um ponto central de acesso que pode ser usado consoante as necessidades de cada utilizador. Para que isso seja possível, a Smart TV deve implementar um conjunto de normas, ter ligação à Internet e tem de permitir uma interacção entre diversos utilizadores, seja através das redes sociais ou outros meios, assim como já acontece nos actuais S*martphones* e *Laptops*. Para uma melhor facilidade de utilização, esta terá de ser totalmente interactiva usando controlo por gestos e por voz. Sendo assim, a tecnologia

Smart TV actual serve para preparar o futuro dessas soluções integradas. A Figura 1 apresenta de forma genérica a integração da Smart TV num cenário totalmente digital [2].

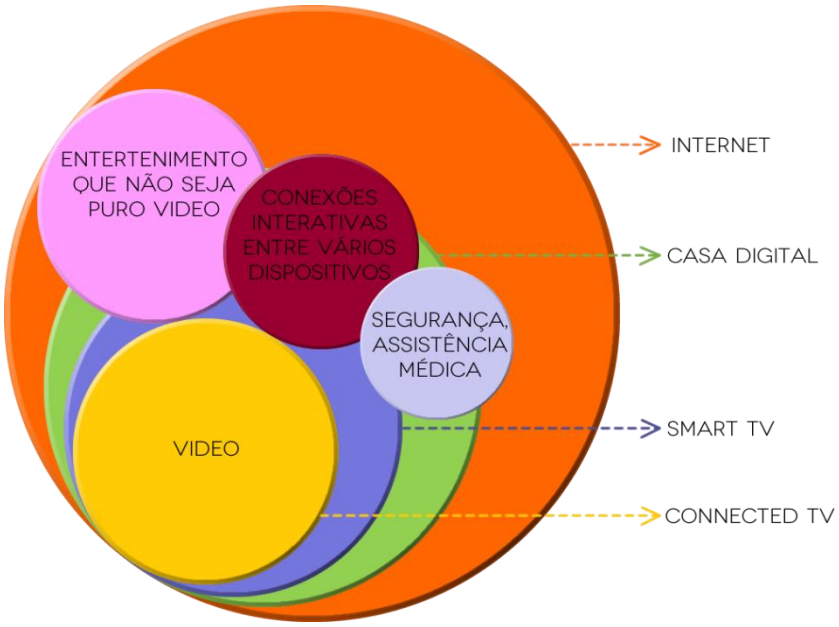


Figura 1 Integração da Smart TV num cenário totalmente digital

2.2. A REVOLUÇÃO DO SERVIÇO DE TELEVISÃO

Diversos modelos de acesso a conteúdos têm vindo a ser criados de forma a permitir a introdução de novas funcionalidades no tradicional serviço de televisão. Exemplos desse paradigma são as tecnologias *Interactive and Social TV* e *Converged TV*.

2.2.1. *INTERACTIVE AND SOCIAL TV*

Assistir a um programa de televisão, tem sido, até agora, quase inteiramente uma experiência passiva. Com o desenvolvimento das Smart TVs, são muitas as possibilidades da verdadeira interactividade. A televisão interactiva (*Interactive TV*) é o termo que descreve qualquer televisão que dê uma informação de retorno perante alguma acção do espectador. Isto é, a informação pode fluir através do ecrã de televisão a partir da fonte de radiodifusão para o telespectador (conteúdo de TV normal), mas o contrário também pode acontecer, ou seja, a TV também pode receber informação, dependendo de uma acção ou instrução que é feita pelo telespectador. Com este recurso é possível entregar conteúdo especificamente para o telespectador, consoante as escolhas que faz através da interacção.

Reality TV, as redes sociais e o desporto são exemplos de áreas onde a TV interactiva pode dar algo mais à experiência de assistir a um programa, permitindo, por exemplo, votar ou

fazer apostas desportivas em tempo real. Isto permite ao telespectador reagir activamente ao que está a ver no momento [3].

É difícil falar sobre a televisão interactiva sem estar constantemente a pensar na parte social da mesma. Televisão Social (*Social TV*) é um termo genérico que descreve a capacidade de um telespectador conseguir aumentar a sua interacção social combinando a actividade social *online* com o conteúdo que está a ser visto no ecrã, ou seja, oferece a capacidade de conversar com outras pessoas enquanto cada um assiste ao mesmo programa, trocando opiniões sobre o conteúdo que está a ser visto.

Alguns exemplos de televisão social são a integração de serviços de *chat*, correio electrónico, mensagens de texto ou mesmo videoconferência. Isto pode ser feito directamente através de uma Smart TV, ou usando outros dispositivos de convergência como os *Tablets* ou os *Smartphones*, seja de forma independente ou em conjunto com o ecrã de televisão. Redes sociais como o *GetGlue* são bons exemplos de partilha de informação *online* enquanto se vê um filme ou uma série televisiva [4].

2.2.2. *CONVERGED TV*

Em termos genéricos, "Convergência" refere-se a uma união de duas ou mais entidades distintas ou fenómenos, sendo um assunto emergente no mundo das tecnologias. Neste contexto, o termo refere-se à combinação de duas ou mais tecnologias diferentes num único dispositivo.

O termo *Converged TV* descreve a forma como os dispositivos móveis estão a unir-se, ou a convergir, com a televisão criando uma experiência combinada. Com isto cria-se um segundo ecrã de visualização.

Para um contínua ascensão da televisão social, é necessário a convergência estar disponível e integrada em todos os dispositivos, de forma a serem usados para acesso a conteúdos sociais de hoje em dia [5].

2.3. ARQUITECTURAS DE SMART TV

São várias as formas de tornar uma televisão inteligente, incluindo ou não *hardware* adicional. A Figura 2 apresenta duas opções base para a arquitectura deste tipo de dispositivos:

- **Usando uma *Set-Top Box* externa:** Existem várias tecnologias com equipamentos externos que tornam o sistema numa Smart TV.
  - *Smart BlueRay Players*: são a evolução dos antigos leitores de DVD, oferecendo imagens de qualidade mais elevada e recursos extra que incluem ligação à Internet;
  - *Internet TV Boxes*: é um dispositivo de convergência que combina algumas ou todas as capacidades de um computador pessoal com uma aplicação de *software* que suporta vídeo, fotografias, reprodução de música e ligação à Internet, permitindo assim o acesso a conteúdos em *streaming*;
  - Consolas de Jogos: consolas como a Playstation 3 ou a Xbox 360, para além de serem consolas normais de jogos permitem também o acesso à Internet para jogar em modo de multijogador, receber *emails* e ver conteúdos em *streaming*.
- **Usando um sistema integrado:** Esta implementação é em muito semelhante com o sistema integrado com uma STB, mas neste caso a STB está integrada na própria televisão. A ligação à Internet é feita directamente na própria televisão, tendo implementada uma placa de rede, seja esta com ou sem fios.

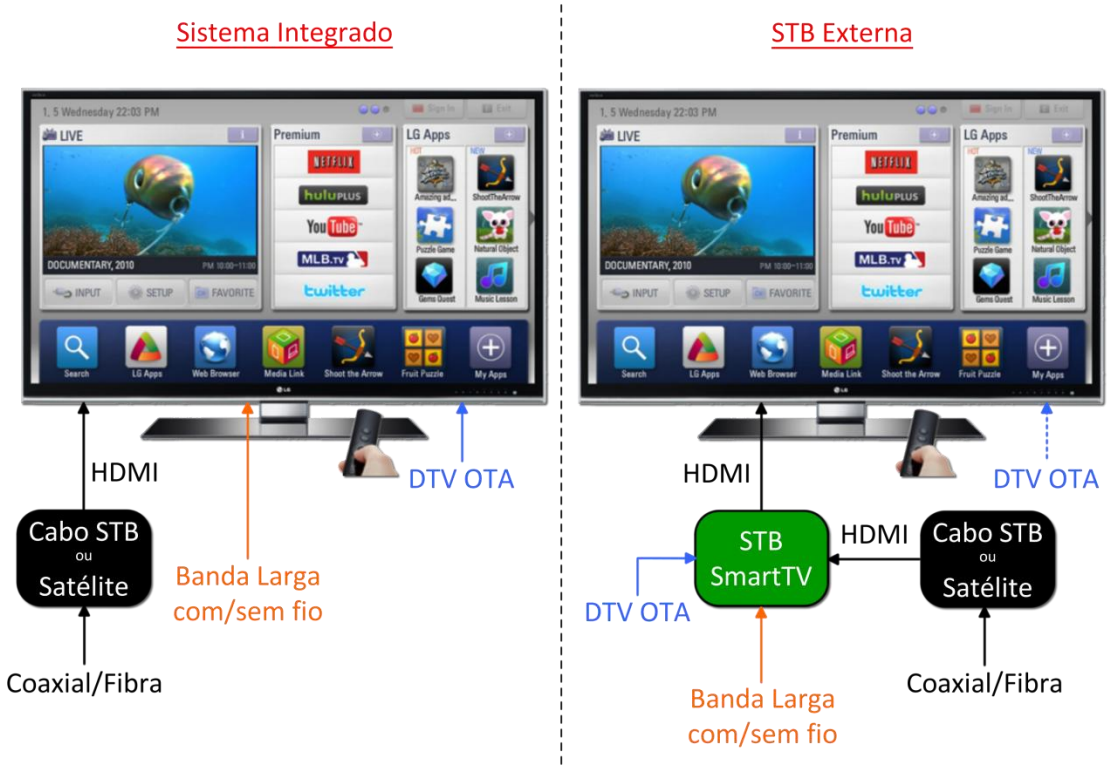


Figura 2 Diagrama das duas arquitecturas existentes de Smart TV

2.4. SOLUÇÕES COMERCIAIS

Os principais fabricantes de Smart TVs têm vindo a apostar bastante nesta nova forma de televisão e têm reconhecido os potenciais lucros de vendas que desta advêm. As marcas líderes do mercado actual são a Samsung e a LG, mas outras estão a ser cada vez mais reconhecidas pela sua tecnologia, tal como a Sony, a Toshiba e a Panasonic.

Para além dos fabricantes das televisões, outros intervenientes neste processo de comercialização de dispositivos inteligentes são a Google e a Apple que disponibilizam as duas plataformas em pareceria com os fabricantes de TVs.

2.4.1. PLATAFORMAS

2.4.1.1. GOOGLE TV

A Google TV é uma plataforma de software que integra o sistema operativo Android da Google juntamente com a versão em Linux do *browser* Google Chrome de forma a criar uma sobreposição de televisão interactiva com a própria televisão e com os sites WebTV existentes. Juntamente com isso é adicionada uma interface gráfica (GUI – *Graphical User Interface*) desenhada para ser apresentada num ecrã de televisão que permite interacção com um comando normal. Assim como o sistema operativo Android da Google, pode ser executado em inúmeros *Smartphones*, a Google TV pode ser executada em várias televisões ligadas à Internet, incluindo STBs, sistemas de satélite e televisões digitais. A plataforma Google TV tem como objectivo permitir a criação de novas experiências que combinam televisão com funcionalidades como o *download* de aplicações, a navegação na Internet, a utilização de um *Smartphone* em vez do tradicional comando, a criação de *playlists*, o acesso a serviços de VOD, entre outras [6].

A Google TV actualmente não é, na verdade, uma gama de modelos de TV. É sim uma plataforma que permite que uma série de funcionalidades sejam adicionadas à televisão. Fabricantes como a Samsung, a LG e a Sony estão actualmente a integrar esta plataforma nos seus próprios produtos (Figura 3). Estas empresas que apoiam o desenvolvimento da Google TV, também criam as suas próprias plataformas para cobrir as suas apostas, segmentar as suas linhas de produtos e manter a oportunidade de inovar.

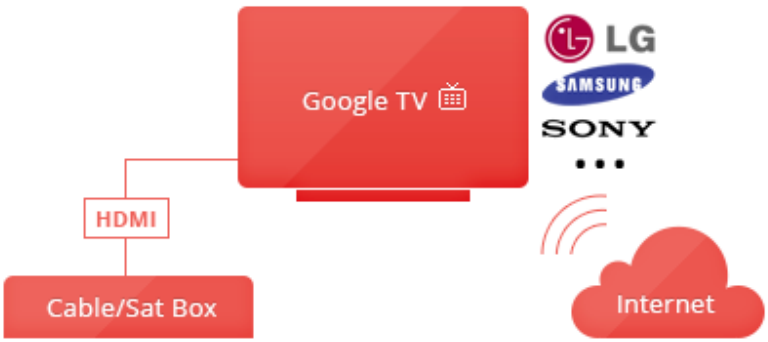


Figura 3 Arquitectura da plataforma Google TV usando um sistema integrado [7]

A adopção da plataforma Android por diversos fabricantes permite à Google ampliar a utilização da mesma, criando o potencial para se tornar uma plataforma *standard* das Smart TVs.

É de esperar que o sistema Android da Google TV vá emergir não só em STBs (Figura 4) mas também em gravadores de vídeo digitais e em televisões ligadas à Internet, oferecendo aos fornecedores de serviços um ecossistema que pode ser integrado em *Smartphones* e *Tablets* com o sistema operativo Android [8].

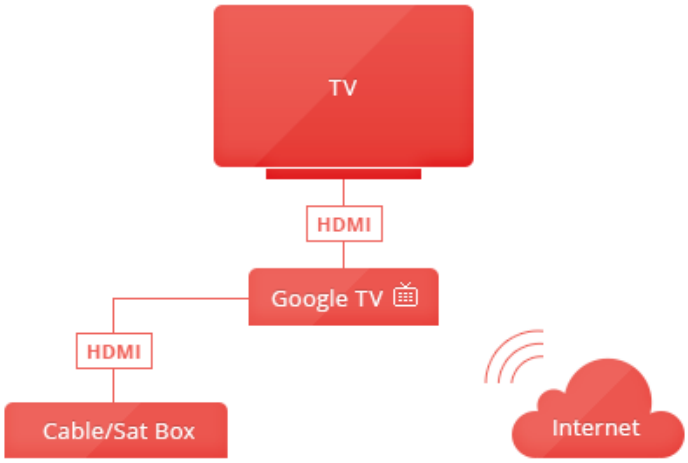


Figura 4 Arquitectura da plataforma Google TV usando uma STB [6]

2.4.1.2. APPLE TV

Apple TV é uma plataforma que permite criar uma pequena rede de dispositivos multimédia, tendo a possibilidade de receber conteúdos tanto a partir dos computadores domésticos, como *online*. Esta é constituída por um pequeno dispositivo que usa um cabo para ligar à televisão e usa rede sem fios para se ligar à rede doméstica e à Internet. A sua

principal finalidade é permitir o acesso a conteúdos multimédia a partir da loja iTunes ou de outros fornecedores como Netflix e Pandora, a partir de um televisor ou de um sistema de áudio.

Os primeiros modelos de TV da Apple utilizavam um disco rígido para sincronizar com as bibliotecas do iTunes dos computadores domésticos. Era possível comprar filmes pela loja do iTunes e grava-los no disco rígido do Apple TV. No entanto, a segunda geração da Apple TV não tem um disco rígido de modo que só pode fazer *streaming* dos conteúdos multimédia.

A Apple TV só pode transmitir música ou filmes armazenados numa biblioteca do iTunes que esteja presente em algum dos computadores domésticos. Se o utilizador possuir vários computadores na sua rede doméstica ou um disco de armazenamento anexado à rede do dispositivo, e quer ter acesso aos conteúdos nele guardados, esta não é a melhor tecnologia a ser usada, acabando por ser um ponto negativo neste sistema.

Actualmente, a Apple TV permite que conteúdos a partir do iTunes, Flickr, MobileMe, YouTube e Netflix, apenas nos EUA. A Apple tem uma série de parcerias com estúdios de cinema e redes de televisão, fornecendo-lhes uma biblioteca de conteúdos impressionante. Uma das próximas apostas da Apple é na alta definição. Espera-se uma versão da Apple TV com suporte para ecrãs de alta definição com 1080p, ao invés dos actuais 720p.

O objectivo base do ecrã de televisão é e continuará a ser o de ver televisão, a questão está em como integrar isso com uma experiência de utilização intuitiva [9].

2.4.2. FABRICANTES DE EQUIPAMENTOS

2.4.2.1. LG

A marca LG aposta também na tecnologia da Google e ainda este ano irá lançar alguns modelos de Smart TV com essa implementação. Através da Google TV, a LG juntou o sistema operativo Android com as tecnologias 3D e Smart TV. Juntamente com a Google TV, a LG continuará a avançar com a sua própria plataforma de TV inteligente, chamada NetCast, usando tecnologias *web* livres incluindo o Linux e o Webkit.

A Google tem apostado bastante neste mercado e por essa razão a Google TV é considerada uma das melhores plataformas de Smart TV, mas no caso da LG Electronics, esta também conseguiu criar de origem a sua própria interface de utilizador, como apresenta a Figura 5. A diferença entre a plataforma Smart TV da LG com a da Google é que a Google TV tem o papel de pesquisar e tentar encontrar aquilo que o espectador possa querer ver, através da análise de conteúdos que este vai adquirindo.

Figura 5 Interface NetCast da LG

A LG possui uma grande parte do mercado global de 3DTV, mas também tem dado grande importância à possibilidade de ligação à Internet nos seus diferentes produtos. Cada vez mais, a LG está focada na forma como os seus vários produtos inteligentes podem trabalhar juntos, permitindo criar a casa inteligente, como por exemplo, a capacidade de um *Smartphone* fazer s*treaming* de conteúdos para a televisão e ao mesmo tempo usa-lo como ecrã secundário. Isto é possível com a plataforma Smart Share da LG, que possibilita o acesso ao conteúdo de dispositivos digitais como câmaras, *Smartphones*, *Tablets* e PCs, sem fio e transferi-los para a televisão [10].

2.4.2.2. SAMSUNG

A Samsung começou por fazer entrega de *feeds* RSS (*Rich Site Summary*) para as suas televisões ligadas à Internet em 2008 e lançou uma loja de aplicações em 2010 que conta agora com mais de 13,000 aplicações, e já somou um total de 100 milhões de *downloads* nos principais países europeus [11].

Apesar de suportar a plataforma Google TV, a Samsung continua a construir a sua própria plataforma, chamada Bada, e também um conjunto de aplicações que estão acessíveis através da interface Smart Hub, apresentada na Figura 6.

Figura 6 Interface Smart Hub da Samsung

De forma a evitar a desactualização dos seus modelos de Smart TVs, a Samsung criou um *hardware* com o nome de Samsung Evolution Kit, que actualiza os dispositivos base instalados, sem a necessidade de comprar uma televisão completamente nova.

A Samsung oferece nos seus mais recentes modelos de Smart TV, um conjunto de funcionalidades que incluem o reconhecimento de voz, controlo gestual e reconhecimento facial, tendo uma câmara integrada na parte de cima do ecrã mas também é possível o acesso com os típicos ratos e teclados sem fio. Esta tecnologia permite o controlo do dispositivo através de comandos verbais como "Diminuir Volume", "Canal 24" em várias línguas ou de interacção gestual em menus rotativos, controlo de volume ou selecção nas aplicações. O reconhecimento do rosto permite que cada membro da família possa ter um perfil próprio, com as aplicações que mais usa, com os seus dados, ficheiros e marcadores para acesso à Internet, não sendo preciso fazer *login* manualmente quando usam a TV.

Apesar de não ser obrigatório o uso do comando em certas televisões, a Samsung está a desenvolver *Tablets* para serem usados como comandos e como dispositivos de visualização em segunda vista para a plataforma das Smart TVs. Estes podem ser usados para uma pesquisa de catálogos de VOD mais prática, e quando escolhido o filme basta fazer um movimento de "arrastar para a TV" para ver o mesmo na televisão.

O que é possível já nos computadores pessoais, a Samsung permite também agora nas TVs: a possibilidade de abrir uma aplicação, abrir outra sem fechar a anterior, e ser possível alternar entre as duas. Também é possível apresentar múltiplos ecrãs numa única TV, permitindo ver vários conteúdos diferentes ao mesmo tempo.

O serviço AllShare, permite a criação de uma rede inteligente que integra diversos dispositivos (TVs, *Tablets*, *Smartphones*, e aparelhos domésticos) sendo possível mover o conteúdo de dispositivo para dispositivo, e do dispositivo para a Internet. Este serviço permite ao utilizador procurar o conteúdo manualmente através da Internet ou ter acesso ao conteúdo directamente na Smart TV ou outros dispositivos móveis. Vídeos que são armazenados no computador podem ser transferidos através da rede local para um S*martphone* e, em seguida, a partir deste para a televisão. Com o histórico de família, tendo como base o conteúdo partilhado/alojado na Internet, os utilizadores podem ter acesso ao conteúdo nos *Smartphones*, *Tablets*, PCs e nas Smart TVs.

Toda esta conectividade vai para além da rede local. É possível estar noutra rede e ter acesso através do *Smartphone* ao conteúdo alojado no computador de casa. Outra interacção interessante entre televisões e outros dispositivos é o conceito de espelhamento do ecrã. É possível, com um *Smartphone* Galaxy da Samsung, aceder a uma página *web* e, em seguida, torna-la visível na Smart TV. O telefone passa então a ser utilizado como controlador, baseando-se no movimento do mesmo para mover o cursor no ecrã. Isto é útil quando se quer um ecrã maior para navegar na Internet [12].

2.5. FACTORES COM IMPACTO PARA O SUCESSO DAS SMART TVS

O sucesso das novas formas de ver televisão dependerá de um conjunto de factores nos quais se inclui a sua facilidade de utilização. Se o acesso envolve o uso de um comando para clicar nos endereços num ecrã, não irá ser necessariamente fácil para um utilizador de televisão. Muitas televisões actuais com ligação à Internet têm interfaces pobres que não tornam a sua utilização prática. O objectivo é ser tão fácil como ligar uma televisão normal. Muitos fabricantes já se aperceberam disso e já estão a abordar o problema em dispositivos mais recentes com uma interface de utilizador muito mais *user friendly*.

Outro aspecto importante é a privacidade dos utilizadores, especialmente a partir de uma perspectiva de televisão social. Quando a televisão é partilhada quase sempre por mais que uma pessoa, não é confortável aparecerem actualizações das redes sociais para que todos possam ver ou interromper a sua visualização.

É possível apontar alguns aspectos que são necessários para o sucesso da Smart TV [13]:

- **O uso abundante de dados**: O acesso livre aos dados em páginas *web* e a facilidade do acesso a esses dados para os utilizadores, mudou a forma dos conteúdos *online*. Isto

trouxe novas possibilidades de mistura de dados de diferentes fontes criando novas formas de educação, informação e entretenimento. Os proprietários dos dados acabaram por verificar que o valor de permitir a livre reutilização dos mesmos em contextos novos foi maior do que estar a protegê-los num local inacessível. Para os *developers*, as APIs das televisões podem possibilitar novos serviços em torno da *web* e em torno dos dados e conteúdos de TV. Os serviços expostos por meio de interfaces podem encapsular funcionalidades reutilizáveis, levando ao fácil desenvolvimento de aplicações de TV mais inovadoras;

- **Acesso ilimitado a conteúdos**: Essas aplicações desenvolvidas para serviços televisivos podem conduzir à emergência de uma nova experiência de televisão integrada. A televisão será muito mais do que conteúdo único, num único fluxo de dados, num único equipamento. A experiência de televisão pode envolver o acesso consistente a conteúdos perfeitamente interligados, obtidos a partir de diversas fontes e reproduzidos através de equipamentos diferentes. Podem ser *mushups* de conteúdos em ecrãs partilhados ou conteúdo complementar em diferentes ecrãs. O conteúdo pode ser não linear, com interacção intuitiva para controlar a sua reprodução, podendo usar a voz e os gestos, por exemplo. A Smart TV não deve ser restringida pelas aplicações nesta disponíveis. Deve ser capaz de obter qualquer conteúdo que esteja disponível na *web*.

Os novos prestadores de serviços digitais de conteúdos, como os famosos Netflix, Hulu e Amazon, têm visto um grande progresso nos EUA nos últimos anos, desde que se tornaram disponíveis nestas televisões. Na Europa existem serviços semelhantes no mercado, mas mais localizados: no Reino Unido, Alemanha, Escandinávia, Dinamarca, Noruega e Suécia é possível usar o serviço LoveFilm, e na França existe o CineSnap [14].

2.6. VISÃO DA TELEVISÃO DO FUTURO

Nos próximos anos a televisão tende a sofrer uma transformação acelerada, com o serviço a tornar-se cada vez mais pessoal e democrático, o que irá revolucionar e ultrapassar o actual modelo de transmissão de conteúdos. A televisão será qualquer experiência audiovisual entregue a partir de qualquer fonte até qualquer dispositivo, onde o dispositivo estará ligado à Internet e as fontes de conteúdo serão múltiplas, mas perfeitamente integradas numa única experiência do consumidor. De seguida são listadas algumas perspectivas futuras do que pode acontecer no mercado televisivo [15]:

- Com o uso de equipamentos móveis a aumentar, a televisão como é vista actualmente poderá tornar-se num meio secundário, como a actual rádio, dependendo cada vez mais da retransmissão de eventos ao vivo para atrair o público em geral, enquanto que os outros modos de distribuição digital baseiam-se a transmitir a programação pré-gravada quando o consumidor quiser e aonde quiser;
- As aplicações globais das redes sociais tendem a proliferar no domínio da televisão, proporcionando interacção comum, avaliações e recomendações em tempo real, criando experiências compartilhadas em torno da visualização assíncrona de conteúdos, para além das fronteiras geográficas;
- Os *Tablets* e os *touch screens* tendem a multiplicar-se e, cada vez mais, áudio e vídeo serão consumidos em equipamentos pessoais móveis em vez do tradicional ecrã partilhado, tornando a experiência de ver televisão mais funcional;
- A maioria dos serviços de televisão e vídeo serão entregues através de redes de dados em vez de utilizar as actuais normas de radiodifusão digital;
- A transmissão para dispositivos de banda larga tende a tornar-se dominante e a maioria dos ecrãs acabará por ter alguma forma de conexão de dados;
- Com as variadas resoluções existentes em diferentes equipamentos, os ecrãs acabarão por se tornar independentes da resolução. Com a tendência da criação de potentes processadores de multimédia, estes irão fornecer descodificação em tempo real entre diferentes formatos e resoluções muito rapidamente, desacoplando os ecrãs das normas de transmissão específicas;
- A alta definição vai acabar por ser norma em todos os equipamentos, duplicando o número de imagens por segundo, oferecendo um movimento mais suave;
- As redes de banda larga tendem a tornar-se mais abrangentes. A distribuição em multidifusão irá permitir que a programação ao vivo seja entregue com uma boa relação qualidade-custo a milhares de utilizadores em simultâneo sobre redes de dados fixas ou sem fio num meio global;
- As redes de fibra-óptica acabarão por chegar directamente a todas as casas. Os operadores de televisão por cabo vão migrar para protocolos de Internet e vão estender as suas redes de fibra-óptica até às instalações dos clientes, oferecendo acesso a uma gama quase ilimitada de meios audiovisuais, entregues mais rapidamente ou em tempo real, sem atrasos ou interrupções;

- As redes domésticas tendem a estar omnipresentes. As tecnologias das redes de dados com e sem fios irão substituir a cablagem dedicada dentro de casa para a distribuição de audiovisual, comunicação e automação da própria casa, enquanto que as ligações universais de baixa tensão irão reduzir a necessidade de vários adaptadores.
- A distribuição de conteúdo audiovisual em suportes físicos tende a diminuir. Os *streams* e os *downloads* irão ser dominantes e os conteúdos audiovisuais licenciados estarão sempre acessíveis na rede de armazenamento da nuvem;
- Os lançamentos globais de conteúdos irão reduzir a pirataria. Os grandes filmes e programas serão distribuídos simultaneamente em todo o mundo para reduzir a pirataria e os eventos globais realizados regionalmente serão financiados através de patrocínios e subscrições;
- A protecção de direitos autorais acabará por ser invisível. A gestão das restrições de direitos digitais será transparente para os utilizadores legais que poderão ter acesso aos conteúdos livremente em qualquer dispositivo dentro dos termos da sua licença. Hoje em dia isso já acontece com algum do conteúdo HD. Para este ser distribuído precisa de sistemas de segurança baseados em *watermarking*, de forma a combater a distribuição desautorizada;
- Com o aumento da falta de tempo das pessoas, estas vão pagar para evitar a publicidade. Apesar da sofisticação da segmentação das mensagens comerciais, tornando-as cada vez mais curtas e concisas, as pessoas vão poder pagar por serviços de subscrição que permitem ao cliente não ser interrompido por anúncios intrusivos.

Em conclusão, a forma como os dados, os serviços e os conteúdos irão coexistir e interagir, irá ser fundamental para os novos modelos de negócio, porque os conteúdos serão ubiquamente utilizados e perfeitamente interligados a novas experiências [16].

3. NONIUS.TV

A tecnologia Smart TV tem vindo a despertar bastante interesse no mercado das telecomunicações e *software*, e como consequência as empresas têm vindo a apostar nesta nova forma de televisão, reconhecendo a sua potencialidade. Foi com base neste contexto que a Nonius Software construiu a sua própria solução avançada de TV Interactiva: a NONIUS.TV.

Este capítulo faz uma breve apresentação das várias plataformas de televisão interactiva construídas pela Nonius, quer a nível dos serviços disponíveis, quer nas suas funcionalidades. Por fim é feita uma breve apresentação dos componentes gráficos mais utilizados nas interfaces das várias plataformas.

3.1. INTRODUÇÃO

A NONIUS.TV é um sistema avançado de entretenimento e multimédia, tendo maior incidência no mercado hoteleiro e hospitalar. Esta solução de TV da Nonius Software oferece uma experiência interactiva ao hóspede ao disponibilizar várias opções de entretenimento e acesso a conteúdos de elevada qualidade e interesse. O hóspede tem à sua disposição serviços de IPTV e VOD em alta definição, acesso a canais de rádio *online*, serviços interactivos (compras através do serviço de quartos, reservas e alugueres nos serviços de desporto e lazer), serviços informativos (meteorologia, farmácias, voos, mapas

e noticias), acesso à Internet (na TV ou por Wi-Fi), interacção com dispositivos móveis com iOS ou Android, jogos, leitor de conteúdos por USB e serviços de localização via *wireless* integrado com sistemas CCTV (*Closed-Circuit Television*).

Figura 7 Esquema alusivo aos serviços e funcionalidades disponíveis na plataforma NONIUS.TV [17]

Muitos dos serviços da NONIUS.TV são baseados em interfaces *web*, via *browser* (alguns através de w*idgets*), que permitem a interacção do utilizador com diversos serviços *web*, possibilitando acesso ao Facebook, YouTube XL, Messenger, entre outros serviços.

O controlo de todos os serviços com custo associado, é feito através de uma interface simples e transparente que é incluído no próprio sistema de gestão das instalações, chamado PMS (*Property Management System*). Este possibilita a facturação directamente através da interface da NONIUS.TV, permitindo a visualização dos custos associados à estadia na própria televisão. É uma ferramenta para comunicar com o hóspede e promover os serviços do Hotel, ao mesmo tempo que reduz os custos de operação e gera receita.

Tudo isto é organizado numa interface gráfica (Figura 8) que permite uma fácil navegação, com um *design* (menus, submenus, imagens e cores) personalizado à imagem do hotel, facilitando utilização do sistema por parte do hóspede.

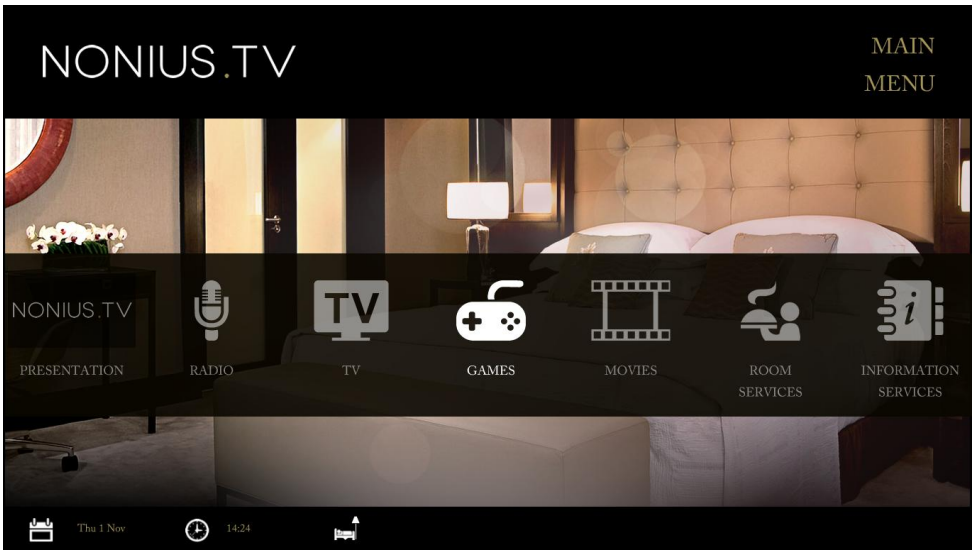


Figura 8 Exemplo de uma das interfaces gráficas da NONIUS.TV

3.2. ARQUITECTURA E COMPONENTES

A arquitectura do sistema NONIUS.TV é composta por vários dispositivos distintos, que estão distribuídos por três sistemas principais: o *In-Room Subsystem* (*Frontend Equipments*), o *Datacenter Subsystem* (*Backend Equipments*) e o *Management Subsystem* (*Front-office/Back-office*).

O esquema representativo na Figura 9, exemplifica a interligação dos diversos componentes direccionados para a área de hotelaria.

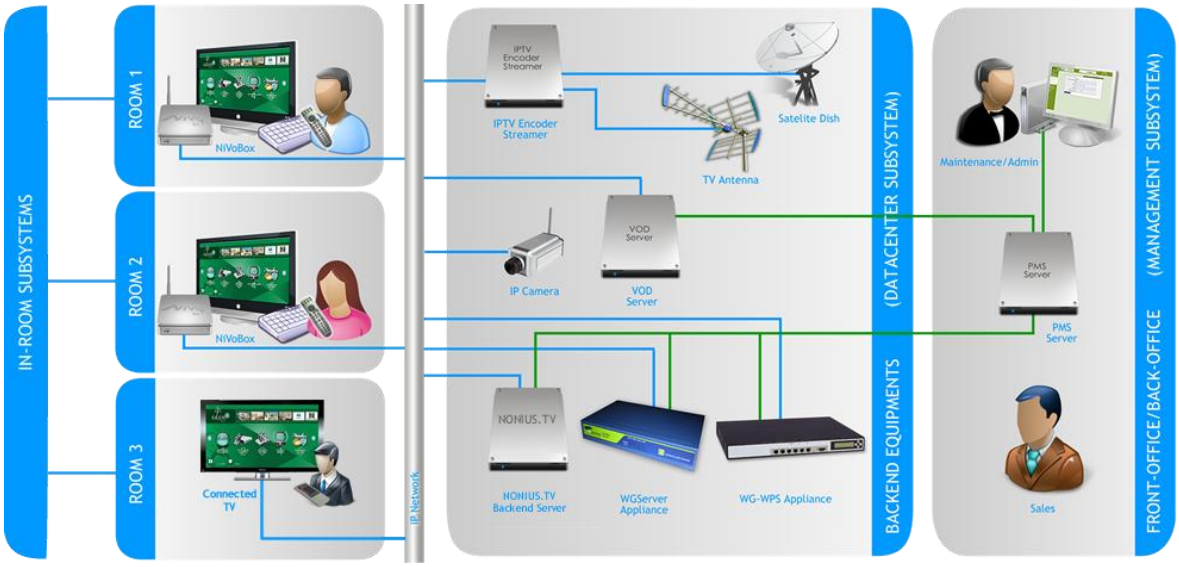


Figura 9 Interligação dos dispositivos relacionados com o sistema NONIUS.TV

3.2.1. *IN-ROOM SUBSYSTEM (FRONTEND EQUIPMENTS)*

Este subsistema é o implementado dentro dos quartos dos clientes e contém os seguintes componentes:

- *Set-Top Box* multimédia (STB): Esta pode ser externa ou já implementada na própria televisão (*Connected TV*);
- Acessórios: Comando, cabos de conexão, receptor de infravermelhos e um teclado óptico;
- Ecrã de Televisão: Modelos compatíveis, que usem, de preferência, o modo chamado *Hotel Mode*, que permite uma melhor integração com a STB.

Este é o ponto final da solução NONIUS.TV, onde todo o conteúdo disponível é apresentado ao cliente final, de uma forma funcional e fácil de usar.

3.2.2. *DATACENTER SUBSYSTEM (BACKEND EQUIPMENTS)*

Os equipamentos do *backend* são os seguintes:

- Servidor NONIUS.TV *Backend*: Este serve de sistema central de gestão da NONIUS.TV e de servidor de aplicações. Faz o armazenamento e gestão de todos conteúdos, assim como a administração das STBs. Disponibiliza também uma interface de facturação usando sistemas PMS e adquire conteúdo de forma automática através da Internet (Serviços de Informação);
- Subsistema de IPTV: Converte sinais de entrada de DVB-S, DVB-T, DVB-C (*Digital Video Broadcasting – Satellite, Terrestrial* e *Cable*) ou analógico para um formato digital que possa ser distribuído por uma rede IP, transmitindo os conteúdos IPTV para a infra-estrutura local;
- Subsistema de VOD: Armazena o conteúdo VOD numa infra-estrutura NONIUS.TV. Esta distribui *streams* de vídeos em alta-definição no formato MPEG-2 e MPEG-4, para toda a rede IPTV da NONIUS.TV. A vantagem deste tipo de abordagem é o facto de este ser capaz de realizar todas as funcionalidades (*fast-forward, fast-rewind, pause*, etc) directamente a partir da RAM, o que não requer um armazenamento adicional ou um maior processamento por parte da STB. Isto permite ter uma STB mais pequena, mais rápida e energeticamente mais eficiente;
- Subsistema de acesso à Internet: Permite o acesso à Internet do servidor *backend* de forma a ser distribuído para sistema da NONIUS.TV a partir da STBs e,

opcionalmente, permite às STBs distribuir Internet através de uma rede local sem fios. Este proporciona também o acesso de gestão para suporte e resolução de problemas;

- Subsistema de *switching*: Faz toda a distribuição de conteúdos e de dados na infra-estrutura da NONIUS.TV e permite a interligação dos diversos componentes do sistema.

3.2.3. *MANAGEMENT SUBSYSTEM* (*FRONT-OFFICE/BACK-OFFICE*)

A solução da NONIUS.TV utiliza sistemas operativos adaptados a cada um dos seus subsistemas:

- Uma versão personalizada do sistema operativo Linux para fins de gestão e configuração dos conteúdos multimédia no servidor *backend* da NONIUS.TV;
- Um sistema operativo Linux incorporado executado em tempo real, configurado pela própria empresa, para suportar vários modelos de TVs, disponibilizando os conteúdos fornecidos pelo servidor *backend*, personalizados ou não, para o utilizador final de acordo com o *design* da empresa.

O sistema operativo do servidor NONIUS.TV *Backend* foi desenhado para se comportar como um sistema centralizado de gestão com a capacidade armazenar e administrar conteúdos, aplicações, gerir e instalar as STBs e actuar como uma interface com vários sistemas PMS disponíveis.

Todas as suas funcionalidades estão disponíveis através de duas interfaces *web* especialmente criadas para o efeito e que são essenciais durante a implementação e testes do sistema NONIUS.TV:

- NONIUS.TV Config (Figura 10) - Composto por dois menus (*System* e *Administration*), esta interface serve para configurações iniciais, manutenção e *backup* das funcionalidades dos servidores. O menu S*ystem* permite criar/gerir categorias aplicadas e criação/configuração de itens. O menu *Administration* permite *backups* de configuração do servidor, criar *passwords* de utilizador, aplicar *upgrades*, fazer *reset* ao servidor para aplicar as definições de fábrica, configuração de SSH/Open VPN e aplicar licenças. Utilizado quase unicamente pela Nonius, quando é necessário configurar ou testar algum sistema à distância.

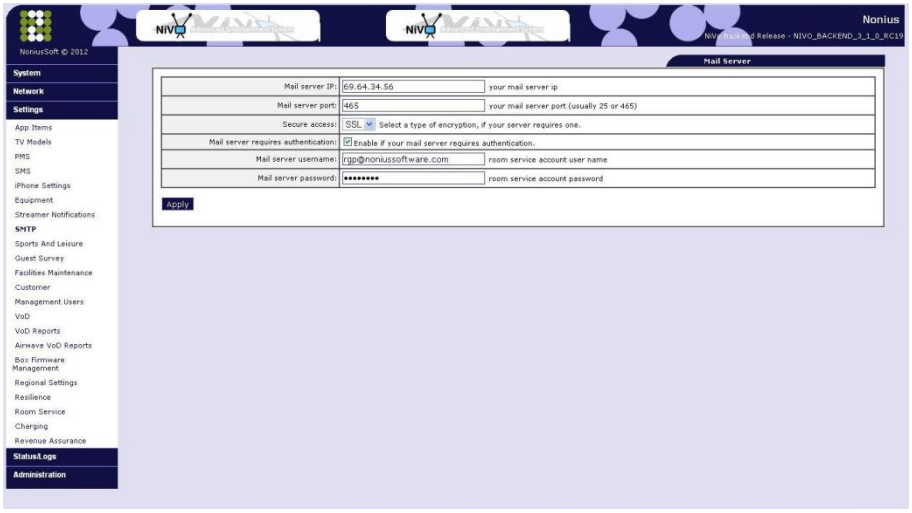


Figura 10 NONIUS.TV Config

- NONIUS.TV Manager (Figura 11) – Este é a principal interface de gestão dos conteúdos da NONIUS.TV. É constituída por diversos menus que contêm as opções disponíveis e as suas combinações. Alguns dos menus principais são: *The General Menu, Box Management, IPTV Management, VOD Management, Browser Management, Information Services Management, Corporate Channel, Room Services, Sports and Leisure, Status/Logs, Administration.* Os restantes menus são usados para aplicações específicas das plataformas. Esta é a interface que é utilizada pelo *staff* do Hotel, quando é necessário alterar, acrescentar ou editar algum conteúdo ou alguma configuração simples no sistema.

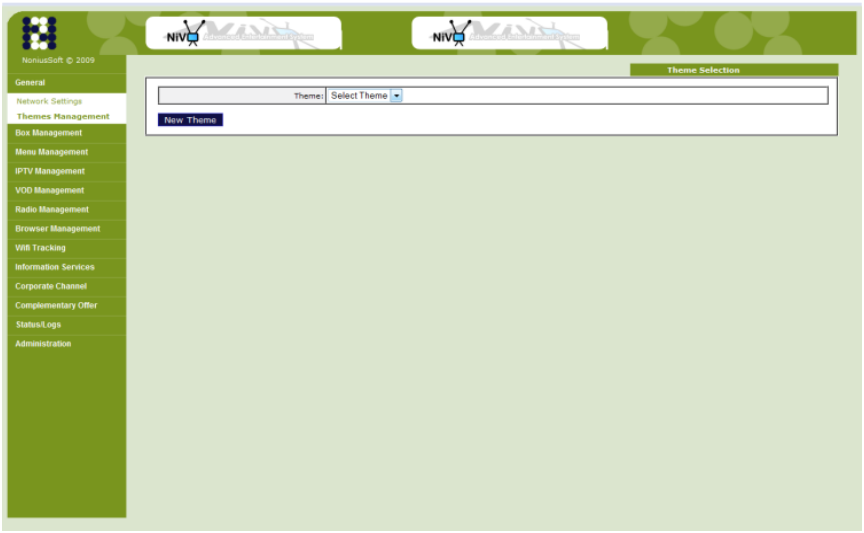


Figura 11 NONIUS.TV Manager

3.3. GAMA DE PRODUTOS NONIUS.TV

Neste momento a Nonius apresenta três soluções de televisão disponíveis para o mercado hoteleiro e hospitalar, umas mais completas que outras, mas todas com diferentes características. As três soluções de TV interactiva que a Nonius fornece são: NONIUS.TV | Premium, NONIUS.TV | Android™ TV e NONIUS.TV | LG Pro:Centric™.

3.3.1. NONIUS.TV | PREMIUM

A solução NONIUS.TV | Premium é a mais completa das soluções de televisão interactiva que a Nonius pode oferecer ao mercado hoteleiro actualmente. É uma solução baseada em tecnologias IPTV, que oferece um Ponto de Acesso Wi-Fi em cada STB e pode ser integrada com outros dispositivos e sistemas, como por exemplo os painéis *MediaHub™*.

As principais vantagens e características deste produto é a Internet na TV, o ponto de acesso Wi-Fi integrado na STB e é possível ser usado como sinalética digital.

3.3.2. NONIUS.TV | ANDROID™ TV

A solução de Televisão Interactiva para hotelaria da Nonius foi implementada recentemente para plataformas Android, o sistema operativo que está presente em muitos *Smartphones* e *Tablets.*

O hoteleiro não tem a necessidade de actualizar as TVs LCD ou LED que já tem, para oferecer a experiência Android TV aos seus hóspedes.

Mantendo as funcionalidades da solução NONIUS.TV | Premium, a plataforma Android TV acrescenta o desempenho, a interface e usabilidade dos jogos (Figura 12) e *apps* do Android Market™. Tudo isto tem um menor custo de aquisição em relação a outras soluções que só é possível graças à massificação da tecnologia Android.

Figura 12 Exemplo do jogo *Angry Birds* no sistema Android™

Este sistema é uma mais valia pelo menor custo de aquisição, pela alta performance que a própria plataforma Android já possui, pela variedade de aplicações e jogos disponíveis e pelo elevado desempenho gráfico.

3.3.3. NONIUS.TV | LG PRO:CENTRIC™

Para entrar no mercado das *Connected TVs*, a Nonius adaptou a sua plataforma televisiva para as TVs LG Pro:Centric™ (Figura 13), sem a necessidade de ter uma STB externa.

Figura 13 Televisão LG Pro:Centric™

As principais vantagens e características deste sistema é a sua fácil instalação e por ter menos componentes activos (*Connected TV*), e a possibilidade de usar a mesma TV para quartos, salas de reuniões e áreas públicas. Para além disso, não necessita de STB externa, tem uma interface HD customizada, é de fácil utilização, faz *zapping* em menos de 1 segundo, suporta alta definição até 1080p e é possível fazer manutenção remota da TV.

A Tabela 1 faz uma análise comparativa entre as soluções disponíveis da NONIUS.TV, apresenta de forma sucinta os respectivos serviços e características.

Tabela 1 Comparação dos serviços e características dos diferentes produtos NONIUS.TV

		Soluções		
		Premium	**Android™ TV**	**LG Pro:Centric™**
Serviços	Mensagem de boas-vindas / Serviço de mensagens	✓	✓	✓
	Video-on-Demand	✓	✓	✓
	Widgets	✓	✓	×
	Compra de serviços do hotel (spa, golf, etc)	✓	✓	✓
	Room Service / Serviços complementares	✓	✓	✓
	Integração com sistema PMS do hotel	✓	✓	✓
	Conta na TV	✓	✓	✓
	Express check-out	✓	✓	✓
	Inquérito de satisfação	✓	✓	✓
	Jogos	✓	✓	✓
	Canais de TV	✓	✓	✓
	Canal Corporativo / informativo	✓		
Características	TV Interactiva	✓	✓	✓
	Internet na TV	✓	✓	×
	Ponto de Acesso *Wireless*	✓	✓	×
	HD TV	720p/1080p	720p/1080p	720p/1080p
	Media Hub	Automático	Manual	×
	Set-top Box	Externa	Externa/Embebida na TV	Interna
	Remote APP	✓	✓	×
	Mercado/Loja de Aplicações	×	*Android Market™*	×

3.4. PERSONALIZAÇÃO DO SERVIÇO NONIUS.TV

O cliente ao escolher um dos serviços da NONIUS.TV, seja ele qual for, tem a possibilidade de personalizar toda a interface gráfica da aplicação (menus, imagens, botões, etc.), de acordo com a sua imagem corporativa.

Para que a personalização da interface seja possível deverão ser considerados alguns requisitos necessários para a sua construção.

3.4.1. COMPONENTES UTILIZADOS E AS SUAS CARACTERÍSTICAS

Como um hotel recebe diversas pessoas de enumeras nacionalidades, é importante que a plataforma suporte diversas línguas, permitindo um sistema de interacção multi-língua. Assim, quando o cliente fizer *check-in* do quarto, a televisão deve passar automaticamente para a língua nativa do mesmo ou para outra língua que tenha sido referenciada no acto de *check-in*. Neste momento, o sistema só tem disponíveis cinco línguas: Inglês (EN – por defeito), Português (PT), Alemão (DE), Espanhol (ES) e Francês (FR).

Estes requisitos servem apenas para as interfaces de navegação. Para cada *plugin* ou aplicação específica, a personalização deverá passar por uma especificação mais detalhada com o cliente. O termo *plugin* é utilizado para referir os serviços disponibilizados pela mesma. Cada serviço corresponde a uma pequena aplicação que pode ser acedida através de um menu. Ao entrar nessa aplicação, esta irá conter diversos ecrãs à medida que se vai interagindo com a mesma. Neste projecto, por norma, cada *plugin* tem a sua própria classe, tornando mais fácil a adição de novos serviços ou funcionalidades.

São vários os componentes utilizados para a criação de um tema para uma interface. De todos os componentes, os que estão mais presentes em todos os ecrãs da plataforma são: o cabeçalho (*header*), fundo (*background*), o rodapé, as barras de fundo e os ícones dos menus, como se pode observar na Figura 14.

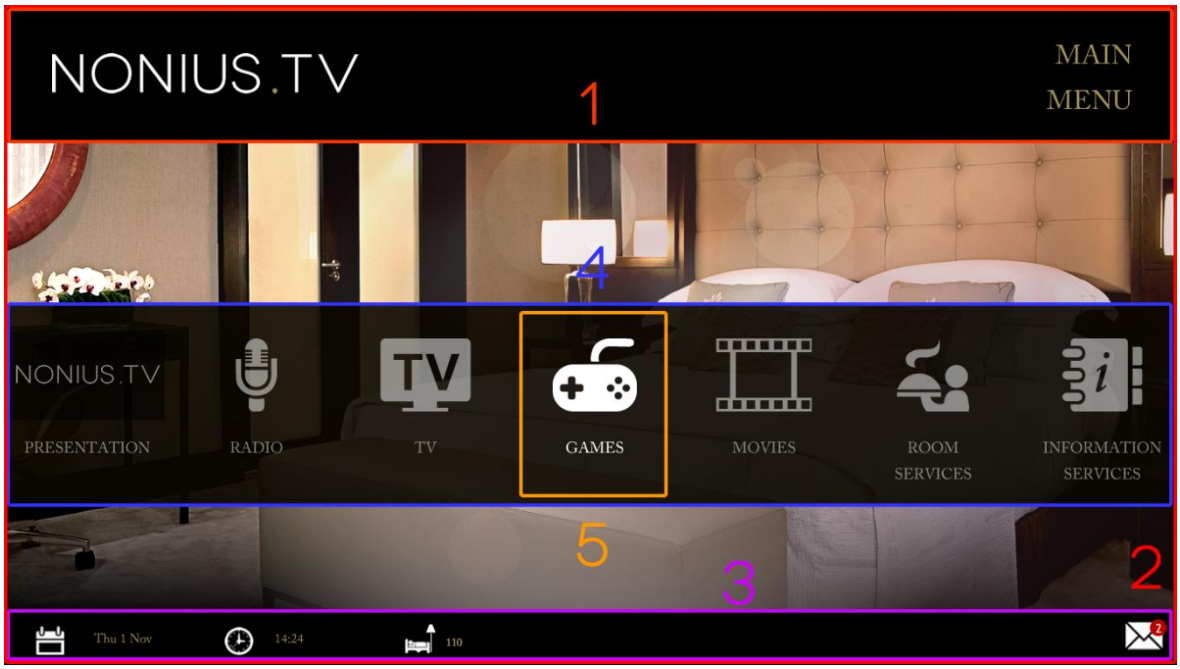


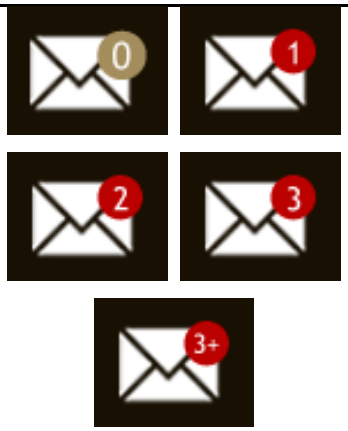
Figura 14 Apresentação dos componentes gráficos mais usados na aplicação

1. **Cabeçalho (*header*)**: Este componente serve para identificar o ecrã em que o cliente está actualmente. Geralmente este apresenta o logotipo da empresa ou um ícone representativo do *plugin* actual. O texto nele presente é sobreposto à imagem e é alterado pela aplicação conforme o idioma que esteja a ser utilizado;

2. **Fundo (*background*)**: Esta imagem irá ser utilizada no fundo de todos os ecrãs dos diversos *plugins*. Logo, vão haver sempre outras imagens/componentes sobrepostos a esta.

3. **Rodapé**: Este contém quatro ícones representativos da data, hora, quarto e número de mensagens não lidas. Seguido aos três primeiros estão as indicações da data, hora e quarto actual. Em relação ao ícone do número de mensagens não lidas este varia consoante o número de mensagens por ler na caixa de mensagens. Este tem um ícone para 0, 1, 2, 3 e mais que 3 mensagens, perfazendo um total de cinco ícones. Na Tabela 2 são listados os vários componentes.

Tabela 2 Componentes apresentados no rodapé

Data	Hora	Quarto	Número de Mensagens Não Lidas
			

4. **Barras de fundo**: Estas barras de fundo são utilizadas em todos os *plugins* e servem para tornar a aplicação mais elegante, mas acima de tudo permitem dar uma melhor percepção do conteúdo que é apresentado por cima, seja ele texto ou imagens.
5. **Ícones**: Os menus são constituídos por ícones representativos de cada *plugin* ou de cada funcionalidade. Existem dois estados para cada ícone do menu: seleccionado ou não seleccionado. Podiam ser criados dois ícones para os diferentes estados, mas de forma a optimizar o sistema foi usado um só ícone. A diferenciação de seleccionado para não seleccionado foi feita variando a opacidade de cada ícone, ou seja, o ícone seria 100% opaco quando estivesse seleccionado e 50% opaco quando estivesse não seleccionado (Figura 15). Outra forma de selecção usada noutros ecrãs foi aumentando o ícone que estivesse seleccionado em relação aos não seleccionados.

Figura 15 Exemplo de um ícone seleccionado e não seleccionado

Os restantes componentes utilizados ao longo da aplicação variam de ecrã para ecrã. Durante a interacção com a interface, o hóspede depara-se com enumeras operações que são necessárias fazer com o comando, e algumas delas não são imediatas se não estiver presente no ecrã algo a indicar o que fazer. Ao navegar num menu é necessário premir as teclas de navegação (Direita, Esquerda, Cima ou Baixo), de forma a escolher o pretendido. Estas e outras operações, apesar de serem muitas vezes intuitivas, são reforçadas com uma indicação no ecrã (através de ícones e texto) que dê informações ao hóspede do que é possível fazer no ecrã em que se encontra.

Em muitos dos *plugins* o hóspede necessita de controlar aonde quer ir e o que quer fazer através de opções que são apresentadas no ecrã, seja voltar para o ecrã anterior, seja para efectuar uma compra, etc. Para isso, são usados ícones que representam os botões do comando e uma pequena descrição em texto, tornando a interface mais *user friendly*.

Na Tabela 3 são apresentados os vários ícones de navegação e de opções disponíveis nos vários ecrãs da aplicação.

Tabela 3 Apresentação dos ícones de navegação e opções presentes nos vários *plugins*

Ícones	Tecla	Descrição
	*Right*	Indica que pode deslocar para o lado direito num menu.
	*Left*	Indica que pode deslocar para o lado esquerdo num menu.
	*Up*	Indica que pode deslocar para cima numa lista.
	*Down*	Indica que pode deslocar para baixo numa lista.
	*Ok*	Indica que pode confirmar operações ou entrar num menu seleccionado.
	*Back*	Indica que pode ir para o ecrã anterior.
	*Menu*	Indica, na sua maioria, que pode ir para o menu principal.
	*Exit*	Indica que pode sair de alguma operação ou *plugin.*
	*Green*	Indica, na sua maioria, para confirmar operações ou outros.
	*Red*	Indica, na sua maioria, para anular operações ou outros.

	*Blue*	Permite diversas operações, dependendo do *plugin.*
	Yellow	Permite diversas operações, dependendo do *plugin.*

Outro dos requisitos necessário para a construção de um novo tema para a plataforma, é a normalização no tamanho dos ícones e das imagens utilizadas. Como os ícones e as imagens têm posições e espaçamentos fixos, que estão definidos no próprio código da aplicação, tendo como base a resolução utilizada pela televisão da LG Pro:Centric (1360x760), estes não podem ser alterados posteriormente. Essa alteração só poderá ser feita no próprio código fonte. Uma vez que a criação de um novo tema baseia-se só na alteração dos conteúdos gráficos e não do código fonte, é necessário haver uma dimensão normalizada desses conteúdos. As medidas dos principais componentes são:

- *Background* – 1360x760;
- *Header* – 1360x160;
- Ícones de Menu – 180x280;
- Ícones de navegação e de operações – 40x40.

Para um maior detalhe, no Anexo A é feita uma apresentação da árvore da estrutura de ficheiros implementada nos temas presentes no *backend* da NONIUS.TV.

4. SMART TV LG PRO:CENTRIC

O objectivo principal do presente trabalho foi a migração das soluções NONIUS.TV para a Smart TV LG Pro:Centric. A Pro:Centric da LG é uma plataforma unicamente dedicada para difusão de dados para diversas aplicações e serviços, especialmente desenvolvida para o segmento hoteleiro. Desta forma, a empresa inicia a migração dos seus serviços e funcionalidades para este modelo de Smart TV.

Para a construção deste sistema foi necessário recorrer à linguagem de programação suportada pela API da televisão. A Pro:Centric é uma plataforma baseada em aplicações GEM (*Globally Executable MHP*) e Flash Lite 3.0 (explicados mais à frente), suportando unicamente o ActionScript 2.0. Com uma API e uma arquitectura muito próprias, é possível controlar todo o *hardware*, desde a televisão até ao comando, o que permite uma maior margem de desenvolvimento para novas funcionalidades.

Este capítulo irá fazer uma breve apresentação da linguagem de programação utilizada e da API da televisão, assim como do ambiente de desenvolvimento. Para contextualizar, será feita uma descrição de um dos protocolos usados para a implementação de uma das partes da aplicação, o RTSP.

A linguagem de programação utilizada e a única permitida pela API da LG Pro:Centric foi o ActionScript 2.0. A linguagem foi implementada no ambiente de desenvolvimento FlashDevelop.

4.1. FLASH: ACTIONSCRIPT 2.0

O Flash é um *software* de uso bastante comum que foi desenvolvido pela Adobe. A sua utilização permite criar programas baseados em animações de elementos gráficos vectoriais com interfaces de navegação em *full-screen* e ilustrações gráficas. É de interactividade simples e possui um formato de ficheiro redimensionável, pequeno o suficiente para permitir o *stream* através da uma ligação normal de Internet. Este *software* está bastante presente na *web* (páginas *web*, aplicações, etc.), tanto pela sua velocidade de execução, como pela forma que esta trabalha com os elementos gráficos. Os arquivos finais de *flash* são do tipo *Shockwave Flash File* (SWF) e são compactos, eficientes e concebidos para uma utilização optimizada. Para ter suporte a este tipo de aplicações, basta usar um *software* de leitura gratuito, por exemplo, o Adobe Flash Player.

Introduzido no *Flash MX 2004* e no *Flash MX Profissional 2004*, o ActionAcript 2.0 (AS2) é uma grande revisão gramatical do ActionScript que era usado em Flash 5 e Flash MX (retroactivamente apelidado de ActionScript 1.0). O AS2 faz uma alteração relativamente pequena no que toca ao tempo de execução da linguagem, mas melhora radicalmente o desenvolvimento orientado a objectos em *Flash* formalizando a sintaxe e metodologia da programação orientada a objectos (OOP – *Object Oriented Programing*).

Apesar do ActionScript 1.0 (AS1) poder ser usado de uma forma orientada a objectos, esta linguagem não possuía um vocabulário formal para a criação de classes e objectos. O AS2 adiciona um suporte sintáctico para os tradicionais recursos orientados a objectos com a introdução do conceito *class* para a criação de classes e *extends* para estabelecer uma herança [18].

A Tabela 4 apresenta a evolução da linguagem ActionScript e a sua relação com o pacote Flash.

Tabela 4 Evolução da linguagem ActionScript e a sua relação com o pacote Flash

Ano de Lançamento	Versão do ActionScript	Informação
2000	1.0	Lançado juntamente o *Flash 5*. Evoluiu a partir das *Actions* do Flash 4.
2003	2.0	Surgiu com o lançamento do Flash MX 2004 e do Flash Player 7.
2006	3.0	Lançado juntamente com o Adobe Flex 2.0 e o Adobe Flash Player 9.

Em vez de tentar melhorar o AS2, adicionando novas funcionalidades e fazer ajustes de performance, a Adobe decidiu reformular o AS2 para chegar ao AS3. Uma mudança no AS3 é a estrita adesão de boas práticas de codificação. Com o AS2, é possível deixar passar muitos formalismos de sintaxe durante a codificação sem que sejam detectados como erros. No AS3 isso já não acontece. A utilização de variáveis globais foi também minimizada em AS3, forçando os programadores a usar OOP de uma forma mais correcta. Outro dos pontos positivos do AS3 é quantidade de métodos que este já possui para tornar mais fácil a implementação e percepção do código, ou seja, enquanto que em AS2 são necessárias enumeras linhas de código para fazer o que é pretendido, em AS3 já existem métodos pré-definidos, tornando a construção do código muito mais simples [19].

Em conclusão, a linguagem que deveria ser usada no projecto seria o ActionScript 3.0, mas existem limitações por parte do *hardware* que não tornam possível a sua utilização. A API da televisão vem pré-definida de fábrica, e esta utiliza Flash Lite 3.1 para reproduzir as aplicações, que não é mais nem menos que uma versão mais leve do Adobe Flash Player. O Flash Lite foi inicialmente desenhado para dispositivos móveis e a versão 3.1 só suporta o ActionScript 2.0.

O ambiente de desenvolvimento usado para construção da aplicação foi o FlashDevelop. O Anexo B faz uma breve apresentação deste *software*.

4.2. *MIDDLEWARE* DA LG PRO:CENTRIC – HCAP FLASH

Pro:Centric™ é um ambiente de aplicações desenvolvido pela LG Electronics, que é disponibilizado em certos modelos de televisão. Foi dentro deste ambiente que foi

implementada a solução NONIUS.TV, alterando-a de forma a ser compatível com esta nova plataforma.

O objectivo principal da escolha deste ambiente de aplicações por parte da Nonius foi o facto de este ter sido especialmente desenvolvido para o segmento hoteleiro e por proporcionar uma experiência de TV interactiva sem serem necessários componentes externos à televisão.

4.2.1. INTRODUÇÃO AO HCAP FLASH

A plataforma Pro:Centric é uma implementação da especificação do HCAP (*Hotel Common Application Platform*). Este é um *middleware* próprio da LG para transmissão de dados que presta serviço interactivo especializado para ambientes comerciais, tais como hotéis e hospitais. De forma a perceber a plataforma Pro:Centric, é essencial ter conhecimento sobre o HCAP [20].

O HCAP é baseado em DVB-GEM (*Digital Video Broadcasting – GEM*) e DVB-MHP (*Digital Video Broadcasting – Multimedia Home Platform*). Este foi desenhado dentro do enquadramento que o GEM estabelece para as especificações de equipamentos terminais.

O GEM é uma especificação DVB de um *middleware* baseado em *Java* para receptores de transmissão de TV e terminais IPTV. Este define um conjunto de normas e funcionalidades comuns que são independentes da sinalização e protocolos de uma rede específica de transmissão e permite escrever aplicações interoperáveis para a TV [21]. Desta forma, o HCAP inclui e adere completamente ao DVB-GEM. Esta usa uma aplicação de sinalização chamada OCAP Profile 1.0. O OCAP (*Open Cable Application Platform*) é uma camada do sistema operativo projectada para produtos electrónicos de consumo que se ligam a um sistema de televisão e que foi criado para fornecer uma base comum para sistemas de cabo digital de TV [22].

As aplicações HCAP são transmitidas a partir de servidores e executados em receptores de TV digital. As aplicações podem conter código de controlo em Flash, gráficos, áudio e vídeo. O HCAP compatível com o *middleware* dos receptores fornece o ambiente necessário para as aplicações serem executadas.

Actualmente, o HCAP define aplicações baseadas em Flash. No entanto, este pode suportar outro formato de aplicações baseadas em diferentes ambientes que podem ser integrados com o *software* de sistema HCAP.

4.2.2. ARQUITECTURA DO HCAP FLASH

A arquitectura do HCAP Flash inclui a plataforma Flash da Adobe Corporation. O HCAP Flash tem funcionalidades como XAIT (eXtended Application Information Table), *Object Carousel* e Gestor de Aplicações. O *Object Carousel* define que o fluxo de transmissão é fornecido de forma cíclica.

O *middleware* compatível com o HCAP Flash fornece um ambiente de execução para as aplicações em *Flash*. Os recursos necessários, sejam eles gráficos ou mecanismos de processamento de canais de áudio, vídeo e comunicação, são fornecidos pelo próprio *middleware*. A disposição arquitectónica das Aplicações HCAP Flash, do *middleware*, da plataforma em Flash e dos descodificadores são apresentados na Figura 16.

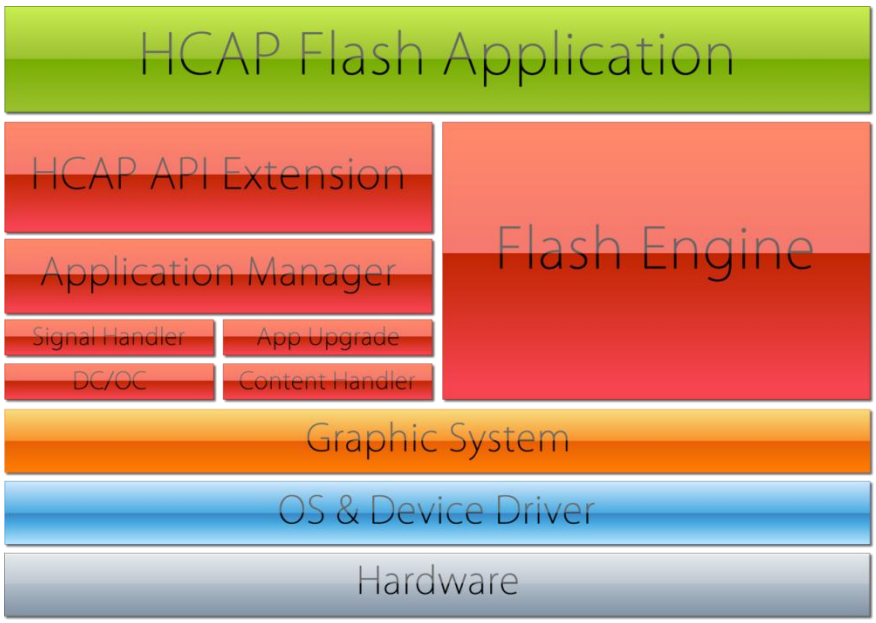


Figura 16 Disposição arquitectónica das Aplicações HCAP Flash

Cada uma das camadas apresentadas tem atribuída um conjunto de responsabilidades/funcionalidades:

- ***Application Manager:*** Este módulo faz a gestão de todo o ciclo de vida, desde o início até à destruição dos conteúdos *Flash;*
- ***App Upgrade:*** Efectua o *download* e armazena os conteúdos *Flash;*

- ***Content Handler:*** Armazena o ciclo de vida dos conteúdos *Flash* e guarda-o depois de ser recebido através do mecanismo DC/OC (Data Carousel / Object Carousel);
- ***Signal Handler:*** Recebe e interpreta o sinal XAIT/AIT.
  - *AIT Handler*: AIT (*Application Information Table*) é usado em *middleware* CDTV (*Conventional Definition Television*) baseado em *Flash*. É transmitido a cada serviço que contenha uma aplicação Flash. O AIT contém uma entrada para cada aplicação que é válida para esse serviço. O *AIT Handler* gere a recepção de dados AIT, podendo-o bloquear, dependendo do estado do sistema;
  - *XAIT Handler*: XAIT (*eXtended Application Information Table*) é uma tabela usada para a execução e gestão do ciclo de vida das aplicações não ligadas, dentro da especificação da plataforma de aplicações OCAP. A aplicação não ligada é inicializada e controlada pelo XAIT. Este *handler* gere a recepção de dados XAIT. Este pode bloquear a API, dependendo do estado do sistema.
- ***HCAP API Extension:*** são APIs para as funções solicitadas adicionalmente pelos Integradores de Sistemas;
- ***DC/OC Engine:*** módulo que recebe os ficheiros de conteúdos em *Flash*;
- ***Graphic System:*** fornece funções gráficas relacionadas, incluindo a gestão de janelas.

O HCAP Flash dispõe unicamente de duas classes onde estão definidos os diversos métodos de interacção com a televisão:

- **`IHcap`**: A class `IHcap` pertence ao pacote `procentric`, que é fornecido para os programadores que desenvolvem conteúdos em Flash, no *middleware* da DTV (*Digital Television*) comercial baseado em Flash. Esta classe fornece vários métodos que permitem dar acesso às diversas funcionalidades que a televisão possui. Usando esta API, podem ser criadas aplicações que permitem um controlo mais variado do equipamento. De uma forma mais abrangente, as funções são as seguintes:
  - Funções relacionadas com o modo HCAP;
  - Funções relacionadas com canais de TV;
  - Funções de desenho de vídeo;
  - Funções relacionadas com dados;
  - Funções de controlo de entrada da TV;
  - Funções de controlo de áudio;

- Funções de controlo de tempo;
- Funções de controlo de energia.

- **`IHcapConst`**: Pertencendo também ao pacote `procentric`, a classe `IHcapConst` define constantes para a classe `IHcap` tais como os códigos de resultado e os códigos principais do comando, ou seja, é uma classe que define o mapeamento das teclas do comando de forma a permitir a sua interacção com a televisão.

4.3. PROTOCOLO RTSP

Durante a construção da aplicação, foi necessário estudar novos conceitos e analisar novas tecnologias para implementar os diversos serviços e funcionalidades. Num desses serviços, o VOD, foi necessário implementar um cliente RTSP para fazer o controlo do *streaming* de vídeo.

O RTSP é um protocolo de nível aplicativo para o controlo sobre a entrega de dados com propriedade em tempo real. Este fornece uma estrutura extensível para permitir, de forma controlada, a entrega *on-demand* de dados em tempo real, tais como áudio e vídeo. Tanto pode incluir *feeds* de dados ao vivo como vídeos armazenados. Este protocolo destina-se a controlar várias sessões de entrega de dados, fornecendo um meio para canais de distribuição, como o UDP, *multicast* UDP e TCP, e um meio para a escolha de mecanismos de entrega com base em RTP (*Real-Time Transport Protocol*).

Embora semelhante em alguns aspectos ao HTTP, o RTSP define sequências de controlo úteis no domínio da reprodução multimédia. Não existe um conceito de conexão em RTSP; em vez disso, o servidor mantém uma sessão marcada por um identificador. Uma sessão RTSP não está, de modo algum, ligada a uma única conexão de nível de transporte como uma conexão TCP. Durante uma sessão RTSP, um cliente RTSP pode abrir e fechar muitas ligações de transporte fiáveis para emitir pedidos RTSP. Enquanto a maioria das mensagens de controlo do RTSP são enviadas do cliente para o servidor, alguns comandos vão na direcção contrária, isto é, do servidor para o cliente (Figura 17).

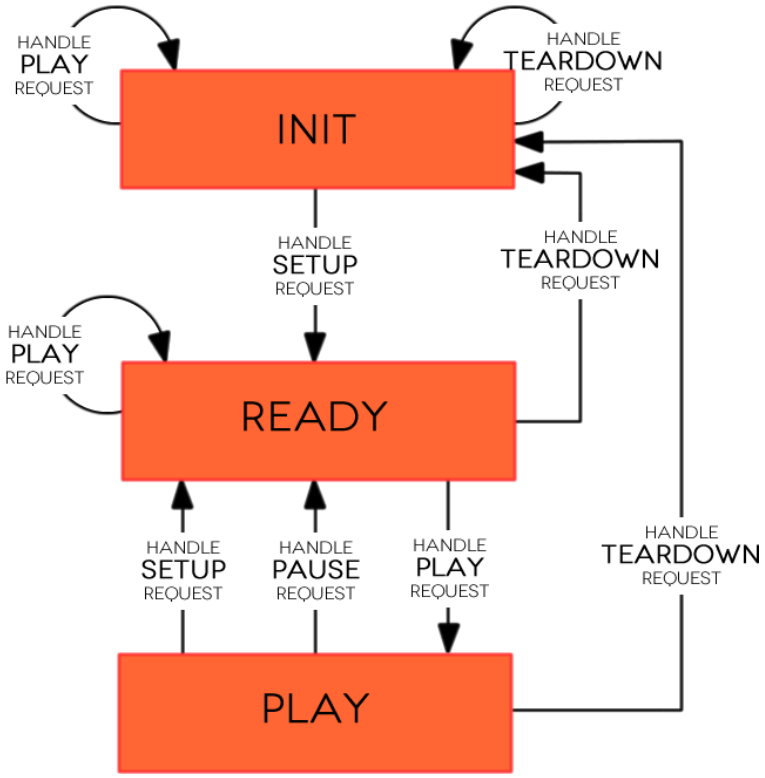


Figura 17 Esquema demonstrativo da troca de pedidos do protocolo RTSP

As *streams* controladas por RTSP podem utilizar RTP, mas a operação de RTSP não depende do mecanismo de transporte utilizado para transportar meios contínuos. O protocolo é intencionalmente similar ao HTTP/1.1 em sintaxe e operação para que os mecanismos de extensão HTTP possam, na maioria dos casos, também ser adicionados ao RTSP.

Alternativamente, pode utilizar um protocolo de transporte sem conexão, como o UDP. O número da porta usado por defeito pela camada de transporte é o 554 [23].

5. IMPLEMENTAÇÃO DO SISTEMA NONIUS.TV NA LG PRO:CENTRIC

O trabalho efectuado na implementação da aplicação para a plataforma NONIUS.TV | LG Pro:Centric, passou pela criação *frontend* de diversos serviços e funcionalidades de forma a estes interagirem com o *backend* e com o utilizador. Ou seja, a implementação feita para o projecto passou por quase todos os *plugins* da aplicação. Isto permitiu ganhar um conhecimento aprofundado sobre o funcionamento dos diversos *plugins*.

O projecto criado no ambiente de desenvolvimento é bastante extenso, possuindo diversos directórios com os ficheiros essenciais e com as diversas classes com o código fonte que foi implementado. A lista seguinte apresenta e explica as classes que foram criadas durante o estágio e apresenta outras das quais foi necessário trabalhar (acrescentando novos métodos ou utilizando os que já existiam). Cada classe está associada a um único ficheiro.

- `DateTime`: Faz o tratamento da data e hora actual, e de quase todos os controlos temporais necessários na aplicação, através da criação de temporizadores, permitindo receber ou enviar dados actualizados, num intervalo de tempo definido, do ou para o

servidor. Esta classe é a que apresenta a data, hora, quarto e número de mensagens não lidas no rodapé do menu principal;

- `HeartBeat`: Faz o registo da televisão no *backend* e informa-o do seu estado (*online*/*offline*);
- `Main`: Faz o arranque da aplicação, executando métodos de *parsing* de ficheiros criados na classe `XML_Parser` que são essenciais para a sua execução e inicia os métodos principais;
- `NiVo_ActivitiesCategories`: Classe que apresenta uma interface gráfica com um menu gerado a partir do conteúdo recebido por um *WebService*;
- `NiVo_ActivitiesProducts`: Apresenta uma interface gráfica onde o hóspede pode seleccionar um produto pertencente a um serviço de desporto e lazer;
- `NiVo_ActivitiesReservation`: Faz a gestão da reserva do produto seleccionado pelo hóspede e posteriormente a taxação do produto comprado;
- `NiVo_Centric`: Faz a centralização de toda a aplicação. É aqui que são inicializadas todas as classes, permitindo um acesso central entre elas. Ao terminar o arranque da aplicação, é nesta classe que é executado o método responsável pela apresentação da interface gráfica do menu principal, assim como a execução e a destruição de qual o *plugin* a ser iniciado ou terminado;
- `NiVo_INBOX`: Apresenta um GUI que dá acesso a uma caixa de mensagens, onde o hóspede pode ler e apagar mensagens enviadas pelo hotel;
- `NiVo_INFOMAPS`: Faz a apresentação de uma simples imagem, neste caso um mapa;
- `NiVo_INFONEWS`: Esta classe recebe um *feed* de notícias, através de um ficheiro XML, faz o *parsing* do mesmo e apresenta o seu conteúdo numa lista organizada por páginas;
- `NiVo_INFOPHARMACIES`: Apresenta uma grelha com diversas farmácias que estão disponíveis através de um XML gerado pelo *backend*;
- `NiVo_INFOWEATHER`: Desenha uma interface gráfica onde são apresentados dados meteorológicos, obtidos através de um XML gerado pelo *backend*;
- `NiVo_INFOSERVICES`: Cria um menu que permite o acesso aos serviços informativos criados pelas classes `NiVo_INFOMAPS`, `NiVo_INFONEWS`, `NiVo_INFOPHARMACIES`, `NiVo_INFOWEATHER` e `NiVo_FLIGHTSHOW`;
- `NiVo_Language`: Apresenta uma *pop-up* com a lista de idiomas disponíveis para a aplicação e onde pode ser feita a alteração do idioma do sistema;

- `NiVo_MainMenu`: Classe que cria o menu principal;
- `NiVo_RoomService`: Apresenta uma GUI onde o hóspede pode seleccionar uma categoria pertencente ao serviço de quartos;
- `NiVo_RoomServiceProducts`: Classe que apresenta uma interface gráfica com um menu gerado a partir do conteúdo recebido por um *WebService*, permitindo criar uma espécie de "carrinho de compras" onde pode ir adicionando produtos e elimina-los quando pretender. No final tem de validar a compra;
- `NiVo_VOD`: Permite o acesso ao serviço de *Video-on-Demand*, apresentando uma lista de categorias com os respectivos filmes;
- `NiVo_WakeUp`: Permite definir um despertador na televisão;
- `PMS_Events`: Faz a gestão dos dados do hóspede que recebe através *backend*, armazenando-os na memória interna da televisão;
- `RTSP_Client`: Cria um cliente RTSP que faz o controlo dos pedidos entre a aplicação e o servidor, durante um *streaming* de vídeo;
- `Soap_Requests`: Faz a gestão dos pedidos e respostas aos *WebServices*.
- `Translator`: Faz as traduções de todas as *strings* existentes na aplicação;
- `TV_Controler`: Classe onde são criados *listeners* que permitem escutar os eventos efectuados no comando, executando os métodos necessários em resposta a essa acção;
- `Utils`: Aqui são criados todos os métodos que são necessários em mais que uma classe, funcionando como uma biblioteca;
- `XML_Parser`: Classe que faz a maior parte dos *parsings* dos ficheiros XML, correspondentes aos menus existentes na aplicação ou outros dados importantes;
- `Wake_Up_Ring`: Cria o ecrã que é executado quando está na hora definida no despertador.

Neste capítulo serão apresentados todos os *plugins* criados neste estágio e as principais funcionalidades implementadas ao longo de toda a aplicação, referenciando a maioria das classes anteriormente descritas. Outras não serão directamente referidas porque só foram criados alguns métodos nessas classes para serem usados pelas classes implementadas.

Para dar uma ideia dos outros *plugins* existentes nesta aplicação mas que não foram directamente implementados neste estágio, o Anexo C apresenta uma breve descrição sobre cada um deles.

5.1. MENU PRINCIPAL (*MAIN MENU*)

Como foi referido anteriormente, o menu principal é o que dá acesso aos diversos *plugins* existentes na aplicação. Os métodos existentes no *plugin* do menu principal são apresentados na Tabela 5:

Tabela 5 Lista de métodos da classe `NiVo_MainMenu` e a sua descrição

	Classe `NiVo_MainMenu`	
	Método	Descrição

Criação do GUI	`createBackgroundContainer()`
	`createHeaderContainer()`
	`createDateTimeContainer()`
	`createMenu()`
	`create_icon()`
	`mm_buttons_tf()`
		Estes métodos criam os componentes gráficos existentes nesta interface: *background*, *header*, rodapé (data, hora, quarto e ícone de mensagens) e menu. Essa criação, em ActionScript, é possível através do uso de `MovieClips` e `TextFields`. O menu é constituído por uma barra, ícones e caixas de texto, por essa razão o `createMenu()` é que irá invocar os métodos `create_icon()` e `mm_buttons_tf()`.

Invisibilidade	`destroy()`
	`hide_background()`
	`hide_header()`
	`hide_buttons()`
	`hide_datetime()`
		É essencial ter um controlo da visibilidade dos diferentes componentes gráficos quando se está a trabalhar em GUI. Quando é necessário transitar para outro ecrã, os componentes que estavam visíveis têm de desaparecer, seja através sua destruição em memória ou tornando-os invisíveis.

Visibilidade	`show_mainmenu()`
	`show_background()`
	`show_buttons()`
	`show_header()`
	`show_datetime()`
		Assim como é necessário esconder os componentes, também tem de ser possível mostrá-los, seja através da sua criação ou tornando-os visíveis. Um componente só dá para ficar visível se este já foi criado antes. O `show_mainmenu()` invoca todos os outros métodos de visibilidade.

Deslocamento	`right()`
	`left()`
	`slide_right()`
	`slide_left()`
		Para ser possível percorrer o menu, é necessário haver um deslocamento. No caso do menu principal, o deslocamento pode ser para o lado direito ou para o lado esquerdo. Ao fazer o deslocamento, dependendo da posição, o menu irá também deslizar (*slide*).

Métodos Específicos	`init()`	Faz a inicialização de todas as classes necessárias para implementação do código do menu principal.
	`web_service_call_no_params()`	Envia pedidos a determinados *WebServices*.
	`soapResponseReady()`	Recebe a resposta aos pedidos feitos pelos *WebServices* utilizados por este *plugin*.
	`update_room_name()`	Faz a actualização do número do quarto.
	`load_buttons()`	Carrega todos os ícones que foram obtidos através *parsing* do ficheiro *theme.xml* que indica quais os botões que vão ser apresentados.
	`startup()`	Este método verifica se é necessário executar automaticamente algum outro *plugin* logo no arranque, através de uma variável presente no ficheiro *variables.txt*.

	`translation_keys()`	Se o *plugin* de arranque for algum canal de televisão ou de rádio, esse canal tem de ser traduzido numa acção que vai apresentar o respectivo canal.
	`change_mm_tf_lang()` `mm_buttons_tf_change()`	Faz a tradução das *strings* que estão nas caixas de texto de cada botão do menu principal, através da invocação do método `mm_buttons_tf_change()`.

Após a criação do menu principal, verificou-se que este não era fluído ao transitar de ícone para ícone, acabando por ser um ponto negativo para a interface. Apesar de isto se dever à pouca capacidade de memória e processamento da televisão, tentou-se contornar o problema analisando outras possíveis razões para isto acontecer:

- A aplicação estar demasiado pesada. Podem existir imagens que ocupem grande espaço em memória ou outros ficheiros carregados que são desnecessários enquanto se está a utilizar o menu principal;
- A forma como o menu está implementado em termos de deslizamento ou a forma como está estruturado;
- Processos desnecessários que estão a ser executados em paralelo com o menu principal e que precisam de ser cancelados porque não estão a ser precisos no momento.

Uma análise detalhada da situação permitiu concluir que o problema era provocado por todos os três pontos anteriores. Como já foi referido anteriormente, todos os temas existentes para as diferentes plataformas da NONIUS.TV contêm para cada botão dos diferentes menus um ícone ‘ON’ e um ícone ‘OFF’. Ou seja, um para quando o botão está seleccionado e outro quando não está seleccionado. Mas a aplicação só usa um único botão, o seleccionado. Para situações em que o botão não está seleccionado, a opacidade do ícone é reduzida, dando a entender ao cliente que aquele botão não é o seleccionado mas sim o que tem a opacidade a 100%. O mesmo acontece com os cabeçalhos de cada *plugin*. Cada *plugin* tinha o seu cabeçalho, mas o que realmente mudava em cada um desses era o texto, que seria o nome de cada *plugin*. Para contornar o problema, criou-se um cabeçalho genérico e a gestão do texto passou a ser feita pela própria aplicação. Esta alteração permitiu optimizar a aplicação ao retirar diversas imagens do pacote do tema, diminuindo assim o pacote final da aplicação e incrementando um pouco o desempenho da mesma.

De forma a permitir melhorias adicionais do menu principal, foi testada uma nova forma do deslizamento dos ícones. A forma inicial está apresentada esquematicamente na Figura 18.

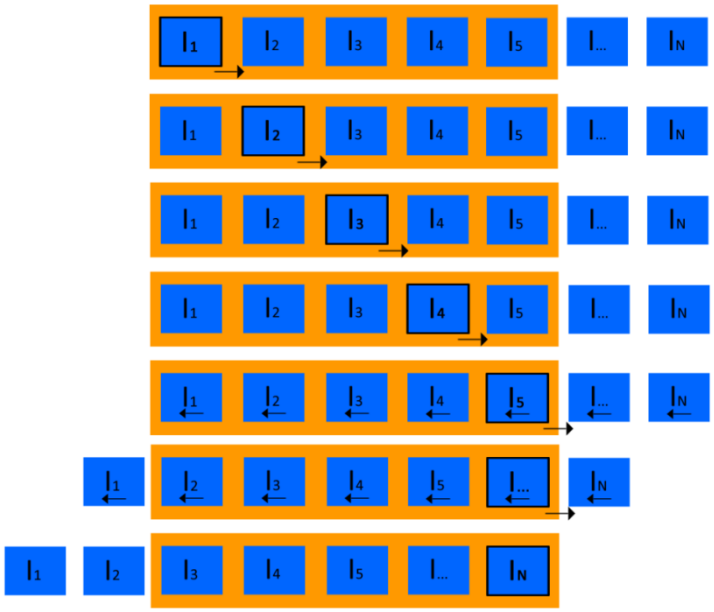


Figura 18 Esquema representativo do funcionamento inicial do menu principal

Neste exemplo, temos um ecrã que só tem a capacidade de apresentar cinco ícones ao mesmo tempo. Como se pode verificar, se a aplicação se encontrar entre o primeiro e o quinto ícone não existe nenhum tipo de deslizamento quando se prime a tecla direita, mas sim uma troca da opacidade do ícone seleccionado (que vai passar a estar desseleccionado) para o próximo (que vai passar a estar seleccionado). Após ultrapassar estes limites, os primeiros ícones deixam de aparecer e começam a ser apresentados os próximos, porque estes ultrapassam o limite da televisão. Este efeito tem o nome de deslizamento horizontal, porque cada ícone terá de ser reposicionado para o lado contrário que o utilizador escolheu. Esse reposicionamento dos ícones, quando o deslocamento dos mesmos é para o lado esquerdo, é feito através do seguinte cálculo:

$$(1)\ x_{final} = x_{inicial} - (w + k)$$

$x_{final} \rightarrow Posição\ final\ do\ ícone\ após\ o\ deslocamento$
$x_{inicial} \rightarrow Posição\ inicial\ do\ ícone\ antes\ o\ deslocamento$
$w \rightarrow Largura\ do\ ícone$
$k \rightarrow Espaçamento\ entre\ ícones$

O mesmo efeito verifica-se quando o deslocamento dos ícones é para o lado direito, mas desta vez soma-se a largura do ícone mais o respectivo espaçamento:

$$(2)\ x_{final} = x_{inicial} + (w + k)$$

Esta abordagem coloca dois tipos de problemas: afecta o desempenho da aplicação e tem um impacto visual negativo, já que o cliente não é capaz de ter uma vista geral sobre os

ícones, porque estes, a partir de um ponto, se encontram fora do limite direito ou esquerdo do ecrã.

A nova proposta de deslizamento do menu (Figura 19) é mais apelativa em termos de construção da interface para o cliente, e da sua fácil utilização e percepção dos conteúdos.

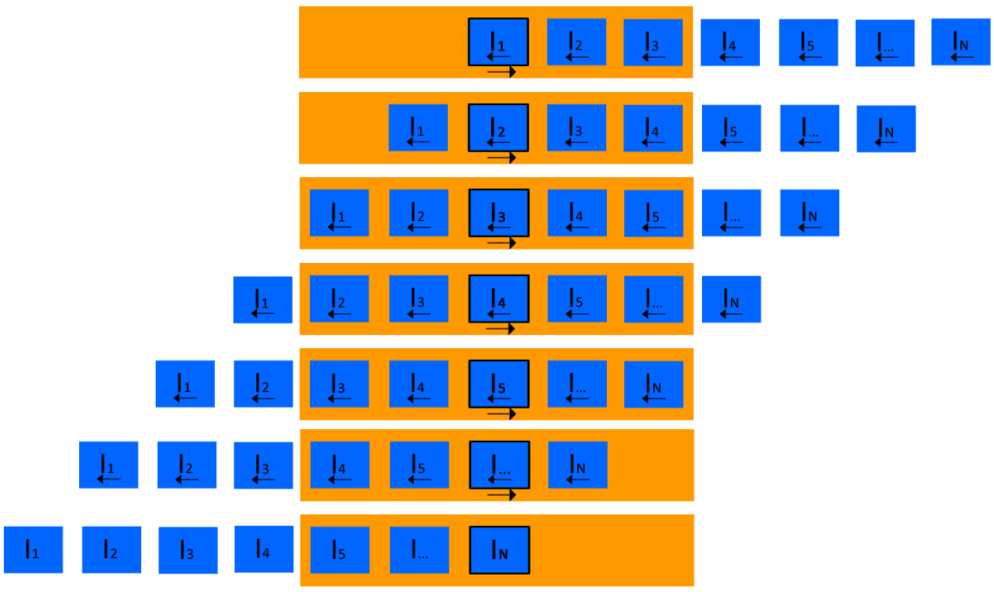


Figura 19 Esquema representativo do novo funcionamento do menu principal

A forma como este menu está construído, permite ao cliente ver sempre com algum avanço os ícones que estão para além do que está seleccionado, facilitando um pouco mais a sua procura. Para dar uma melhor percepção do ícone seleccionado no momento, este encontra-se sempre posicionado no meio do ecrã. A única desvantagem deste menu é que todos os ícones têm que ser reposicionados sempre que se queira percorrer o menu, independentemente de onde se encontrem.

Analisando o resultado final desta abordagem, conseguiu-se concluir que em termos de desempenho do menu não teve uma diferença significativa em relação à abordagem anterior. No entanto, com esta implementação, o cliente consegue ter uma visualização mais abrangente de grande parte do menu, em cada deslizamento.

Outro problema de desempenho detectado estava relacionado com o facto de o menu principal estar a ser executado muito antes de acabarem os processos de inicialização da aplicação.

O arranque da aplicação (feito pela classe `Main`) começa por apresentar um ecrã básico que informa o cliente, em tempo real, da sequência do processo de inicialização do equipamento:

1. A aplicação começa por carregar todas as línguas que suporta, através do *parcing* de ficheiros XML existentes para cada língua (através da classe `XML_Parser`). Cada ficheiro destes contém todas as palavras e frases usadas na aplicação para o respectivo idioma. A televisão arranca no idioma que está definido no `SharedObject`. O `SharedObject` é usado para ler e armazenar quantidades limitadas de dados no computador do utilizador, ou neste caso, na televisão. Estes oferecem partilha de dados em tempo real a objectos que são bastante usados localmente. São um método eficaz para exibir informações que foram deixadas na visita anterior ou adaptar uma visita com base na experiência anterior. A classe `SharedObject` dá uma memória à aplicação [23];
2. De seguida, são carregadas as principais variáveis utilizadas no sistema através do *parsing* do ficheiro de sistema *variables.txt*. Este ficheiro evita que todas as variáveis sejam definidas directamente no código. Assim, qualquer alteração que seja necessária realizar, por exemplo a posição de algumas imagens ou espaços entre ícones, não obriga a uma nova compilação do programa, bastando executá-lo novamente para se verificarem as alterações;
3. Obtém a estrutura de botões que irão ser apresentados no menu principal. Isto é feito através do *parsing* do ficheiro *nivoTheme.xml*, desenvolvido com o intuito de definir uma hierarquia de menus, ou seja, níveis de menus a apresentar e a ordem que vão ser apresentados;
4. Seguidamente são carregados os *paths* de todos os outros componentes que fazem parte do menu principal: o *background*, os ícones do quarto, do relógio, da data e da caixa de correio. Isto é feito através do que está definido no ficheiro *base.xml*;
5. Posteriormente, obtém todos os ícones e imagens necessários para a aplicação através do *parsing* do ficheiro *theme.xml*;
6. Para os *plugins* mais complexos, é necessário fazer o carregamento de alguns elementos que os compõem e distribui-los em estruturas de dados, de forma a estes estarem acessíveis mal seja feito o arranque da aplicação. O VOD, a TV e o Rádio possuem, cada um, uma *playlist* própria em XML com a lista de conteúdos que

estão disponíveis para o respectivo cliente. Para esses conteúdos estarem disponíveis na aplicação é necessário carregar as respectivas *playlists* no arranque da aplicação, de forma a tornar o *plugin* fluído logo após a sua execução.

Após concluídos todos estes processos, foi detectado outro inconveniente, aquando da apresentação dos elementos gráficos do menu principal. Observou-se que estes são apresentados em ordens diferentes, em vez de serem apresentados todos em simultâneo. Percebeu-se então que seria necessário fazer algo que só permitisse apresentar os elementos gráficos quando estes estivessem completamente todos carregados nos respectivos `MovieClips`. Isso seria possível aplicando um *listener* em cada um dos `MovieClips`.

Os *listeners* ou *event listeners* em ActionScript, acompanham o que acontece na aplicação Flash, ficando à espera de uma acção específica. Essa acção, neste caso em particular, é a finalização do carregamento de um conjunto de imagens. Quando o *listener* detecta a acção, o Flash executa processos em resposta ao respectivo evento. A classe `MovieClipLoader` fornece uma série de eventos que são activos em diferentes pontos ao longo do processo de carregamento de um `MovieClip`. Estes eventos permitem personalizar a forma como o código interage com os conteúdos carregados externamente durante todo o processo de carregamento [24].

Para contextualizar, é importante referir que a televisão executa diversos processos em *background* quando esta se encontra no menu principal. A aplicação usa temporizadores (*timers*) para lançar esses processos de forma constante, permitindo obter informações actualizadas pelo *backend*. De minuto em minuto é lançado um *heartbeat* que contém o endereço MAC da TV, a versão em que está, o tempo do próximo *heartbeat* e o tipo de *hardware* (este processo é efectuado através da chamada de métodos presentes na classe `HeartBeat`). Este sistema serve para indicar ao *backend*, que a televisão se encontra ligada, ou seja, indica o seu estado (*online* ou *offline*). De dois em dois minutos é actualizada a informação sobre o hóspede (*WebService* `getRoomInfo`, tratado na classe `PMS_Events`). De três em três minutos, a televisão é registada no *backend*, no caso de haver algo novo a registar. Já de quatro em quatro minutos é feito um ajuste da hora da televisão, de forma a sincronizar o relógio da TV com o do *backend*, porque ao longo do tempo vão-se acumulando pequenos atrasos.

Após aplicar a última solução, o menu principal é apresentado com sucesso de uma só vez e de forma fluída. Com esta abordagem, muitos dos *plugins* só são realmente executados quando o cliente os seleccionar no menu, poupando tempo no arranque da aplicação e dando mais performance ao menu principal.

5.2. CLIENTE RTSP

Este subcapítulo apresenta uma aplicação em Flash que irá permitir a comunicação da televisão com um servidor RTSP, funcionando a televisão como um cliente RTSP. Esta aplicação irá permitir enviar e receber um conjunto de pedidos e respostas que irão possibilitar o controlo de *streamings* de vídeo.

Os métodos existentes na aplicação do cliente RTSP são apresentados na Tabela 6.

Tabela 6 Lista de métodos da classe `RTSP_Client` e a sua descrição

	Classe `RTSP_Client`	
	Método	Descrição
Inicio do Client RTSP	`request_movie()`	Quando o hóspede confirma a compra de um filme, este método é invocado, fazendo o *parsing* dos vários elementos que vêm no URL do filme (IP, Porta e o nome do filme), executando de seguida um `new_connection()`.
	`new_connection()`	Este método começa por receber o endereço do servidor RTSP, a porta a que se vai ligar e o nome do respectivo filme. A conexão é feita através do envio de um pedido OPTIONS.
Envio e recepção de pedidos RTSP	`options()`	O cliente recebe neste método a resposta ao pedido OPTIONS e faz o envio do pedido seguinte, ou seja, um DESCRIBE.
	`describe()`	Este método recebe a resposta ao pedido DESCRIBE e faz o envio do pedido seguinte, ou seja, um SETUP.
	`setup()`	O cliente recebe neste método a resposta ao pedido SETUP e para começar então a fazer o *streaming* do filme, é necessário efectuar o envio de um pedido PLAY.
	`play()`	Este método recebe a resposta ao pedido PLAY e inicia o *streaming* de vídeo.
	`teardown()`	Este método recebe a resposta do servidor após ter sido feito um pedido de TEARDOWN, confirmando a finalização da transmissão da *stream* de vídeo.
	`pause()`	Se o hóspede premir o botão *pause*, este método é invocado e envia um pedido de PAUSE ao servidor RTSP.

	`rewind()`	Quando o hóspede premir o botão de *rewind*, este método é invocado. É feito um envio de um pedido PAUSE para obter a posição actual da *stream*, decrementa-se um valor constante à posição e é enviado um pedido PLAY com a nova posição.
	`forward()`	O hóspede ao premir o botão de *forward*, este método é executado. É feito um envio de um pedido PAUSE para obter a posição actual da *stream*, incrementa-se um valor constante à posição e é enviado um pedido PLAY com a nova posição.
	`stop()`	Este método envia um pedido de TEARDOWN.
	`get_parameter()`	Este método envia um pedido de GET_PARAMETER.
	`on_get_parameter()`	Este método recebe a resposta ao pedido de GET_PARAMETER.
Outros Métodos	`check_ok()`	Este método faz a validação de todas as respostas efectuadas pelo servidor, ou seja, verifica se o pedido foi feito com sucesso ou se algum erro surgiu.
	`get_vod_info()`	Quando é necessário ir verificar a posição de algum filme ao *array* de posições no `SharedObject`.
	`set_vod_info()`	Para guardar a posição de algum filme no *array* de posições do `SharedObject`.
	`clear_vod_info()`	Sempre que for necessário limpar algum registo do *array* de posições do `SharedObject`.
	`get_range()`	Quando o cliente RTSP recebe uma resposta do servidor, se nesta conter a posição actual do filme, este método é executado para fazer o *parsing* da *string* de resposta, de forma a obter essa posição.

De forma a ser possível iniciar uma conexão, foi usado um `XMLSocket`. O objecto `XMLSocket` implementa *sockets* do cliente que permitem que a máquina que está a executar o Flash Player comunique com o servidor identificado pelo endereço IP ou pelo seu domínio. Para usar o `XMLSocket`, o servidor deve executar um *daemon* que compreenda o protocolo usado pelo objecto `XMLSocket`. O protocolo é o seguinte:

- Mensagens XML são enviadas através de uma conexão de *sockets* de fluxo TCP/IP *full-duplex*;
- Cada mensagem XML é um documento XML completo, terminado por um *byte* zero;
- Um número ilimitado de mensagens XML pode ser enviado e recebido por uma conexão `XMLSocket`.

O protocolo RTSP define um conjunto de sequências de controlo úteis para controlar a reprodução multimédia. A base da implementação deste cliente RTSP foi gerir essas sequências. De forma a contextualizar, serão apresentadas as funcionalidades de cada uma dessas sequências de controlo [25]:

- **OPTIONS**: Neste primeiro pedido, o cliente envia uma *string*, que contém o URL com o endereço IP ou o domínio do *media server*, juntamente com o *path* do ficheiro multimédia seleccionado. Na mesma *string* é enviada a versão do protocolo RTSP, o número da sequência para um par RTSP pedido-resposta (`CSeq`) que é incrementado ao longo dos pedidos, e por fim as informações sobre o agente que origina o pedido (`User-Agent`);
- **DESCRIBE**: Este método obtém a descrição de um objecto multimédia identificado pelo URL pedido a partir do servidor. Este pode usar o cabeçalho `Accept` para especificar os formatos de descrição que o cliente consegue interpretar. O servidor responde com uma descrição do conteúdo solicitado. A par pedido-resposta do DESCRIBE constitui a fase de iniciação do conteúdo RTSP. A resposta a este pedido contém todas as informações de inicialização do conteúdo multimédia para os recursos que este descreve. Nesta fase o cliente RTSP tem de guardar certas informações em relação ao conteúdo multimédia que pediu, para futuras trocas de pedidos. Através da *string* de resposta do servidor, foi possível guardar:
  - O valor gerado para a porta do cliente, de forma a esta já ficar definida;
  - A duração total do filme em segundos, para ser usado em futuras abordagens.
- **SETUP**: Este define como é que uma *stream* multimédia deve ser transportada. Isto deve ser feito antes de um pedido PLAY ser enviado. O SETUP contém o URL da *stream* multimédia e a especificação do transporte da mesma. Esta especificação geralmente inclui uma porta local para receber dados RTP (áudio e vídeo), e outra para dados RTCP (meta-informação). A resposta do servidor confirma os parâmetros enviados, e preenche as partes em falta, tais como as portas escolhidas pelo servidor. Cada *stream* multimédia deve ser configurada usando um pedido SETUP antes desta ser reproduzida. Depois de verificar a *string* recebida, é necessário o cliente guardar o identificador de sessão (`Session`). Este cabeçalho de pedido e resposta identifica uma sessão RTSP iniciada pelo servidor de multimédia e é concluído num pedido de TEARDOWN. O identificador de sessão é escolhido pelo servidor de multimédia. Uma vez que um cliente recebe um identificador de sessão, ele deve devolvê-lo em qualquer pedido relacionado com essa sessão.

No *plugin* do VOD, o cliente ao comprar um filme, pode optar por não o querer ver seguido. Este tem a possibilidade de retomar o filme na posição onde o deixou. Uma televisão pode ser usada por mais que uma pessoa, logo, pode haver mais que um filme

comprado, que ainda não tenha sido visto até ao fim. Por isso, a aplicação tem de ser capaz de guardar as posições dos filmes que o cliente tenha deixado por ver. Isso é possível fazer através do cabeçalho `range`. Para assegurar o bom funcionamento deste sistema, foram tidas em consideração três situações onde o `range` vai ser igual a zero:

- Não existir nenhum `range` guardado para o respectivo filme;
- O `range` existente ser igual ao `total_range` do filme. Se o filme já tiver sido visto até ao fim, o cliente tem de ser capaz de o ver novamente, mas para isso o `range` tem de ser reinicializado;
- O `range` existente ser maior que o `total_range` do filme. Em casos de erro isto pode acontecer, e para tal tem de ser contornado.

Continuando com a apresentação das sequências de controlo:

- **PLAY**: O método PLAY diz ao servidor para iniciar o envio de dados através do mecanismo especificado no SETUP. Um cliente não deve emitir um pedido de PLAY até que todos os pedidos pendentes de SETUP tenham sido reconhecidos como bem sucedidos. O pedido PLAY posiciona o tempo de reprodução normal para o inicio do intervalo especificado e proporciona o fluxo de dados até que o fim do intervalo é atingido ou até que este seja interrompido. A resposta do servidor a este pedido recebe alguns cabeçalhos que são relevantes durante a implementação do cliente:
  - O `timeout` indica ao cliente o tempo que o servidor está preparado para esperar entre os comandos RTSP antes de fechar a sessão por falta de actividade. O tempo de espera é medido em segundos, e por defeito são 60 segundos;
  - O `ntp` (*Normal Play Time*) indica a posição total da *stream* em relação ao início da apresentação. Não sendo especificado o início e o fim do intervalo do `npt`, o servidor vai assumir a posição zero e a posição total da *stream*, ou seja, o início e o fim do filme;
  - O `Scale` indica a velocidade de reprodução. Se for igual a 1 indica uma velocidade de reprodução normal. Nos outros casos, o valor corresponde a uma velocidade de reprodução múltipla à velocidade normal. Por exemplo, um `Scale` igual a 2 indica o dobro da velocidade normal de visualização (*fast-forward*) e um `Scale` igual a 0.5 indica metade da velocidade normal de visualização (*slow-motion*). Um valor negativo indica o sentido inverso (*fast-rewind*).

- **FORWARD** e **REWIND**: Apesar dos pedidos FORWARD e REWIND estarem previstos no protocolo RTSP, optou-se por não os incluir dado que a funcionalidade não estava contemplada nos outros produtos. Como todas as outras plataformas não utilizam RTSP, foi necessário criar um cliente RTSP com base no que já existia. Os *media server* usados pela Nonius são capazes de fazer *fast-forward* e *fast-rewind* através da utilização do cabeçalho `Scale` com valores diferentes de um. Mas, como neste caso o `Scale` será sempre igual a um, o sistema irá aplicar uma espécie de salto, adicionando ou subtraindo um valor pré-definido ao `range`, consoante se quer andar para a frente ou para trás no filme, como apresentado nas fórmulas a seguir:

  (**3**) $R_{fw} = R + k$

  (**4**) $R_{rw} = R - k$

  $R_{fw} \rightarrow Novo\ posição\ para\ um\ pedido\ fast - forward$
  $R_{rw} \rightarrow Nova\ posição\ para\ um\ pedido\ fast - rewind$
  $R \rightarrow Posição\ actual\ do\ filme\ (range)$
  $k \rightarrow Constante\ de\ tempo\ pré - definida$

  À medida que se vai premindo o botão *forward* ou *rewind*, vai-se adicionando ou subtraindo a constante de tempo. Aproveitando o facto de que o servidor envia a posição actual (*timestamp*) do filme em todos os GET_PARAMETER e PAUSE, optou-se por obter essa posição fazendo um pedido PAUSE. Através do valor da posição é feita a operação matemática de acordo com o pedido efectuado, e depois é enviado um pedido PLAY com a nova posição. Em termos visuais para o utilizador, este vai ver uma pequena paragem e um avanço ou recuo temporal no filme, e não um deslocamento contínuo das *frames* como acontece no *fast-forward* ou *fast-rewind* normal, daí ser aplicado o termo 'salto'. Se o hóspede fizer um *forward* ou um *rewind* perto do início ou no fim do filme e se a constante somada ou subtraída for maior ou menor que o alcance total do filme ou inferior à posição inicial do mesmo, o filme pára ou começa do início, de forma a evitar erros na aplicação;
- **GET_PARAMETER**: Este pedido recupera o valor de um parâmetro da *stream* especificado no URI (*Uniform Resource Identifier*). O conteúdo do pedido e resposta é deixado para a implementação. O GET_PARAMETER sem qualquer conteúdo pode ser usado para testar a vivacidade do cliente ou do servidor ("*ping*");
- **PAUSE**: Este pedido faz com que a *stream* de entrega seja interrompida (parada) temporariamente. Esta pode ser retomada enviando outro pedido PAUSE ou a partir de um pedido PLAY;
- **TEARDOWN**: Este pedido é usado para terminar a sessão, parando todos os fluxos multimédia e libertando todos os dados de sessão do servidor. Um dos objectivos que já foi referido, é que o hóspede possa retomar o filme aonde o deixou. Para isso é

necessário guardar o `range` actual quando for feito o *stop*. Mas a resposta do servidor ao pedido de TEARDOWN, não contém o cabeçalho com o valor do `range` actual. Por essa razão, foi necessário arranjar outra alternativa. Da mesma forma que foi feito para o *forward* e *rewind*, será necessário lançar um pedido PAUSE antes de enviar o TEARDOWN.

Foi com base nestes pedidos que o cliente RTSP para a plataforma NONIUS.TV | LG Pro:Centric foi implementada, sendo este o primeiro cliente do género a ser criado na empresa. De forma a interpretar melhor a funcionalidade de todos estes pedidos na aplicação aqui referida, a Figura 20 apresenta um MSC (*Message Sequence Chart*) com uma sequência desses pedidos, abrangendo todos os que aqui foram descritos.

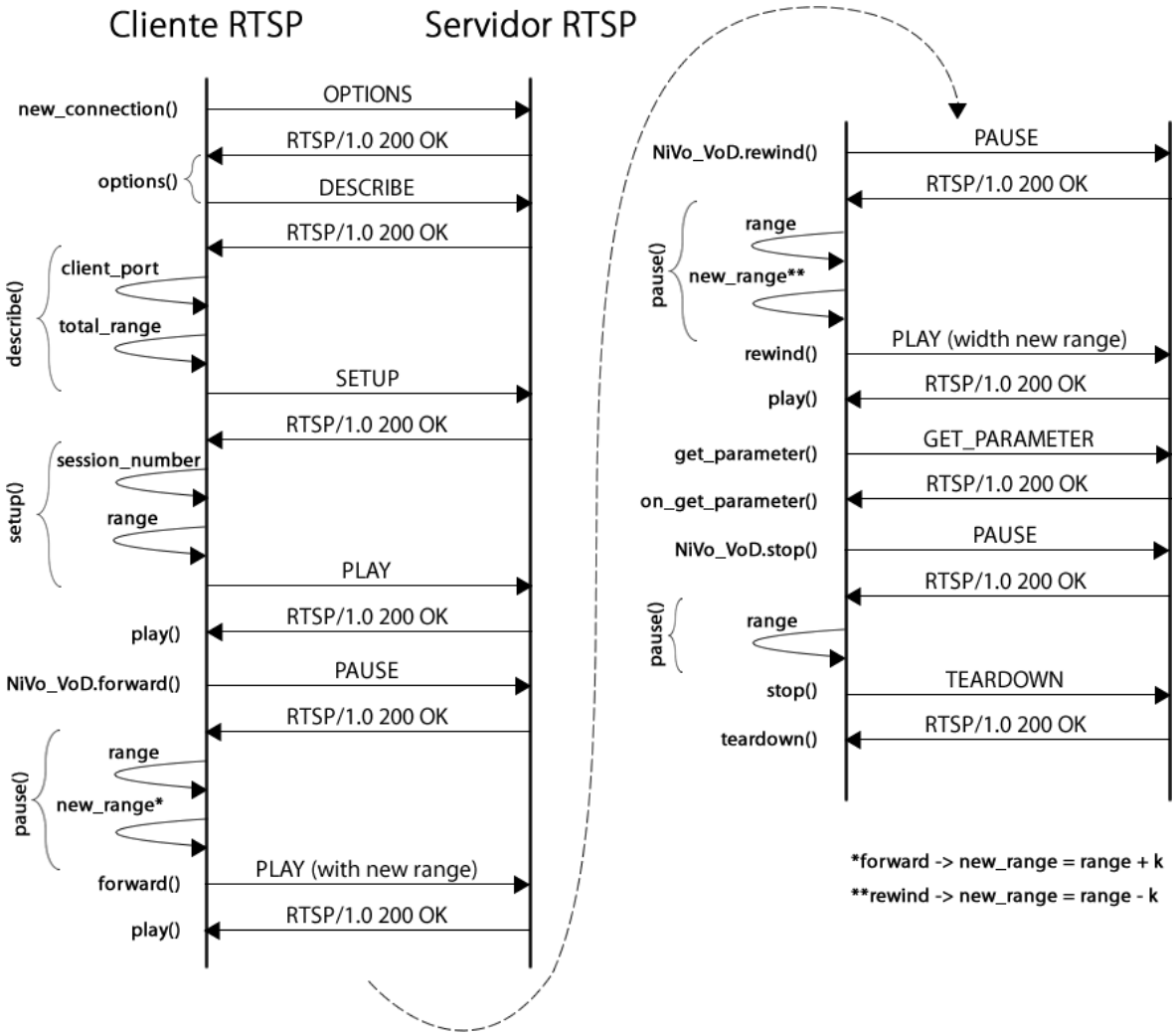


Figura 20 *Message Sequence Chart* da troca de pedidos e respostas entre o cliente e servidor RTSP usado na aplicação

5.3. CAIXA DE MENSAGENS (*INBOX*)

O *plugin Inbox* permite ao hóspede ter acesso a uma caixa de mensagens no seu quarto. É uma forma de contacto entre o hóspede e a assistência do hotel, que permite a promoção de serviços existentes no hotel, promoções, alertas de marcações de serviços, alertas de taxação de produtos, mensagem de boas vindas e outro tipo de mensagens enviadas directamente pelo *staff* do hotel a partir do *backend* ou então geradas automaticamente. A sua interface de utilização está apresentada na Figura 21.

Figura 21 Interface gráfica da caixa de mensagens

Os métodos existentes na aplicação da caixa de mensagens são apresentados na Tabela 7. Através das referências numéricas apresentadas na Figura 21, é possível ver quais os métodos que criam os respectivos componentes gráficos.

Tabela 7 Lista de métodos da classe `NiVo_INBOX` e a sua descrição

		Classe `NiVo_INBOX`	
		Método	Descrição
Criação do GUI	1	`create_header()`	Cria o cabeçalho com o título do *plugin*.
	2	`create_headerMsg()`	Cria uma barra de fundo com caixas de texto que irão conter a contagem de mensagens e os dados referentes à mensagem seleccionada (remetente, assunto e data-hora).
	3	`create_bodyMsg()`	Cria uma barra de fundo com uma caixa de texto que irá apresentar o corpo da mensagem.
	4	`create_msgsContainer()`	Cria a barra de fundo e a barra de selecção de mensagens. Este método faz também a gestão da posição de cada mensagem, invocando depois o método `create_msg()`.
		`create_buttons()`	Cria os botões presentes na lista de mensagens.

		`create_msg()`	Cria cada mensagem da lista de mensagens. Uma mensagem da lista contém três caixas de texto com o remetente, assunto e data-hora da mensagem, e contém também uma imagem que apresenta o estado da mensagem (lida / não lida). Cada um desses componentes é criado na posição enviada pelo método `create_msgsContainer()`.
	5	`create_opt()`	Cria a barra de fundo, as imagens dos botões e as respectivas caixas de texto.
Visibilidade	`show()`		Apresenta todo o GUI do *plugin* invocando todos os métodos que criam os componentes gráficos.
	`show_del()`		Apresenta só a parte da lista de mensagens.
	`hide()`		Destrói toda a interface gráfica.
	`hide_del()`		Destrói só a parte que apresenta a lista de mensagens.
Deslocamento	`down()`		Permite o deslocamento para baixo da lista de mensagens. Quando for necessário, este invoca o `slide_up()`.
	`up()`		Permite o deslocamento para cima da lista de mensagens. Quando for necessário, este invoca o `slide_down()`.
	`slide_down()` `slide_down_Y()`		Quando invocado, este método faz o deslizamento para baixo da lista de mensagens através do `slide_down_Y()`, controlando a visibilidade dentro e fora do *offset*.
	`slide_up()` `slide_up_Y()`		Quando invocado, este método faz o deslizamento para cima da lista de mensagens através do `slide_up_Y()`, controlando a visibilidade dentro e fora do *offset*.
Gestão *WebServices*	`GetMessages()`		Invocado quando se pretende obter a lista de mensagens através do *WebService* `GetMessages`.
	`DeleteMessage()`		Invocado quando se pretende apagar uma mensagem através do *WebService* `DeleteMessage`.
	`ReadMessage()`		Invocado quando se pretende colocar uma mensagem como lida através do *WebService* `ReadMessage`.
	`web_service_call_no_params()`		É o método que permite efectuar o envio de pedidos de todos os *WebServices* usados neste *plugin*, através dos respectivos métodos da classe `Soap_Requestes`.
	`soapResponseReady()`		Recebe a resposta aos pedidos feitos pelos *WebServices* utilizados por este *plugin*.
Outros Métodos	`verify_messages()`		Muda o estado de uma mensagem após esta estar seleccionada durante três segundos, invocando depois o método `ReadMessage()`.
	`parse_inbox_xml()`		Este método faz o *parsing* da lista de mensagens recebida através da resposta do *WebService* `GetMessages`.
	`deleteMessage_popUp()`		Apresenta o *pop-up* que pede a confirmação para eliminar uma mensagem.
	`alert_message_showed_state()`		Método que permite retornar para o menu principal caso não hajam mensagens em caixa.

	`init()`	Faz a inicialização de todas as classes necessárias para implementação do código do menu principal.
	`textDefine()`	Faz a alteração do conteúdo das caixas de texto que contêm o remetente, assunto, data-hora e o corpo da mensagem, à medida que se percorre a lista de mensagens.
	`getPositionFromTextField_X()`	Retorna o valor calculado da posição no eixo dos xx relativamente ao tamanho de uma caixa de texto.

A lista de mensagens para o respectivo hóspede é obtida através da chamada de um *WebService*, na classe `NiVo_Inbox`. O conceito de *WebService* é utilizado em diversas partes da aplicação. O Anexo D apresenta um exemplo de como é feito um pedido a um *WebService*. Para contextualizar, irão ser apresentados quatro *WebServices* relacionados com as mensagens de um hóspede:

- **`GetMessages`**: Obtém um XML com a lista de mensagens para o respectivo hóspede, em que as *tags* definem os constituintes de cada mensagem;
- **`ReadMessage`**: Após enviar o *id* da mensagem a colocar como lida, recebe *true* ou *false* consoante a mensagem foi registada como lida ou não com sucesso;
- **`DeleteMessage`**: Após enviar o *id* da mensagem a eliminar, recebe *true* ou *false* consoante a mensagem foi apagada ou não com sucesso;
- **`GetTotalUnreadMessages`**: Obtém o número de mensagens não lidas registadas na base de dados.

O deslizamento da lista de mensagens é feito verticalmente, mas a construção é em muito semelhante ao que foi feito no menu principal. Apesar do menu principal ter um deslizamento horizontal, a diferença aqui será na troca das coordenadas a serem manipuladas: em vez de se usarem as coordenadas de xx, usam-se as coordenadas yy. A diferença na lista de mensagens do *Inbox* está na existência de um *offset*. O *offset* é o número máximo de mensagens que podem ser apresentadas no ecrã ao mesmo tempo, num espaço definido (Figura 22).

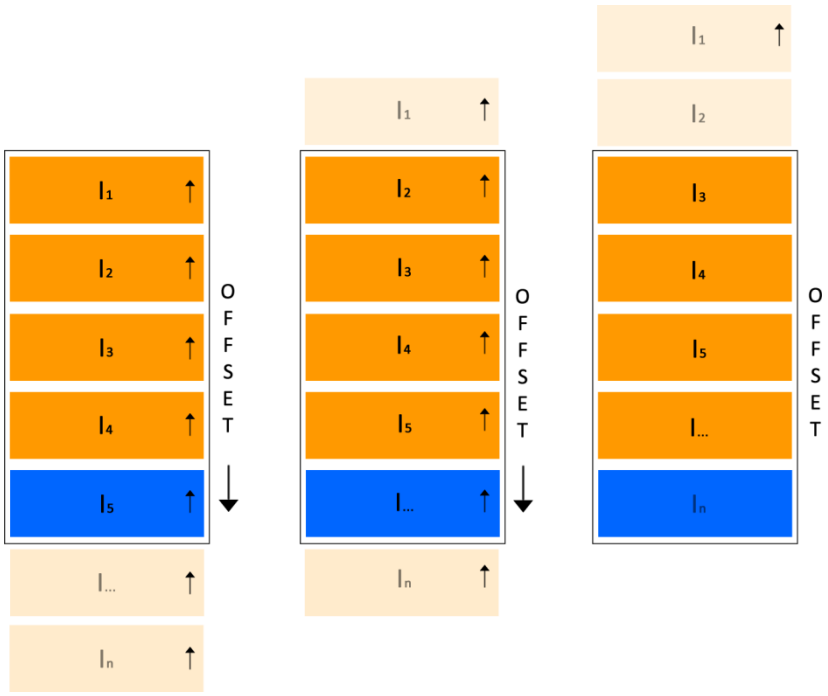


Figura 22 Esquema demonstrativo de um *offset*

No caso da Figura 22, só é possível apresentar 5 mensagens ao mesmo tempo. Ao contrário do menu principal, aqui não é possível usar todo o comprimento vertical do ecrã para apresentar as mensagens, restringindo a listagem a um pequeno espaço.

O hóspede tem também a possibilidade de eliminar mensagens. Para tal é usado o *WebService* `DeleteMessage`. Para eliminar uma mensagem, o hóspede terá de premir na tecla vermelha do comando e proceder à sua confirmação, como apresentado na Figura 23.

Figura 23 *Pop-up* de confirmação para apagar a mensagem seleccionada

Apesar de não estar directamente implementado na classe `NiVo_INBOX`, mas sim na classe `NiVo_MainMenu`, o `GetTotalUnreadMessages` faz parte do conjunto de *WebServices* criados para o controlo e gestão das mensagens do hóspede. Este quando é invocado devolve o

número de mensagens não lidas à aplicação. Esse valor é convertido numa imagem indicadora do respectivo número: 0, 1, 2, 3 ou mais que 3 (como indicado na Tabela 2 do capítulo anterior).

Juntamente com os processos lançados pelo *timer* no menu principal, alguns *WebServices* são chamados, também de minuto a minuto, como é o caso do `GetTotalUnreadMessages`. O controlo periódico do envio desses processos é feito através do *timer* criado na classe `DateTime`. Um único *timer* consegue gerir vários processos de forma eficaz, porque este não tem de lançar todos os métodos ao mesmo tempo em que está definido. Por exemplo, o processo de registo da TV e da sincronização do relógio é algo que não é necessário fazer com muita frequência.

O pedido `GetTotalUnreadMessages` também é feito sempre que hóspede volta para o menu principal, de forma a indicar de imediato se tem alguma mensagem nova. O conteúdo da resposta a esse pedido é somente um valor numérico, representativo do número mensagens não lidas. Como o *timer* está sempre a correr em *background*, é importante não permitir que o ícone indicador do número de mensagens não lidas, quando actualizado, seja apresentado num ecrã em que este não é suposto aparecer.

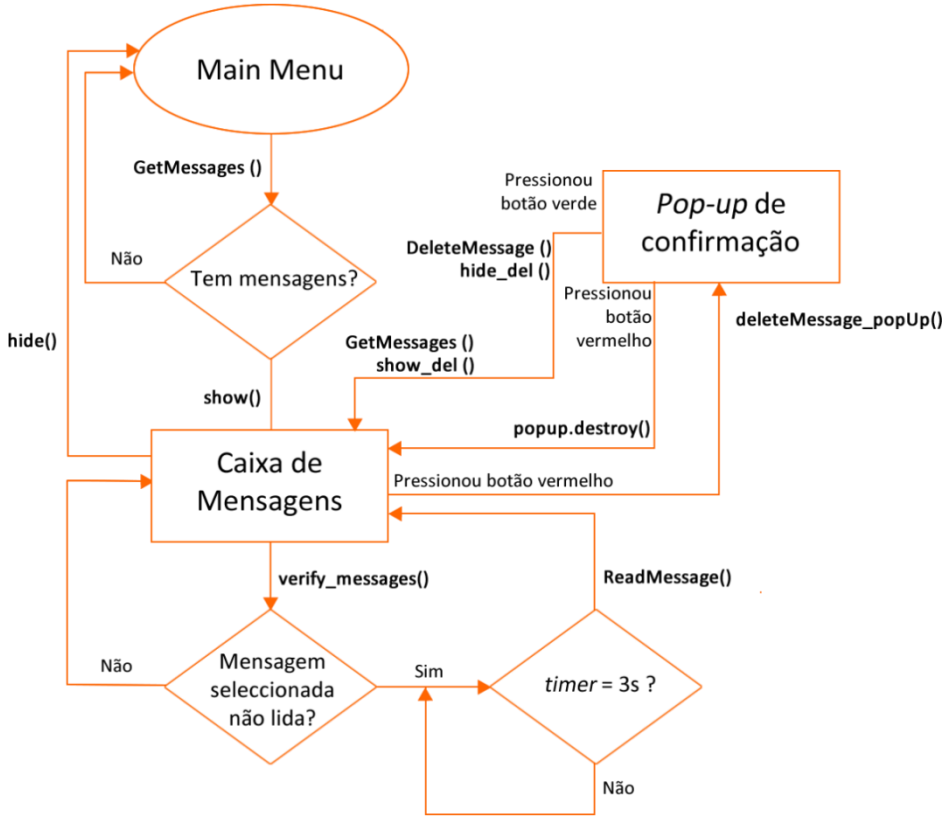


Figura 24 Fluxograma que representa o funcionamento da caixa de mensagens

5.4. DESPORTO & LAZER (*SPORTS & LEISURE*)

O *plugin* Desporto e Lazer apresenta uma lista de serviços de desporto e lazer que estão disponíveis no hotel (SPA, Golf, restaurante, aluguer de carros, etc). Cada um desses serviços tem vários pacotes diferentes associados, tentando ir de encontro com as preferências dos hóspedes. Por exemplo, no SPA pode existir vários tipos de massagens, assim como existem diferentes marcas de carros para alugar. Esses pacotes podem ser adquiridos de imediato ou por marcação.

Em termos de aplicação, o *plugin Sports & Leisure* (conhecido também como *Activities*), é dividido por vários ecrãs e cada um deles representa uma classe:

- **`NiVo_ActivitiesCategories`**: Onde estão dispostos os serviços por categorias. A Figura 25 apresenta o ecrã com o menu das categorias e a Tabela 8 lista os métodos desta classe.

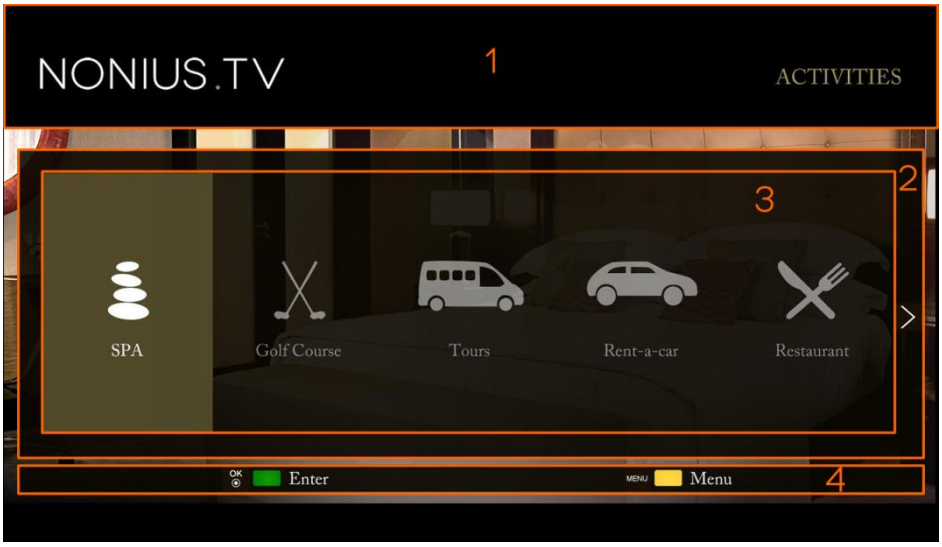


Figura 25 Interface gráfica do ecrã das categorias no *Sports & Leisure*

Tabela 8 Lista de métodos da classe `NiVo_ActivitiesCategories` e a sua descrição

	Classe `NiVo_ActivitiesCategories`		
		Método	Descrição
Criação do GUI	1	`create_header_container()`	Cria o cabeçalho com o título do *plugin*.
	2	`create_menu_container()`	Cria uma barra de fundo do menu das categorias.
	3	`create_categories_container()`	Cria a barra de selecção das categorias e faz a gestão da posição de cada categoria no menu, invocando depois o método `create_category()`.

		create_category()	Cria cada botão do menu. Um botão inclui um ícone e uma caixa de texto. Cada um desses componentes é criado na posição enviada pelo método create_categories_container().
	4	create_options_container()	Cria a barra de opções.
Visibilidade	show_actCategories()		Apresenta todo o GUI deste ecrã invocando todos os métodos que criam os componentes gráficos.
	hide()		Esconde os componentes que não vão ser usados no próximo ecrã.
	clear_all()		Destrói todo o ecrã.
Deslocamento	right()		Método que permite o deslocamento para a direita do menu de categorias. Quando for necessário, este invoca o slide_left().
	left()		Método que permite o deslocamento para a esquerda do menu de categorias. Quando for necessário, este invoca o slide_right().
	slide_right() slide_right_X()		Quando invocado, este método faz o deslizamento para a direita do menu de categorias através do slide_right_X().
	slide_left() slide_left_X()		Quando invocado, este método faz o deslizamento para a esquerda do menu de categorias através do slide_left_X().
Gestão *WebServices*	getSportsAndLeisureCategories()		Invocado quando se pretende obter a lista de categorias através do *WebService* getSportsAndLeisureCategories.
	web_service_call_no_params()		É o método que permite efectuar o envio do pedido do *WebService* getSportsAndLeisureCategories, através do respectivo método da classe Soap_Requestes.
	soapResponseReady()		Recebe a resposta ao pedido feito pelo *WebService* getSportsAndLeisureCategories.
Outros Métodos	categories_to_products_change_listener()		Método que faz a troca do *listener* usado no ecrã das categorias para o *listener* do ecrã de produtos da classe NiVo_ActivitiesProducts.
	categories_to_mainMenu_change_listener()		Método que faz a troca do *listener* usado no ecrã das categorias para o *listener* do ecrã do menu principal da classe NiVo_MainMenu.
	parse_xml_category()		Este método faz o *parsing* da lista de categorias recebida através da resposta do *WebService* getSportsAndLeisureCategories.
	setNewListener()		Cria um novo *listener* para o ecrã das categorias.
	service_unavailable()		Método que apresenta um *pop-up* a informar que o serviço está indisponível.

	`init()`	Faz a inicialização de todas as classes necessárias para implementação do código do menu principal.

- **`NiVo_ActivitiesProducts`**: Onde estão listados os pacotes de produtos de uma respectiva categoria. A Figura 26 apresenta o ecrã dos produtos e a Tabela 9 lista os métodos desta classe.

Figura 26 Interface gráfica do ecrã dos produtos no *Sports & Leisure*

Tabela 9 Lista de métodos da classe `NiVo_ActivitiesProducts` e a sua descrição

		Classe `NiVo_ActivitiesProducts`	
		Método	Descrição
Criação do GUI	1	`create_productsMenu_container()`	Cria a barra de fundo do menu dos produtos e os botões laterais.
		`create_products_container()`	Cria a barra de selecção e faz a gestão da posição de cada produto na lista, invocando depois o método `create_product()`.
		`create_product()`	Cria cada item da lista de produtos. Um item inclui uma imagem e uma caixa de texto. Cada um desses componentes é criado na posição enviada pelo método `create_products_container()`.
	2	`create_infoProduct_container()`	Cria o painel de informações do lado direito incluindo uma imagem e as caixas de texto da descrição e nome do produto.
	3	`create_opt_buttons()`	Cria uma barra de opções.
		`show_buttons()`	Cria os botões da barra de opções.
Visibilidad		`show_actProducts()`	Apresenta todo o GUI deste ecrã invocando todos os métodos que criam os componentes gráficos.
		`hide()`	Esconde os componentes que não vão ser usados no próximo ecrã.

	`hide_infoProduct()`	Esconde só o painel de informações.
	`clear_all()`	Destrói todo o ecrã.
Deslocamento	`down()`	Método que permite o deslocamento para baixo no menu dos produtos, dentro do respectivo *offset*. Quando for necessário, este invoca o `slide_up()`.
	`up()`	Método que permite o deslocamento para cima no menu dos produtos, dentro do respectivo *offset*. Quando for necessário, este invoca o `slide_down()`.
	`slide_down()` `slide_down_Y()`	Quando invocado, este método faz o deslizamento para baixo do menu dos produtos através do `slide_down_Y()`.
	`slide_up()` `slide_up_Y()`	Quando invocado, este método faz o deslizamento para cima do menu dos produtos através do `slide_up_Y()`.
Gestão *WebServices*	`getSportsAndLeisureProducts()`	Invocado quando se pretende obter a lista de produtos através do *WebService* `getSportsAndLeisureProducts`.
	`web_service_call_no_params()`	É o método que permite efectuar o envio do pedido do *WebService* `getSportsAndLeisureProducts`, através do respectivo método da classe `Soap_Requestes`.
	`soapResponseReady()`	Recebe a resposta ao pedido feito pelo *WebService* `getSportsAndLeisureProducts`.
Outros Métodos	`products_to_categories_change_listener()`	Método que faz a troca do *listener* usado no ecrã dos produtos para o *listener* do ecrã das categorias da classe `NiVo_ActivitiesCategories`.
	`setNewListener()`	Cria um novo *listener* para o ecrã dos produtos.
	`parse_xml_products()`	Este método faz o *parsing* da lista de produtos, recebida através da resposta do *WebService* `getSportsAndLeisureProducts`.
	`init()`	Faz a inicialização de todas as classes necessárias para implementação do código do menu principal.
	`textDefine()`	Faz a alteração do conteúdo das caixas de texto da descrição e nome do produto e imagem no painel de informações, à medida que se percorre a lista de produtos.

- **`NiVo_ActivitiesReservation`**: Onde é feita a reserva do produto que se pretende obter e a respectiva confirmação. A Figura 27 apresenta o ecrã da reserva, a Figura 28 apresenta o ecrã de confirmação ou não da reserva e a Tabela 10 lista os métodos desta classe.

Figura 27 Interface gráfica do ecrã da reserva do produto no *Sports & Leisure*

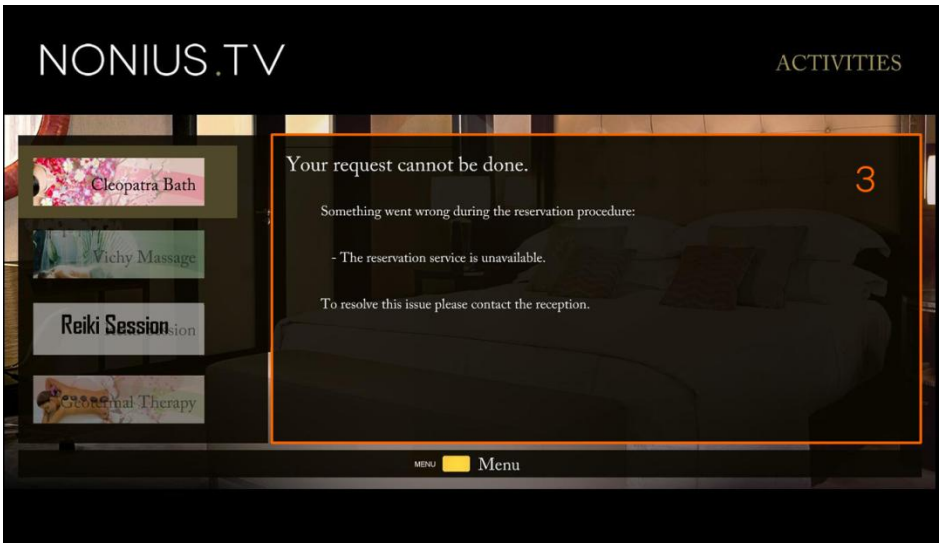


Figura 28 Interface gráfica do ecrã de uma reserva sem sucesso no *Sports & Leisure*

Tabela 10 Lista de métodos da classe `NiVo_ActivitiesReservation` e a sua descrição

	Classe `NiVo_ActivitiesReservation`		
		Método	Descrição
Visibilidade e Criação	1	`show_actReservation()`	Cria e apresenta todas as caixas de texto necessárias para a reserva, em cima do painel de informações já existente, deixada pelo ecrã dos produtos.
	2	`show_buttons()`	Cria os botões da barra de opções.
	3	`show_confirmation()`	Cria e apresenta todas as caixas de texto necessárias para a confirmação da reserva, em cima do painel de informações já existente, deixada pelo ecrã dos

		produtos.
	`hide()`	Esconde os componentes que não vão ser usados no próximo ecrã.
	`clear_all()`	Destrói todo o ecrã.
Alterar Posição/Conteúdo	`right()`	Método que permite alterar a selecção do campo actualmente seleccionado, passando para o que está ao seu lado direito.
	`left()`	Método que permite a selecção do campo actualmente seleccionado, passando para o que está ao seu lado esquerdo.
	`up()`	Método que permite incrementar o conteúdo do campo seleccionado.
	`down()`	Método que permite decrementar o conteúdo do campo seleccionado.
Gestão *WebServices*	`setSportsAndLeisureOrders()`	Invocado quando se pretende enviar o pedido de reserva através do *WebService* `setSportsAndLeisureOrders`.
	`web_service_call_no_params()`	É o método que permite efectuar o envio do *WebService* `setSportsAndLeisureOrders`, através do respectivo método da classe `Soap_Requestes`.
	`soapResponseReady()`	Recebe a resposta do pedido feito pelo *WebService* `setSportsAndLeisureOrders`.
Outros Métodos	`reservation_to_products_change_listener()`	Método que faz a troca do *listener* usado no ecrã da reserva para o *listener* do ecrã dos produtos da classe `NiVo_ActivitiesProducts`.
	`setNewListener()`	Cria um novo *listener* para o ecrã das reservas.
	`init()`	Faz a inicialização de todas as classes necessárias para implementação do código do menu principal.

Neste *plugin* foi necessário num dos ecrãs inserir valores (ecrã de reserva). Como se pode observar pela Figura 27, muitos dos componentes gráficos foram reaproveitados do ecrã anterior. A barra lateral esquerda, com a lista de produtos permaneceu de forma a dar a entender ao hóspede o produto que está a comprar. Por esta ser de uso informativo neste ecrã, a lista está bloqueada, ou seja, não desliza. Isso pode-se observar pelas opções disponíveis na barra de opções e pela ausência das setas na lista.

O ecrã das reservas apresenta também campos informativos e campos de inserção de valores. Para fazer a reserva, o hóspede tem de indicar a data em que quer usufruir do serviço, a hora e a quantidade (número de pessoas para fazer o mesmo serviço). Para definir a data e a hora, o hóspede tem de percorrer os campos com as teclas 'direita' e 'esquerda' (dia, mês, ano, hora, minutos e quantidade).

O hóspede consegue distinguir os campos editáveis dos não editáveis, assim como o campo seleccionado e os não seleccionados, através da forma e da cor como são apresentados. De forma a tornar a interface intuitiva, o hóspede pode percorrer os campos e incrementar o seu valor, utilizando as teclas de navegação existentes no comando. Esses campos contêm um conjunto de validações consoante o valor inserido, de forma a proibir valores de horas, minutos e datas inválidos. O limite das quantidades dos produtos é imposto pelo valor que vem num dos campos presentes no objecto do produto, que é a quantidade máxima de serviços disponíveis por dia.

Neste ecrã, também é apresentada alguma informação sobre o produto que se está a comprar. Existem serviços que têm um horário diário predefinido e é essencial apresentar essa informação ao hóspede no acto da compra. Futuramente pode ser implementado um sistema de validação de horas de marcação ou então uma espécie de calendário a informar as horas que ainda estão vagas.

Em todos os ecrãs presentes neste *plugin*, o hóspede pode navegar entre eles, voltando para trás sempre que precisar. Quando confirmar a compra do serviço, dois cenários podem acontecer: a reserva pode ser efectuada com sucesso e posteriormente o hóspede será contactado pela recepção com a confirmação da mesma, ou então pode ser apresentada uma mensagem de erro a dizer que a reserva não foi efectuada e as possíveis razões de isso ter acontecido, como apresentado na Figura 28.

A Figura 29 apresenta um fluxograma do funcionamento geral deste *plugin*.

Figura 29 Fluxograma que representa o funcionamento do *Sports & Leisure*

5.5. SERVIÇO DE QUARTOS (*ROOM SERVICE*)

O *plugin* do Serviço de Quartos é muito semelhante ao do Desporto & Lazer. Também funciona por categorias e produtos, mas neste é possível comprar vários produtos de uma só vez. O hóspede pode navegar pelas várias categorias e ir armazenando em memória o que pretende, como se fosse um carrinho de compras frequentemente usado nos *sites* de compras *online*.

Não serão apresentados aqui os métodos dos ecrãs de categorias e produtos já que são muito semelhantes ao das classes `NiVo_ActivitiesCategories` e `Nivo_ActivitiesProducts`, mas será feita uma pequena introdução visto usarem diferentes *WebServices*.

Neste *plugin* existe um novo ecrã que lista os produtos seleccionados até ao momento pelo cliente. Nesse ecrã é possível efectuar as seguintes operações:

- Cancelar a compra actual;
- Adicionar um novo produto à lista actual de compras;
- Eliminar um produto da lista dos produtos seleccionados para compra;
- Confirmar as compras da lista actual, enviando ao servidor a sua encomenda e se tiver sucesso é finalizado o processo de compra.

O hóspede ao cancelar a compra, limpa toda a lista de produtos seleccionados. Quando pretender adicionar um novo produto ao carrinho de compras a aplicação retorna ao menu das categorias de forma a recomeçar o processo da escolha de um novo produto.

A Tabela 11 apresenta uma lista dos métodos criados para este novo ecrã dos produtos seleccionados.

Tabela 11 Lista de métodos necessários para a criação do ecrã dos produtos seleccionados, que foram implementados na classe `NiVo_RoomServiceProducts`

Classe NiVo_ActivitiesReservation	
Método	Descrição
`add_reservation()`	Adiciona uma nova reserva ao *array* de produtos seleccionados (carrinho de compras) existente na memória do *plugin*.
`show_selectedProducts()`	Apresenta todos os componentes gráficos criados para o ecrã de produtos seleccionados.
`create_reservation_table()`	Cria a tabela com todos os produtos existentes no *array* de reservas.
`destroy_reservation_table()`	Destrói a tabela dos produtos seleccionados.
`delete_reservation()`	Apaga um registo do *array* dos produtos seleccionados.
`hide_selectedProducts()`	Esconde o ecrã dos produtos seleccionados.
`down_selProd()`	Método que permite o deslocamento para baixo na tabela dos produtos seleccionados, dentro do respectivo *offset*. Quando for necessário, este invoca o `slide_reservation_up()`.
`up_selProd()`	Método que permite o deslocamento para cima n tabela de produtos seleccionados, dentro do respectivo *offset*. Quando for necessário, este invoca o `slide_reservation_down()`.
`slide_reservation_down()` `slide_down_Y()`	Quando invocado, este método faz o deslizamento para baixo da tabela de produtos seleccionados através do `slide_down_Y()`.
`slide_reservation_up()` `slide_up_Y()`	Quando invocado, este método faz o deslizamento para cima da tabela de produtos seleccionados através do `slide_up_Y()`.

A Figura 30 apresenta um fluxograma com a sequência de todos os *Webservices* utilizados e os respectivos ecrãs que são apresentados.

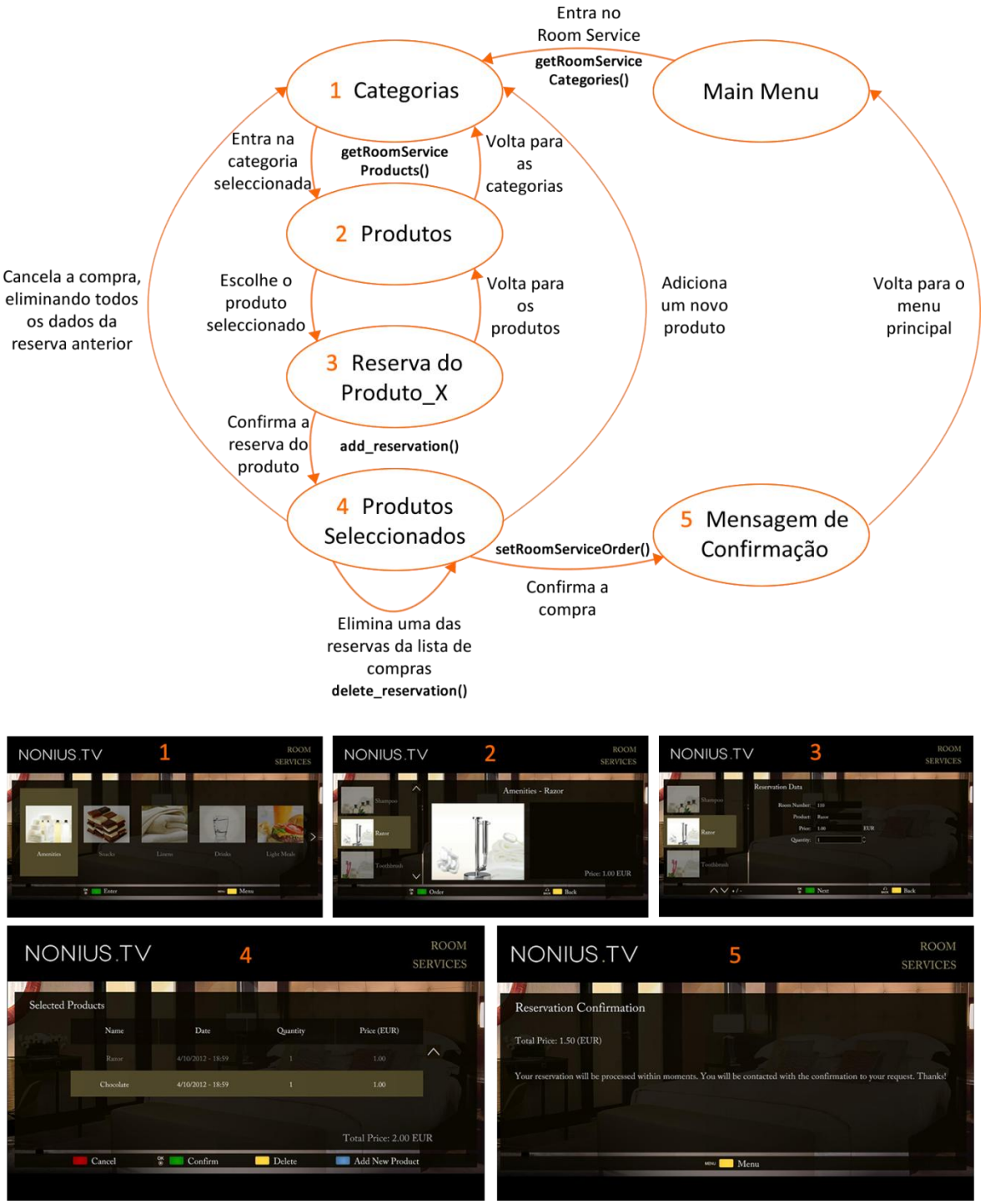


Figura 30 Fluxograma que representa a troca de pedidos dos diferentes *WebServices* usados no *Room Service*

5.6. DESPERTADOR (*WAKE UP CALL*)

O *Wake Up Call* é um *plugin* que permite ao hóspede definir uma hora específica para a televisão tocar, como uma espécie de despertador, mesmo esta estando desligada. A Figura 31 apresenta o ecrã inicial do *plugin* e a Tabela 12 lista os métodos criados.

Figura 31 Interface gráfica do ecrã do despertador por defeito

Tabela 12 Lista de métodos da classe `NiVo_WakeUp` e a sua descrição

		Classe `NiVo_WakeUp`	
		Método	Descrição
Outros		`init()`	Faz a inicialização de todas as classes necessárias para implementação do código do despertador.
		`start()`	Inicia o *plugin*.
		`show()`	Apresenta o ecrã do *plugin*. Por defeito apresenta o ecrã de inserção da hora.
		`showLoadingMessage()`	Apresenta a mensagem de *loading*.
		`update_time()`	Actualiza a hora do relógio.
Definir Layout	1	`setAppNameHeader()` `setNivoTitle()`	Criam o cabeçalho do *plugin*.
	2	`setCenterBar()` `setCenterText()`	Criam a barra central e definem o texto para descrever o *plugin*.
		`setCenterClock()`	Define o *layout* que é composto pela imagem do relógio e o *background* dos dígitos.
		`setCenterDigits()`	Define o texto do relógio. Por defeito usa a hora actual ou a hora seleccionada.
		`setCenterDigitClock()`	Define o *layout* dos dígitos do relógio.
	3	`setBottomBar()`	Cria a barra de baixo para apresentar os botões.
		`setBottomDirectional Buttons()`	Cria os botões direccionais na barra de baixo.
		`setBottomActionButton()`	Cria os botões de acção na barra de baixo.

Alterar Layout	`showNewCenter()`	Apresenta um novo *layout* central, que irá alterar o texto central e a cor dos dígitos.
	`showBottomActionButton()`	Apresenta um novo *layout* nos botões de acção, alterando o texto e a imagem do botão.
	`switchLayoutFor()`	Alternar entre o *layout* de confirmação e o *layout* de eliminação.
	`activateRingClock()`	Altera a cor da imagem do relógio da parte central.
	`showBottomDirectional Buttons()`	Mostra os botões direccionais.
Apag. e Esconder layout	`hideBottomDirectional Buttons()`	Esconde os botões direccionais.
	`destroy()`	Destrói o *plugin* e volta para o menu principal.
	`deleteBottomLayout()`	Elimina todos os botões.
	`deleteCenterLayout()`	Elimina toda a parte central do ecrã do *plugin*.
	`deleteHeaderLayout()` `deleteTitleLayout()`	Eliminam o cabeçalho do *plugin*.
Listener	`setOldListener()`	Repõe o *listener* anterior.
	`getCurrentListener()`	Obtém o *listener* actual.
	`setNewListener()`	Cria um novo *listener*.
Acções do Listener	`selectDigitLeft()`	Selecciona os dígitos do lado esquerdo do relógio.
	`selectDigitRight()`	Selecciona os dígitos do lado direito do relógio.
	`selectDigitUp()` `incrementDigits()`	Selecciona o próximo dígito, incrementando e validando o número.
	`selectDigitDown()` `decrementDigits()`	Selecciona o dígito anterior, decrementando e validando o número.
WebService	`soapResponseReady()`	Recebe a resposta SOAP.
	`setWakeUpTime()`	Define a hora do despertador via *WebService*.
	`deleteWakeUpTime()`	Apaga a hora do despertador via *WebService*.
	`getWakeUpTime()`	Obtém a hora do despertador.
	`validateSetWakeUpTime()`	Valida o resultado do *WebService* `setWakeUpTime()` e altera o *layout* do *plugin*.
	`defineWakeUpTime()`	Define a hora do despertador sem depender do *WebService*.
	`validateDeleteWakeUpTime()`	Valida o resultado do *WebService* `deleteWakeUpTime()` e altera o *layout* do *plugin*.
	`eraseWakeUpTime()`	Apaga a hora do despertador sem depender do *WebService*.
	`existWakeUpTime()`	Verifica se existe ou não uma hora de despertar definida.

Mensag. de Alerta	`existWakeUpError()`	Verifica se existe alguma mensagem de alerta.
	`activeAlertMessage()`	Define a *flag* da mensagem de alerta para *true*.
	`showAlertMessage()`	Apresenta a mensagem de alerta.
	`alert_message_showed_state()`	Desactiva a mensagem de alerta. É chamada por ela própria quando é destruída.
Tecl. Enter	`keyEnterStatus()`	Obtém o estado da tecla *Enter*.
	`keyEnterActivated()`	Define o limite da tecla *Enter* a *true*.
	`keyEnterDeactivated()`	Define o limite da tecla *Enter* a *false*.

Foi já referido várias vezes ao logo deste capítulo, o método `setNewListener()`. De forma a ter uma melhor percepção de como é feita a criação de um *listener*, o Anexo E apresenta um exemplo implementado numa das classes.

Ao definir uma hora no despertador, o despertador passa a estar activo e o *layout* do ecrã vai alterar, como apresentado na Figura 32.

Figura 32 Interface gráfica do ecrã do despertador após definir uma hora

Como o *plugin* só permite ter uma hora definida de cada vez, enquanto a hora guardada não for apagada, o ecrã da Figura 32 será sempre o apresentado.

Um problema que surgiu durante a construção deste *plugin* foi o como conseguir fazer a televisão despertar enquanto esta estivesse desligada. Isso foi contornado graças a um método disponível na API da televisão: `set_power_on_time()`, tendo como argumentos a hora e os minutos que a televisão tem que ligar. Para a televisão saber quando é que tem que iniciar o despertador, a aplicação terá de verificar de minuto em minuto, comparando a

hora actual com a hora definida para o despertador tocar. Se a hora actual for igual à hora definida para o despertador, então o *pop-up* do despertador surge no ecrã através da classe `Wake_Up_Ring` (Figura 33) e começa a emitir um sinal sonoro.

Figura 33 *Pop-up* do despertador enquanto está a tocar

Todos os processos abordados até aqui, podem ser explicados de forma mais resumida através da análise do fluxograma da Figura 34.

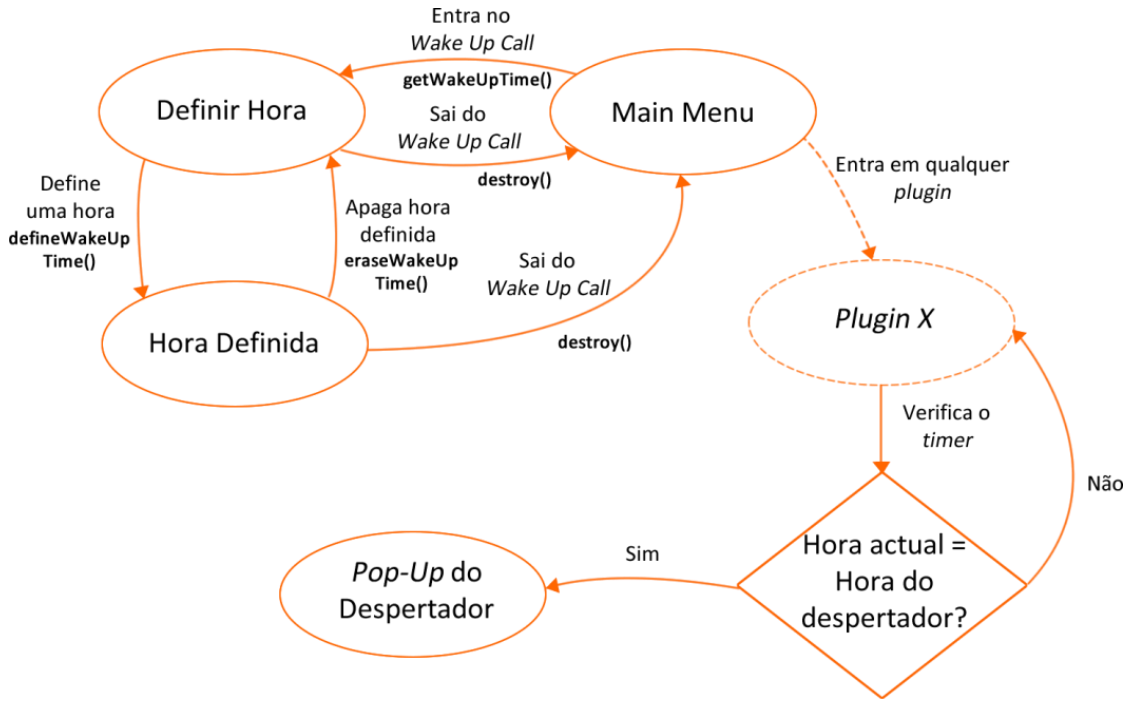


Figura 34 Fluxograma que representa o funcionamento do *Wake Up Call*

5.7. MUDANÇA DE IDIOMA (*CHANGE LANGUAGE*)

Como o próprio nome diz, este *plugin* serve para mudar a linguagem a ser usada pela aplicação. Desta forma, todo o texto contido nos menus e nas caixas de texto é actualizado para o idioma seleccionado. A Figura 35 apresenta o único ecrã deste *plugin* e a Tabela 13 lista os métodos da classe.

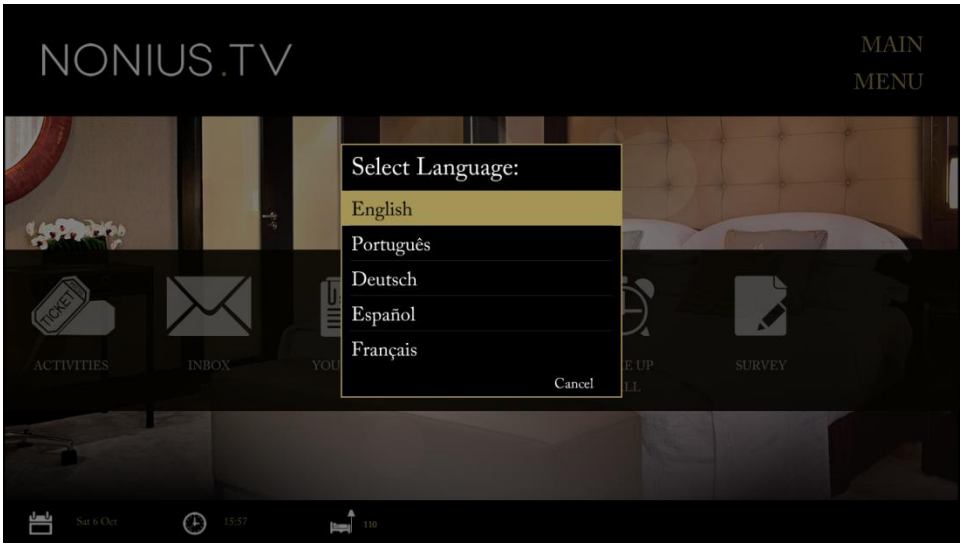


Figura 35 Interface gráfica do *plugin* de mudança de idioma

Tabela 13 Lista de métodos da classe `NiVo_WakeUp` e a sua descrição

Classe `NiVo_WakeUp`	
Método	Descrição
`init()`	Faz a inicialização de todas as classes necessárias para implementação do código da mudança de idioma.
`create_language_pop_up()`	Cria o *pop-up*, com o título e o botão de cancelar.
`create_languages()`	Cria a barra de selecção e faz a gestão da posição de cada idioma na lista, invocando depois o método `create_language()`.
`create_language()`	Cria cada item da lista de idiomas. Um item inclui uma linha e uma caixa de texto. Cada um desses componentes é criado na posição enviada pelo método `create_languages()`.
`show()`	Apresenta todo o GUI deste ecrã invocando todos os métodos que criam os componentes gráficos, obtendo também o *array* de línguas existentes na aplicação, através da classe `Translator`.

`hide()`	Esconde e destrói todos os componentes gráficos.
`down()`	Percorre a lista de idiomas para baixo.
`up()`	Percorre a lista de idiomas para cima.
`change_default_language()`	Altera o idioma por defeito de toda a aplicação, através da classe `Translator`.

A troca de idiomas é implementada tendo como base informação armazenada em ficheiros XML acessíveis pela aplicação, que contêm todas as *strings* usadas na mesma.

Nas plataformas da NONIUS.TV, é possível definir diferentes *playlists* para diferentes línguas. Como existem muitos canais nos mais diversos idiomas, é possível agrupar os canais por língua, criando diferentes *playlists*. Desta forma, quando o hóspede altera a língua do sistema, altera também as *playlists* do rádio, da televisão e do VOD.

O processo de funcionamento deste *plugin* é apresentado no fluxograma da Figura 36.

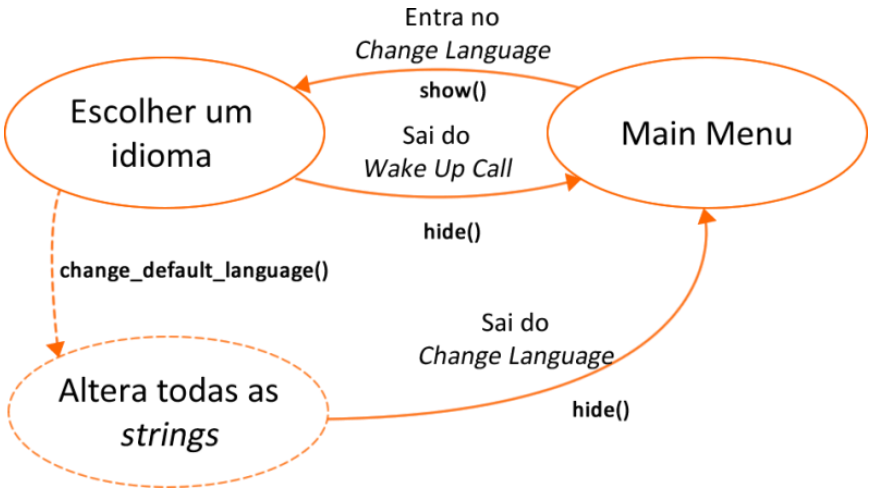


Figura 36 Fluxograma do funcionamento do *Change Language*

5.8. SERVIÇOS DE INFORMAÇÃO (*INFORMATION SERVICES*)

Este *plugin* contém um submenu (Figura 37) com alguns serviços informativos úteis para o hóspede. É possível ver os horários de partida e chegada dos voos em determinados aeroportos, consultar a informação metrológica até cinco dias para certas cidades, aceder a um *feed* de notícias da actualidade, ver as farmácias existentes num raio de alguns quilómetros definidos pelo hotel e ver um mapa das redondezas.

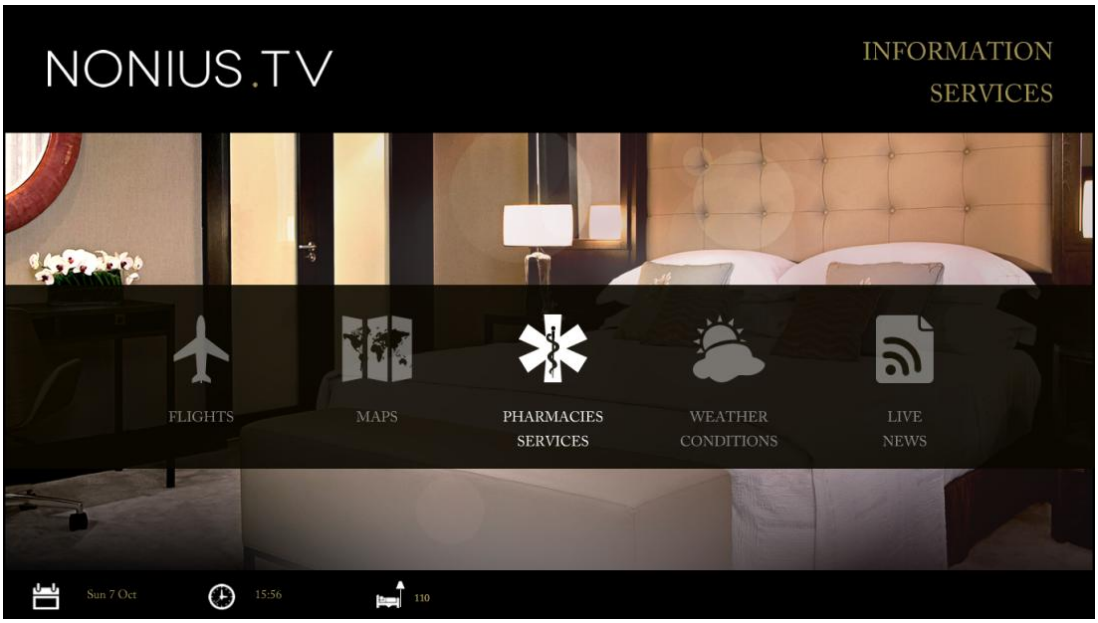


Figura 37 Menu dos Serviços de Informação

Neste subcapítulo só serão apresentados os serviços implementados durante o estágio: mapa, serviço de farmácias, condições meteorológicas e notícias.

5.8.1. MAPAS (*MAPS*)

Este ecrã faz a apresentação de um mapa que, por enquanto, será uma imagem estática (Figura 38). Isto é feito criando um objecto `MovieClip` para o efeito.

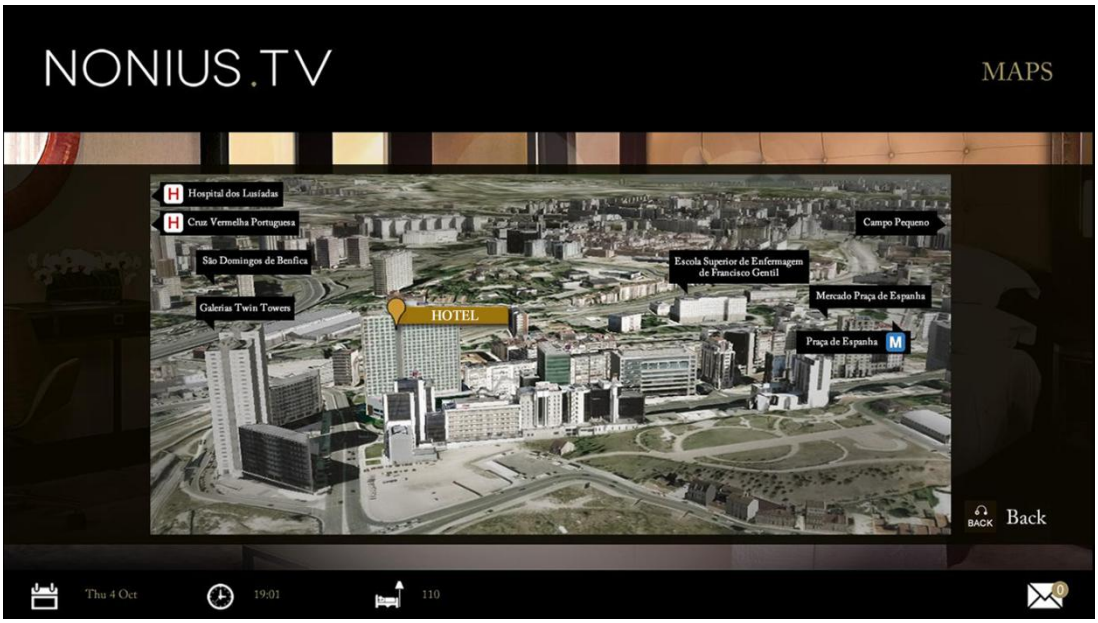


Figura 38 Ecrã do serviço de mapas

5.8.2. SERVIÇO DE FARMÁCIAS (*PHARMACIES SERVICES*)

Ao entrar no serviço de farmácias, a aplicação começa por trocar o *listener* e carrega os dados das farmácias. Isto é feito a partir da localização externa do ficheiro XML, que fica no *backend*. O *backend* obtém esta informação a partir da integração da plataforma do Sapo Farmácias. Através do código postal ou da cidade inserida, e incluindo um raio de quilómetros máximo, a plataforma retorna os dados filtrados em formato XML. Dos quais a aplicação faz o *parsing* dos dados.

A Tabela 14 apresenta os métodos criados para este serviço. As classes `pharmacies` e `pharmaciesItem` fazem o *parsing* do XML e dispõem os dados em objectos para estarem acessíveis na classe `NiVo_PHARMACIES`. Para servir de exemplo aos outros serviços, o código implementado nessas classes está disponível para consulta no Anexo F.

Tabela 14 Alguns dos métodos da classe `NiVo_PHARMACIES` e a sua descrição

Classe `NiVo_PHARMACIES`	
Método	Descrição
`create_header_container()`	Cria o cabeçalho do ecrã.
`create_view()`	Método que faz a construção de todas as farmácias ao longo das páginas, através do método `create_pharmacies_container()`.
`create_pharmacies_container()`	Cria os conteúdos para cada farmácia, através das posições enviadas pelo método `create_view()`.
`create_pharmacies_info_container()`	Cria a barra de opções.
`configure_pharmacy_containers_position()`	A aplicação a cada mudança de página terá de ir destruindo a página anterior e criar a nova. Por essa razão foi criado este método que define um índice inicial e um índice final, fazendo referencia à primeira e última farmácia de cada página.
`translate_images();`	Através do nome das imagens que vem no XML, este método traduz para o nome das imagens que existem na aplicação.
`verify_text()`	Verifica se existe algum campo no XML não esteja definido, de forma a não aparecer *'undefined'* na interface.
`update_total_number_of_pages()`	Método que faz o cálculo do número de páginas que têm de geradas na apresentação.
`up()`	Percorre as páginas anteriores.
`down()`	Percorre as páginas seguintes.

Cada farmácia irá apresentar o nome, a morada, o telefone, as coordenadas, a distância a que fica do hotel e um ícone que representa o tipo de farmácia. A barra de opções mostra ao hóspede o significado de cada um desses ícones: Farmácia 24h, Parafarmácia, Farmácia aberta até às 19h ou 22h.

A Figura 39 apresenta um exemplo de um ecrã de acesso a este serviço.

Página 1

Página 2

Figura 39 Ecrãs do serviço de farmácias, para a página 1 e 2

5.8.3. SERVIÇO METEOROLÓGICO (*WEATHER CONDITIONS*)

O serviço de meteorologia começa também por obter os dados meteorológicos do ficheiro XML. Este ficheiro é obtido através do *backend*, que o actualiza automaticamente em intervalos de tempo regulares. Os dados são obtidos usando a API do Google Weather. Na

interface do *backend*, o hotel define as cidades que devem ser apresentadas ao hóspede, e com informação meteorológica obtida para essas cidades é gerado um ficheiro XML.

Após o *parsing* estar concluído, é apresentado o ecrã da Figura 40.

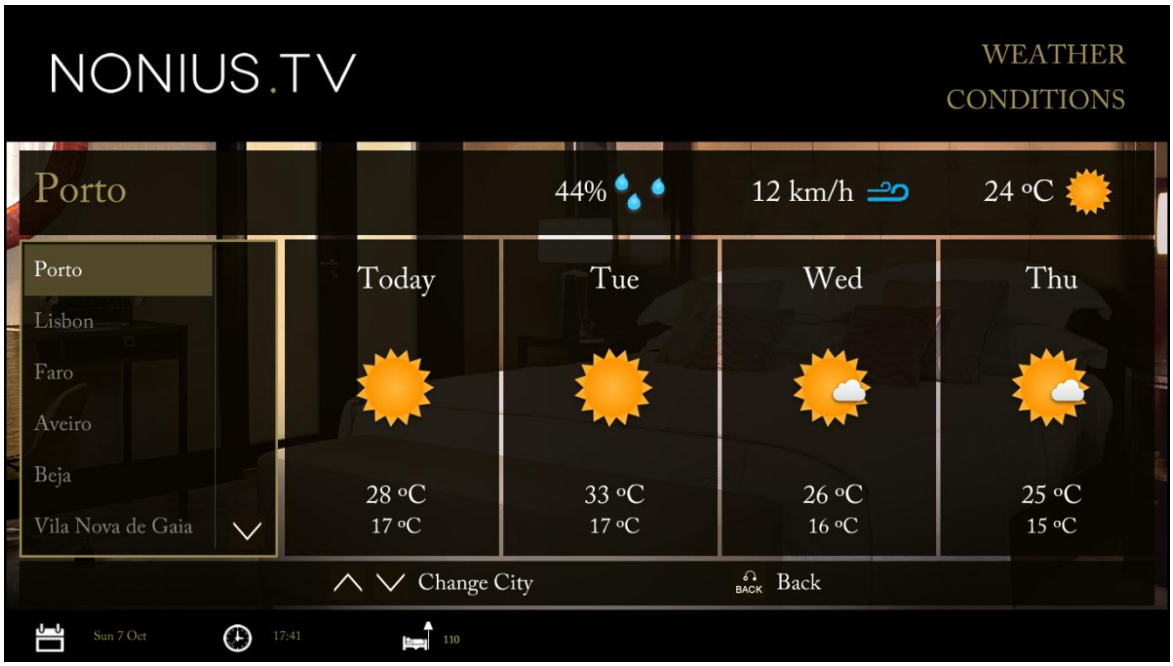


Figura 40 Ecrã do Serviço Meteorológico

A implementação feita para este ecrã é em tudo semelhante à do *plugin* do *Inbox*. A lista de cidades irá funcionar como a lista de mensagens do *Inbox*. À medida que o hóspede vai subindo ou descendo na lista, as temperaturas e as imagens vão mudar, consoante a cidade que está seleccionada, sendo por isso a lógica de implementação do código muito semelhante.

5.8.4. *FEED* DE NOTÍCIAS (*LIVE NEWS*)

Dado existirem notícias para os vários idiomas, deverá existir um ficheiro XML para cada um desses idiomas. Desta forma, é necessário fazer o *parsing* vários ficheiros. As notícias são obtidas através da plataforma do Google News.

Após todos os dados estarem carregados, a aplicação irá apresentar as notícias no idioma definido no sistema, como apresentado na Figura 41.

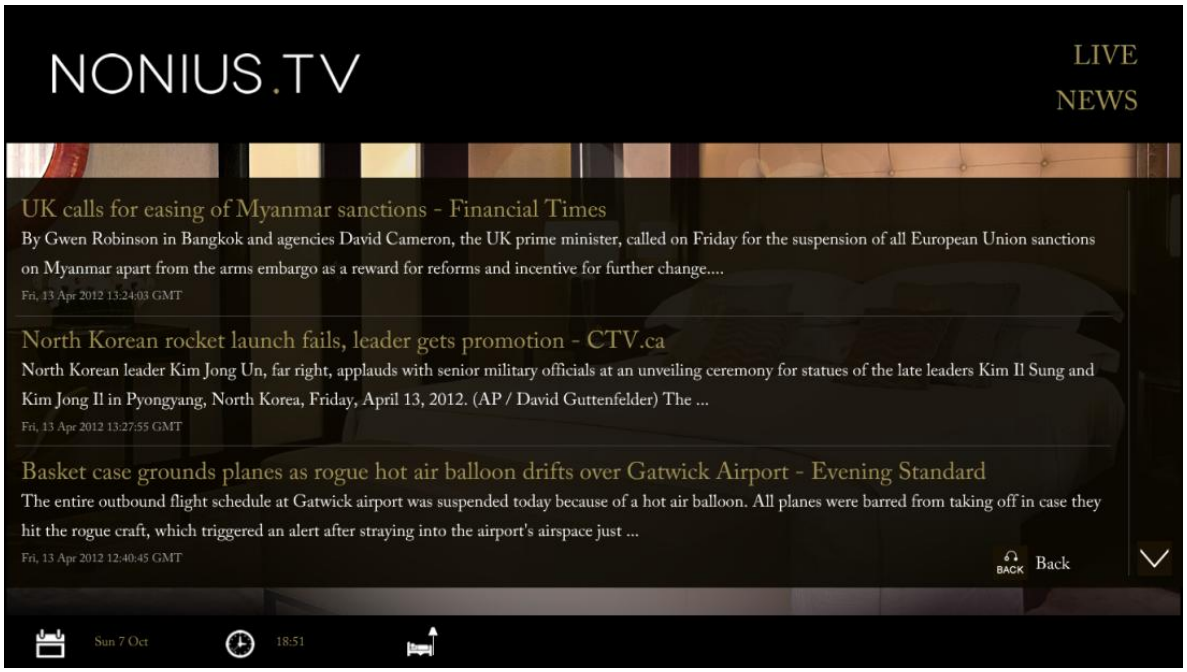


Figura 41 Ecrã do *Feed* de Notícias

A implementação deste serviço, foi em muito semelhante ao das farmácias. O método de paginação foi igual, só o número de itens por página é diferente.

O fluxograma da Figura 42 apresenta de forma resumida o processo de funcionamento dos serviços de informação aqui abordados.

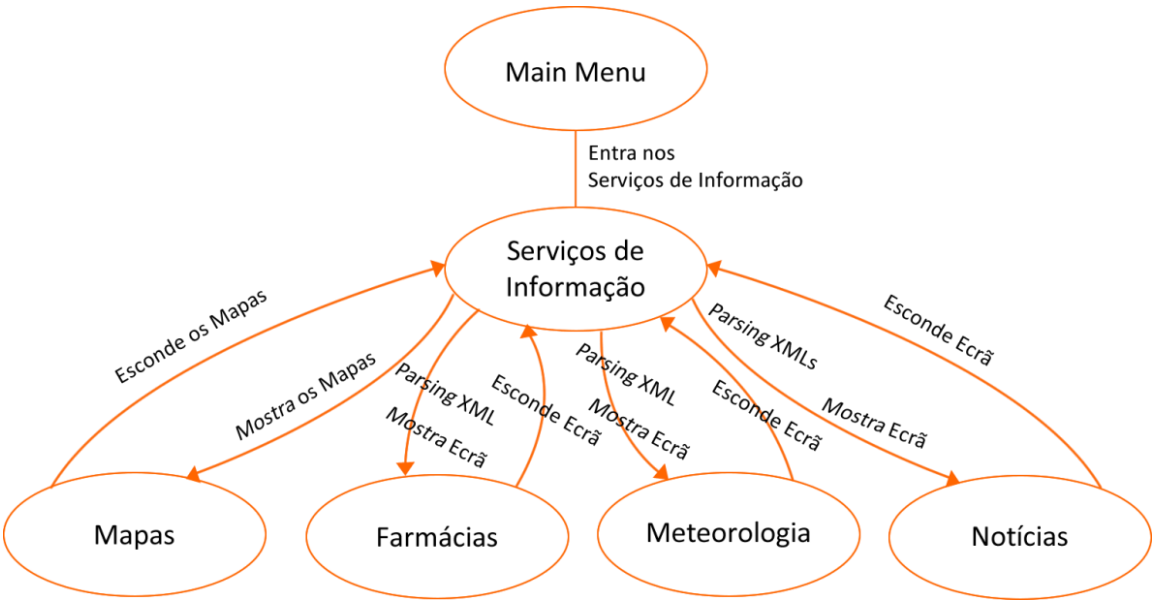


Figura 42 Fluxograma do funcionamento dos Serviços de Informação

5.9. TAXAÇÃO, ÁUDIO E CONTROLO PARENTAL NO VOD

O VOD é um dos *plugins* mais complexos dos sistemas da NONIUS.TV. Este inclui diversas funcionalidades que vão desde a confirmação de compra, taxação dos conteúdos, mudança de áudio num filme, mudança de legendas, controlo parental, entre outros. Neste subcapítulo serão abordados apenas aqueles que foram implementados durante o estágio: taxação de conteúdos, mudança de áudio/idioma e controlo parental.

A Tabela 15 irá apresentar os métodos implementados na classe `NiVo_VOD` para estas três funcionalidades:

Tabela 15 Métodos utilizados para a taxação, áudio e controlo parental na classe `NiVo_VOD`

	Classe `NiVo_VOD`	
	Método	Descrição
Taxação de Conteúdos	`vod_payment()`	Confirma se o filme seleccionado já foi comprado anteriormente pelo hóspede. Constrói uma *string*, mais precisamente um endereço URL, com os parâmetros que são necessários enviar para fazer essa validação: `id` do filme, o endereço MAC da TV e a categoria em que está inserido o filme.
	`vod_confirmation_code()`	Faz a validação do código inserido pelo hóspede: se este foi correctamente inserido, faz a taxação do filme, caso contrário um novo código é gerado e terá de inserir novamente.
Áudio	`change_audio()`	Efectua todo o processo de mudança de áudio numa *stream* de vídeo.
Controlo Parental	`check_state()`	Quando o `GetRoomInfo` envia uma resposta, esta é interpretada por este método. Este método verifica o estado da propriedade no `SharedObject` e se for diferente do estado que vem no campo `tv_rights` do *WebService*, a propriedade é actualizada.
	`adult_content()`	Este método activa ou desactiva o controlo parental consoante o valor da propriedade `parental_control` no `SharedObject`. Se for `"full_access"`, o controlo parental vai ficar desactivo. Caso contrário, fica activo. Por fim é invocado o método `show()` para construir o GUI do *plugin*.
	`create_genre()`	Este método cria a barra de géneros consoante o estado do controlo parental. A partir do nome da categoria, é possível fazer o bloqueio da mesma, tornando a categoria visível mas inacessível, caso o controlo parental esteja activo.

5.9.1. TAXAÇÃO DE CONTEÚDOS

Para o hóspede poder usufruir dos conteúdos disponíveis no VOD, é necessário proceder à sua compra. Quando é feita a confirmação da compra, a aplicação começa por confirmar se o filme já foi comprado por este hóspede anteriormente. Essa validação é feita através da chamada a uma página *web* existente no *backend*, juntamente com alguns argumentos necessários.

Ao receber o pedido, o *backend* executa a página e, consoante os argumentos recebidos, este envia uma das quatro respostas possíveis:

- `OK;`
- `NO_PIN;`
- `ERROR;`
- `[Código PIN aleatório].`

Se o filme que hóspede está a tentar ver nunca tiver sido comprado, então a aplicação recebe um código aleatório gerado pelo *backend* e apresenta um *pop-up* onde tem que inserir um código gerado aleatoriamente (Figura 43), de forma a evitar que sejam comprados conteúdos com o comando, sem intenção.

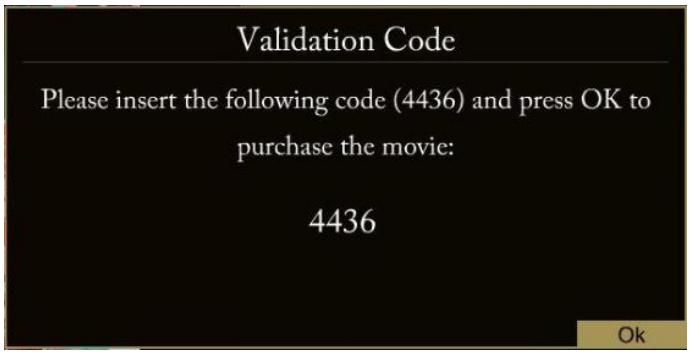


Figura 43 *Pop-Up* para inserir o código de validação da compra do filme

Quando o hóspede insere o código correctamente, então a taxação do filme é efectuada. Para isso a aplicação terá de chamar novamente página anterior, mas desta vez adiciona o argumento '*pin*' que corresponde ao código obtido na resposta anterior.

Após o segundo pedido ser enviado, o *backend* envia um '`OK`' se tudo correr bem. Esta resposta significa que o filme já foi taxado na conta do hóspede e então pode-se proceder à visualização do mesmo.

Em relação à hipótese do filme já ter sido comprado anteriormente, então o *backend* irá enviar uma resposta '`NO_PIN`'. Como o filme já foi comprado, não é necessário receber novo código nem fazer uma nova taxação, por isso procede-se à visualização directa do filme.

Existe sempre a possibilidade de algo correr mal e o *backend* enviar um '`ERROR`' e por essa razão não ser possível proceder à compra do filme. Isto pode ocorrer se houver um problema com o servidor de VOD ou então se o hóspede já tiver feito *check out*.

5.9.2. MUDANÇA DE ÁUDIO

Com a diversidade de pessoas de diferentes idiomas a passarem pelos hotéis, é importante apostar em conteúdos com vários idiomas, seja no áudio como nas legendas. As *streams* de vídeo usadas para o VOD têm, muitas das vezes, várias faixas de áudio associadas a cada uma delas. Desta forma é possível, numa única *stream*, enviar vários áudios ao mesmo tempo, permitindo assim a troca entre eles, em tempo real. A função `change_audio()` permite fazer isso nesta plataforma, sendo esta chamada quando o hóspede prime o botão de mudança de áudio no comando.

O método `change_audio()` começa por obter a lista de idiomas de áudio suportados pela *stream* actual, através de um método da API da TV (ex.: "`en,pt,de`"). Quando o botão de mudança de áudio do comando é premido, é incrementado um contador, indicando o índice do áudio em que está no momento, para se poder localizar quando for premida a tecla uma segunda vez. Para indicar que o áudio é realmente mudado, um OSD (*On-Screen Display*) é apresentado no canto superior direito, com uma imagem alusiva do respectivo idioma. A Tabela 16 apresenta as imagens com os idiomas mais usados:

Tabela 16 Imagens apresentadas quando se muda o áudio de um filme

Inglês (EN)	Português (PT)	Espanhol (ES)	Francês (FR)	Italiano (IT)	Alemão (DE)
					

5.9.3. CONTROLO PARENTAL

O controlo parental é um sistema implementado no *plugin* do VOD para controlar a visualização de certos conteúdos. Em certas *playlists* é possível encontrar categorias com filmes de adulto, definidas como *Adult R18* ou *Adult C18*. Como o sistema pode ser usado por crianças, é necessário implementar uma solução de forma a evitar o acesso a estes conteúdos.

Quando o hóspede faz o *check in* na recepção do hotel, este será questionado se quer ter acesso a estes conteúdos ou não. Se concordar, as respectivas categorias irão aparecer desbloqueadas na lista do VOD, permitindo assim o acesso imediato às capas e às descrições do filme. Caso contrário, as categorias aparecem bloqueadas, estando assinaladas com uma cor diferente, como apresenta a Figura 44. Ao percorrer a lista, a aplicação vai saltar as categorias de adulto, avançando automaticamente para a próxima categoria permitida, escondendo assim o conteúdo.

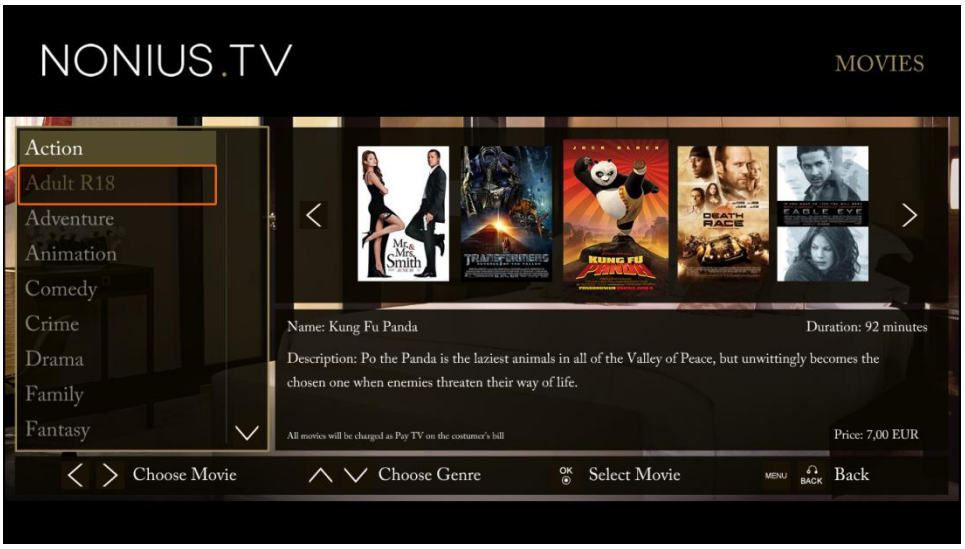


Figura 44 Apresentação do ecrã do VoD, com a categoria de adulto bloqueada

O *WebService* `GetRoomInfo` permite também verificar se o controlo parental está activo ou não.

6. CONCLUSÕES

Num mercado em grande competitividade e susceptível a grandes variações, torna-se imprescindível a uma empresa de telecomunicações como a Nonius Software procurar trabalhar em novos projectos que envolvam as últimas tecnologias do mercado, com o objectivo de tornar os seus produtos mais competitivos e mais atractivos.

Foi com este objectivo que foi criado o projecto "NONIUS.TV na Smart TV LG Pro:Centric", que serviu para integrar as tecnologias das plataformas de televisão interactiva já existentes na Nonius numa Smart TV, mais especificamente na Smart TV LG Pro:Centric.

Ao longo deste relatório apresentaram-se o alguns dos processos de desenvolvimento e os estudos efectuados para a criação da plataforma de televisão interactiva para a LG Pro:Centric. Foi feita uma pequena análise do mercado das Smart TVs em relação às tecnologias e soluções já existentes e foram apresentadas algumas das suas vantagens e desvantagens. Foi descrita a infra-estrutura e os componentes de rede necessários para ter um serviço da NONIUS.TV a funcionar correctamente, assim como uma pequena descrição das outras plataformas de TV interactiva que a empresa tem e os componentes gráficos necessários para a construção da plataforma. Como foi usado o modelo de Smart TV LG Pro:Centric durante a criação deste projecto, foram também apresentadas algumas

das características da mesma a nível de *middleware* e as ferramentas e a linguagem de programação utilizadas. Por fim foi descrita cada etapa da implementação efectuada e os respectivos resultados obtidos.

Todos os estudos efectuados sobre as capacidades da tecnologia Flash, através da linguagem de programação ActionScript, foram muito importantes para a realização deste projecto. O mesmo se pode dizer dos constantes testes efectuados às outras plataformas, de forma a implementar novas funcionalidades ou melhoramentos ao novo projecto.

Os resultados obtidos indicam que é possível fazer um sistema de televisão interactiva usando a tecnologia Flash. No entanto, o sistema criado no projecto sofre de algumas limitações. Uma delas relaciona-se com o facto de que a API da TV LG Pro:Centric não permite ter acesso à Internet, impossibilitando a visualização de páginas HTML, o que acaba por ser um ponto negativo para esta plataforma, em comparação com outros produtos semelhantes no mercado. Outro ponto negativo é que a TV só tem uma memória interna de 50Mb, limitando muito o tamanho total da aplicação. Por essa razão, durante o desenvolvimento, evitou-se anexar ficheiros externos à aplicação e usar o *backend* como alternativa. Assim, sempre que é necessário algum ficheiro, a aplicação tem que fazer o *download* do mesmo.

Neste projecto de estágio os objectivos foram atingidos. Contudo é possível efectuar algumas melhorias para futura implementação. Uma dessas melhorias identificadas é a criação de um *plugin* de promoções que permita aplicar descontos a certos produtos e serviços, em alturas diferentes. Este serviço já existe na plataforma Android da NONIUS.TV e teve uma boa adesão por parte dos clientes, sendo uma mais-valia para a aplicação. Outro aspecto a melhorar é o comportamento dos componentes gráficos quando é premido de forma constante e muito rápida certas teclas do comando, acabando às vezes por sobrepor diversos ecrãs. Apesar do uso correcto do comando não ser esse, é algo que deve ser tido em consideração.

A incapacidade da LG Pro:Centric não conseguir permitir abrir páginas *web* é uma grande deficiência do seu sistema, e a própria LG já tem a percepção disso. Por essa razão, já estão a ser criados novos modelos com um melhor desempenho, acesso à Internet e que permitem criar aplicações em HTML5. Desta forma, já será possível apostar na implementação de conteúdos e serviços através da Internet.

A nível pessoal, este estágio permitiu desenvolver competências e adquirir experiência relevante em vários níveis. A integração e aprendizagem em ambiente empresarial, foi um dos aspectos principais deste estágio, associado ao desenvolvimento de *software*. Não esquecendo a aquisição de novos conhecimentos a nível científico, como o caso da aprendizagem de uma nova linguagem de programação, o ActionScript, bem como a obtenção de experiência na aplicação de conceitos sobre *design* de interfaces gráficas para o utilizador (GUI) e sobre o protocolo RTSP.

Neste sentido, o estágio revelou-se uma mais-valia futura no âmbito de novos projectos, tendo-me potenciado novas capacidades para encarar com mais confiança o mundo profissional e dando-me a oportunidade de continuar a trabalhar na Nonius Software.

Referências Documentais

[1] PENDLEBURY, TY — *Smart TV: what you need to know*, Release 3.0, OSS Interface Developer Guide. Alcatel, 23 de Maio de 2011.

[2] MONTPETIT, Marie-José; CESAR, Pablo; MATIJASEVIC, Maja; LIU, Zhu; CROWCROFT, John; MARTINEZ-BONASTRE, Oscar — *Surveying the Social, Smart and Converged TV Landscape: Where is Television Research Headed?*, Cornell University Library, Ithaca, NY, 13 de Setembro de 2012

[3] ENGLAND, Elaine; FINNEY, Andy — *Interactive Media -What's that? Who's involved?*, ATSF White Paper-Interactive Media UK, actualizado em 2011

[4] KLYM, Natalie; MONTPETIT, Marie-José — *Innovation at the Edge: Social TV and Beyond*, Value Chain Dynamics Working Group (VCDWG), MIT Communications Futures Program (CFP), 1 de Setembro de 2008

[5] COOPER, William — *CONNECTED VISION: Broadband and Broadcast Convergence – Network Television Revolution*, informitv, England, 2009

[6] MOSKOVCIAK, Matthew — *Google TV: What you need to know (FAQ)*, CNET, http://news.cnet.com/8301-17938_105-20019591-1.html, 14 de Outubro de 2010

[7] *Google TV: How It Works*, Google, http://www.google.com/tv/features.html, visitado em 1 de Outubro de 2012

[8] KARCH, Marziah — *What Is Google TV?*, About.com Guide, http://google.about.com/od/googleonyourtv/p/what-is-google-tv.htm, visitado em 14 de Outubro de 2012

[9] informitv — *Google and Apple TV updates*, http://informitv.com/news/2012/02/13/googleandapple/, 13 de Fevereiro de 2012

[10] MOULDING, John — Google TV is looking good on Sony and LG, videonet, http://www.v-net.tv/google-tv-is-looking-good-on-sony-and-lg/, 12 de Janeiro de 12

[11] MANNINEN, JP — *Bubbling under: Samsung's Bada app store hits 100M downloads*, VentureBeat, http://venturebeat.com/2011/03/24/samsung-app-store-100m/, 24 de Março de 2011

[12] informitv — *Smart TVs show voice and gesture control*, http://informitv.com/news/2012/01/16/smarttvsshow/, 16 de Janeiro de 2012

[13] CLAUSER, Grant — What Smart TVs Need to Succeed, CE Pro, http://www.cepro.com/article/what_smart_tvs_need_to_succeed/, 7 de Dezembro de 2010

[14] MoviesOnline — *Netflix Alternative for Europe*, http://www.moviesonline.ca/2011/02/netflix-alternative-europe/, 14 de Outubro de 2012

[15] LEVY, Carmi — *Future of television is online and on-demand*, The Toronto Star, http://www.thestar.com/business/media/article/876278--future-of-television-is-online-and-on-demand, 15 de Outubro de 2010

[16] NIXON, Lyndon — *From SmartTV to LinkedTV: a vision of television in the next five years*, LinkedTV, http://www.linkedtv.eu/vision/from-smarttv-to-linkedtv-a-vision-of-television-in-the-next-five-years/, 27 de Janeiro de 2012

[17] Nonius Software — http://www.noniussoftware.com/web2/index.php/pt/produtos/noniustv, NONIUS.TV, visitado em 15 de Outubro de 2012

[18] ELST, Peter; YARD, Todd — *Object-Oriented ActionScript for Flash 8*, friendsoft, USA, 2006

[19] CARR, Dan — *Migrating from ActionScript 2 to ActionScript 3: Key concepts and changes*, Adobe, Flash Developer Center, http://www.adobe.com/devnet/flash/articles/first_as3_application.html, 17 de Janeiro de 2011

[20] LG Digital TV Lab — *Pro:Centric Software Development Kit (SDK), Version 1.0*, LG Electronics Inc, 10 de Setembro de 2010

[21] DVB — *Digital Video Broadcasting (DVB); Globally Executable MHP (GEM) Specification 1.2.2 (including IPTV)*, DVB Document A 139 r4, http://www.mhp.org/specs/a139_GEM_122_r4.pdf, Junho de 2009

[22] MORRIS, Steven — *An Introduction To OCAP*, TV Without Borders, http://www.interactivetvweb.org/tutorials/ocap, visitado em 17 de Outubro de 2012

[23] STRATFORD, Jesse — *SharedObjects*, ActionScript.org, http://www.actionscript.org/resources/articles/118/1/SharedObjects/Page1.html, visitado em 20 de Outubro de 2012

[24] Brajeshwar — *Class MovieClipLoader*, Flash 8 ActionScript 2.0 Language Reference, http://docs.brajeshwar.com/as2/MovieClipLoader.html, visitado em 20 de Outubro de 2012

[25] H. Schulzrinne; A. Rao; R. Lanphier — *Real Time Streaming Protocol (RTSP)*, RFC 2326, Network Working Group, Standards Track, http://tools.ietf.org/html/rfc2326, Abril de 1998

[26] FlashDevolep — Main Page, http://www.flashdevelop.org/wikidocs/index.php?title=Main_Page, 1 de Novembro de 2012

[27] WILLIAMS, Michael — Beginner's Guide to FlashDevelop – Basix, http://active.tutsplus.com/tutorials/beginners-guide-to-flashdevelop-intro-basix/, 7 de Janeiro de 2011

Anexo A. Estrutura dos temas do sistema NONIUS.TV

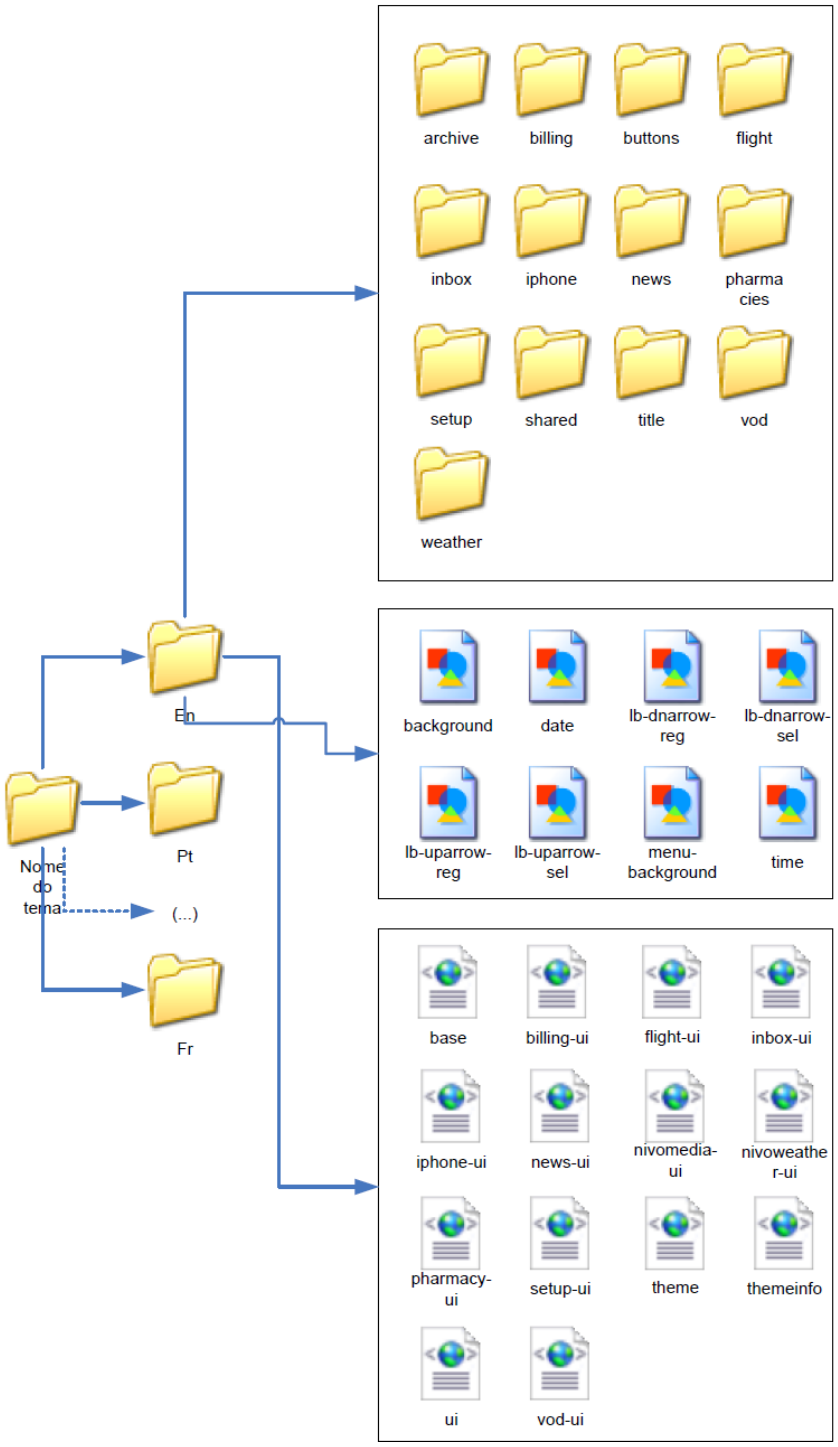


Figura 45 Estrutura dos temas

Anexo B. Ambiente de desenvolvimento FlashDevelop

O código ActionScript pode ser desenvolvido e editado com recurso a diversos editores com suporte a ActionScript. Para além do *software* da Adobe, actualmente com o Adobe Flash CS5, que necessita de uma licença, existem outros *softwares* gratuitos para o mesmo efeito. O escolhido para desenvolver este projecto foi o FlashDevelop, o qual é indicado pela própria Adobe.

O FlashDevelop, para além de ser um editor de código fonte gratuito e aberto (com licença do MIT), possui outras características importantes. Este suporta o desenvolvimento em Flash ActionScript (2 e 3): gera e completa o código de forma rápida e eficaz, assim como a compilação e o *debugguing* e permite uma extensa exploração do conteúdo dos ficheiros SWF/SWC já criados.

A partir da Figura 45 é possível observar a interface deste *software*, que permite uma boa organização no trabalho do utilizador. Através desta página inicial é possível obter mais informações e actualizações sobre o produto. Como é possível observar, este possui um painel onde é possível gerir o projecto que o utilizador esteja a desenvolver, por exemplo, criando classes a partir de modelos.

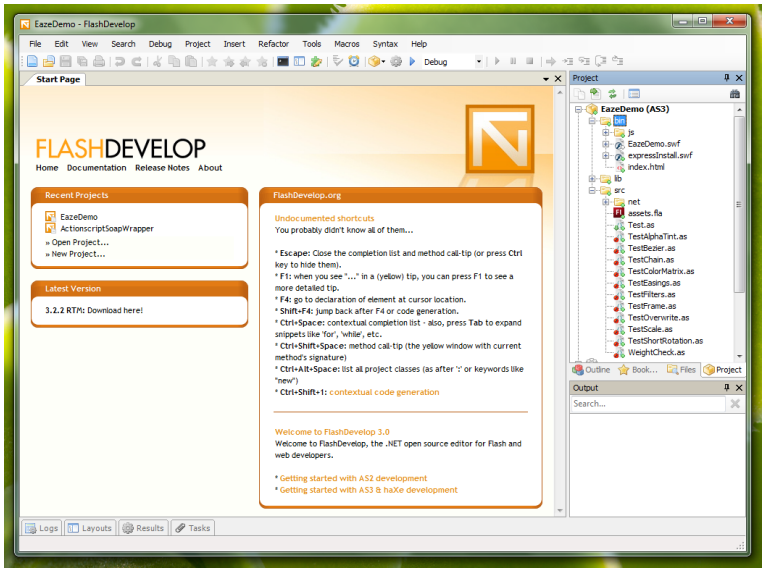


Figura 46 Interface gráfica do FlashDevelop

Estão disponíveis modelos de projecto, como apresentado na Figura 46, onde o utilizador pode facilmente personalizar ou criar os seus próprios modelos.

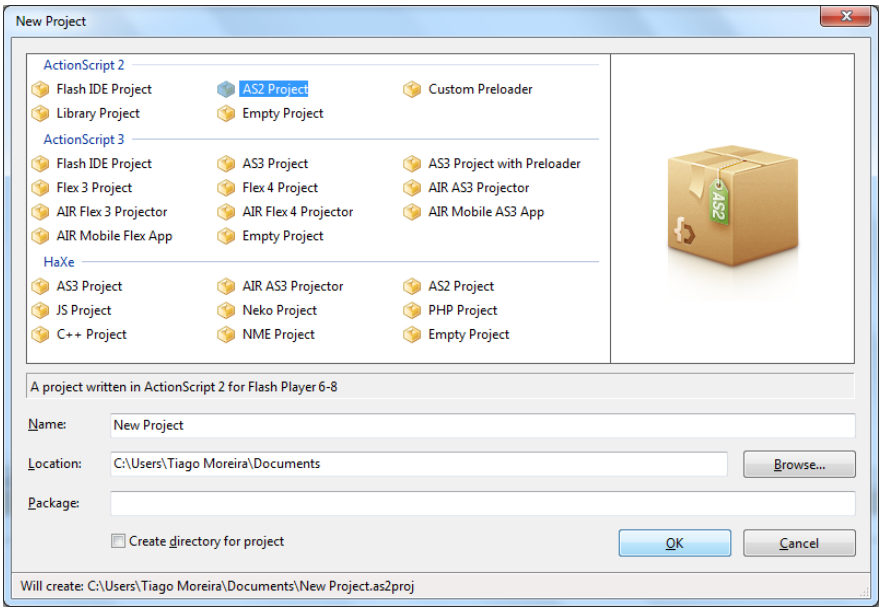


Figura 47 Modelos de projecto disponíveis no FlashDevelop

O painel de controlo (Figura 47) dá uma visão geral do código que foi desenvolvido. Basta clicar na árvore para saltar no código ou abrir classes importantes.

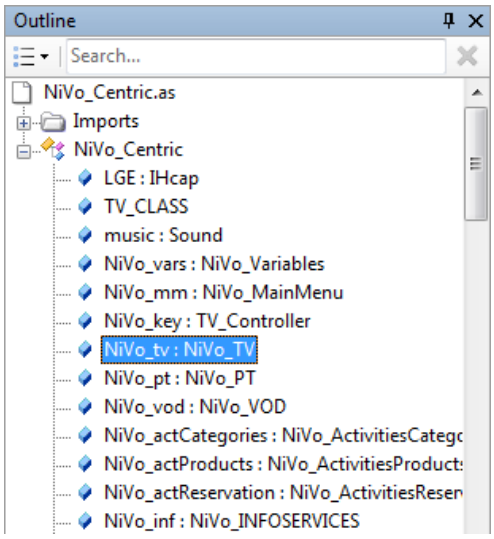


Figura 48 Painel de controlo

O painel de tarefas (Figura 48) ajuda a controlar o estado do projecto em desenvolvimento. Se for colocada uma *tag* TODO ou FIXME ao longo do código, estas *tags* vão estar listadas neste painel, assim como os erros que faltam corrigir. Também adicionar *tags* personalizadas.

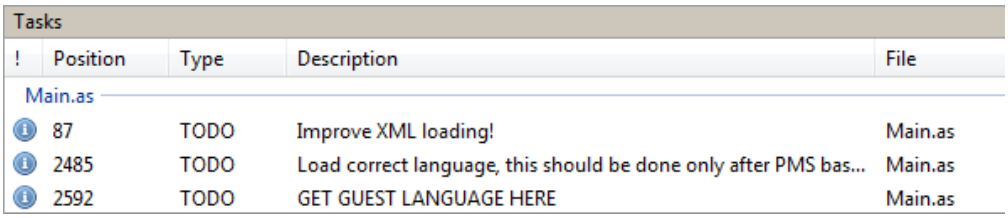


Figura 49 Painel de tarefas

Após compilar o projecto, se este contiver erros, estes vão ser listados no painel de resultados (Figura 49), de forma a permitir o acesso directo à localização dos erros relatados [26].

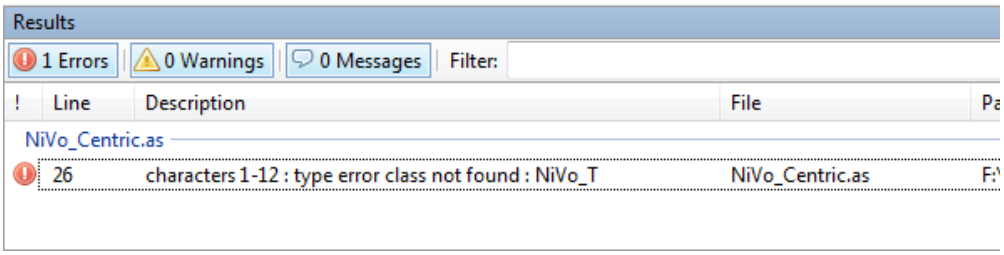


Figura 50 Painel de resultados

Uma funcionalidade muito prática e também muito usada ao longo do projecto é a possibilidade de alterar o nome de uma variável, função ou classe após esta já ter sido declarada e usada ao longo do código. No local onde estas são declaradas, é possível fazer um *Rename* (Figura 50), assim, todas as referências à variável, função ou classe, ao longo de todo o código, são renomeadas para o que for introduzido. Caso contrário, em projectos muito grandes, seria muito natural o esquecimento da alteração do nome em alguns locais do código, correndo assim o risco de dar erros em futuras compilações.

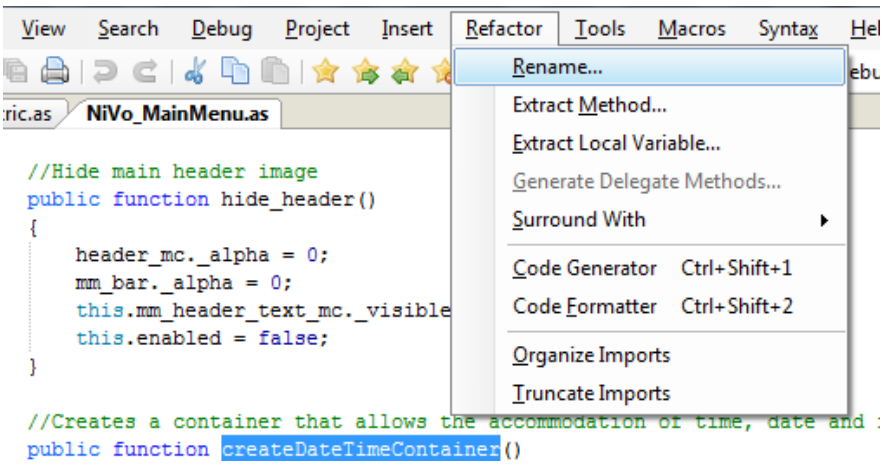


Figura 51 Funcionalidade de *Rename* no FlashDevelop

A conclusão do código no FlashDevelop funciona com classes criadas pelo utilizador ou criadas por outros. À medida que o utilizador escreve, este vai dando dicas com uma lista de possíveis classes que o utilizador queira inserir, tornado o desenvolvimento do código mais rápido e prático. O mesmo acontece com as classes do projecto: o FlashDevelop pode importa-las automaticamente (Figura 51) [27].

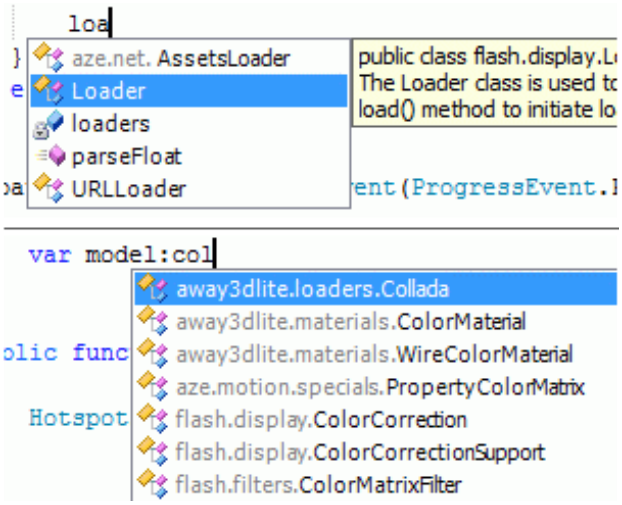


Figura 52 Funcionalidade de importação automática de classes

Anexo C. Apresentação dos principais ecrãs da NONIUS.TV | LG Pro:Centric

Como já foi referido anteriormente, a plataforma é constituída por diversos *plugins* com diferentes características. Essas características têm de ser manipuladas e acessíveis através dos diversos ecrãs interactivos que os compõem. A seguir são apresentadas algumas figuras dos outros *plugins* existentes na aplicação:

- **Canal Corporativo:** Este serviço permite fazer uma breve apresentação do hotel ao hóspede, através de um vídeo demonstrativo e ao mesmo tempo pode ir lendo pequenos trechos de notícias da actualidade através de um *feed*.

Figura 53 Esquema de apresentação do plugin do Canal Corporativo

- **Rádio:** É apresentada uma lista com várias emissoras de rádios nacionais e internacionais, onde o hóspede pode escolher o que quer ouvir.

Figura 54 Esquema de apresentação do *plugin* do Rádio

- **TV:** É apresentada uma lista com vários canais de televisão nacionais e internacionais, onde o hóspede pode escolher o que quer ver.

Figura 55 Esquema de apresentação do *plugin* da TV

- **Jogos:** É apresentada uma lista com vários jogos básicos em *flash* (que sejam possíveis jogar com o comando), onde o hóspede pode escolher o que quer jogar.

Figura 56 Esquema de apresentação do *plugin* dos Jogos

- ***Video-on-Demand*:** Ao entrar neste *plugin*, o primeiro ecrã tem uma *scroll-bar* que apresenta uma lista de géneros de filmes que o hóspede pode escolher, através de uma playlist disponibilizada pelo hotel. Juntamente com a lista de géneros, aparece uma barra com as capas dos filmes referentes ao género seleccionado no momento. Existe outro componente onde são apresentadas informações referentes ao filme seleccionado no momento (nome, descrição, duração, preço, etc.). Por fim, existe uma barra de carácter informativo, para ajudar o hóspede a manusear os diversos componentes presentes no ecrã. O hóspede ao escolher um filme, é apresentado um *pop-up* com as informações mais detalhadas sobre o filme, e a respectiva capa em tamanho maior, apresentando a opção de compra, ver trailer ou voltar para o ecrã anterior. Escolhendo a opção de compra, é apresentado um novo *pop-up* para confirmar a compra do filme. Esta confirmação é feita através da inserção de um código que está presente no ecrã. Por fim, o cliente pode ver o filme, escolhendo a legenda e o áudio nos idiomas que estão disponíveis.

Figura 57 Esquema de apresentação do *plugin* do VoD

- **Serviços de Informação - Voos:** Este permite ver os horários de partida e chegada de diversos aeroportos.

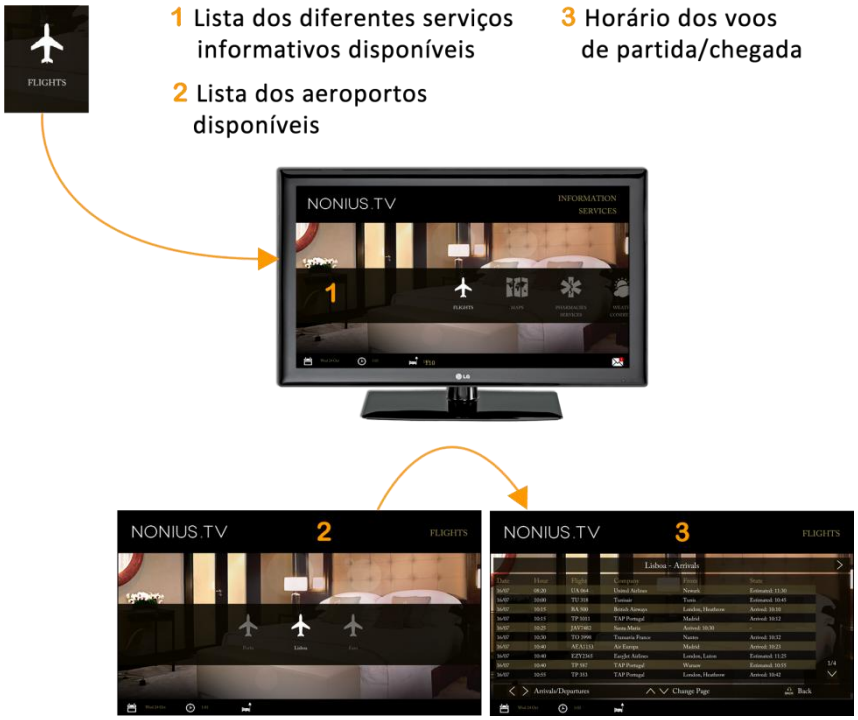


Figura 58 Esquema de apresentação do *plugin* dos Serviços de Informação

- **Conta:** Aqui são apresentadas todas as transacções efectuadas durante a estadia no hotel. Desde compra de filmes no VOD, compra de produtos no serviço de quartos e serviços do desporto e lazer. O cliente ao fazer *check-in* pode associar a sua conta a um cartão de crédito, e se não quiser perder tempo a fazer *check-out* na recepção do hotel, com este *plugin* é possível fazer um *Express Check-out* directamente do quarto, e todas as despesas serão pagas através do cartão de crédito associado.

Figura 59 Esquema de apresentação do *plugin* da Conta

- **Questionário:** Este *plugin* apresenta um número de perguntas que é definido pelo hotel, com as respectivas opções de resposta, de forma a classificar os vários serviços do hotel em termos de qualidade, experiencia durante a estadia, etc.

Figura 60 Esquema de apresentação do *plugin* do Questionário

Anexo D. Exemplo geral de um pedido *WebService*

Escolhido aleatoriamente, o código seguinte apresenta um método criado na classe `Soap_Requests` que permitirá enviar o pedido do *WebService* `setSportsAndLeisureOrders`.

```
public function setSportsAndLeisureOrders(order:String,
   language:String, oObj:Object)
{
    this.otherObject = oObj;

    updateVars();

    operationName = "setSportsAndLeisureOrders";

    headers = new Array("Content-Type", "text/xml; charset=utf-8",
       "SOAPAction","http://" + backendIP + "/WebServices/index.php/" +
       operationName);

    inputXMLSoap = "<?xml version=\"1.0\" encoding=\"UTF-8\"?><SOAP-
       ENV:Envelope xmlns:SOAP-ENV=\"http://schemas.xmlsoap.org/soap/
       envelope/\"xmlns:ns1=\"\"xmlns:xsd=\"http://www.w3.org/2001/
       XMLSchema\" xmlns:SOAP-ENC=\"http://schemas.xmlsoap.org/soap/
       encoding/\" SOAP-ENV:encodingStyle=\"http://schemas.xmlsoap.org/
       soap/encoding/\"><SOAP-ENV:Body><ns1:" + operationName +">" +
       "<OrdersXML i:type=\"d:string\">" + order + "</OrdersXML>" +
       "<language i:type=\"d:string\">" + language + "</language><ns1:"
       + operationName +"/>" +"</SOAP-ENV:Body></SOAP-ENV:Envelope>";

    getXML(true);
}
```

O método `setSportsAndLeisureOrders()` recebe como argumentos a *string* com os dados da reserva do produto, a língua que foi utilizada e a classe em que este pedido foi feito. O método começa por executar a função `updateVars()` que actualiza os campos necessários para o envio do pedido (endereço do *backend* e o caminho no *backend* onde se encontra página dos métodos dos *WebServices*). Posteriormente é criado o cabeçalho do pedido que é constituído por uma *string* onde contém o URL de destino e o nome do *WebService*. De seguida é criado um XML com uma estrutura específica de forma a ser interpretada pelo protocolo SOAP com a os dados da reserva, a língua e o nome do *WebService*. Por fim, o método `getXML()` recebe a resposta e posteriormente será reencaminhada para a classe onde foi feito o respectivo pedido.

Anexo E. Exemplo geral da criação de um *listener*

```
/**
 * Set New Listener
 * @param listener_current
 */
private function setNewListener(listener_current:Object):Void
{
    // Save the current listener
    this.listener_old = listener_current;
    // Remove listener
    Utils.clean_active_key_listeners();
    // Set new listener
    Key.addListener(this.listener_nivo_wakeup);
    // Save wake up app
    var wake_up_helper:NiVo_WakeUp = this;

    // Add onKeyDown listener
    this.listener_nivo_wakeup.onKeyDown = function() {
           var key_code = Key.getCode();
           switch (key_code)
           {
                  // Confirm Wake Up Call Phone
                  case IHcapConst.KEY_ENTER:
                  case IHcapConst.KEY_GREEN:
                         if (!wake_up_helper.existWakeUpTime()
                           && !wake_up_helper.keyEnterStatus())
                         {
                                wake_up_helper.defineWakeUpTime();
                         }
                         break;
                  // Delete Wake Up Call Phone
                  case IHcapConst.KEY_PAGE_DOWN:
                  case IHcapConst.KEY_RED:
                         if (wake_up_helper.existWakeUpTime())
                         {
                                wake_up_helper.eraseWakeUpTime();
                         }
                         break;
                  // Directional Keys used to set
                  case IHcapConst.KEY_UP:
                         if (!wake_up_helper.existWakeUpTime())
                         {
                                wake_up_helper.selectDigitUp();
                         }
                         break;
                  case IHcapConst.KEY_DOWN:
                         if (!wake_up_helper.existWakeUpTime())
                         {
                           wake_up_helper.selectDigitDown();
                         }
                         break;
                  case IHcapConst.KEY_LEFT:
                         if (!wake_up_helper.existWakeUpTime())
                         {
                                wake_up_helper.selectDigitLeft();
                         }
                         break;
                  case IHcapConst.KEY_RIGHT:
                         if (!wake_up_helper.existWakeUpTime())
                         {
```

```
                                wake_up_helper.selectDigitRight();
                            }
                            break;
                    // Exit Keys
                    case 27:
                    case IHcapConst.KEY_MENU:
                    case IHcapConst.KEY_BACK:
                    case IHcapConst.KEY_EXIT:
                            wake_up_helper.destroy();
                            break;
            }
    };
}
```

Anexo F. *pharmacies.as* e *pharmaciesItem.as*

Estas classes são aqui apresentadas para servir de exemplo para os outros serviços de informação. Em quase todos eles é necessário fazer um *parse* de um ficheiro XML e para isso são feitos alguns processos, que acabam por ser iguais em todos os serviços.

pharmacies.as

```
/**
 * ...
 * @author tsm@noniussoftware.com
 *
 */

import nivo_adapters.pharmaciesItem;
import HTML.Entities;
import ui.Alert_Message;

class nivo_adapters.pharmacies
{
 private var external_url:String;
 private var internal_url:String;

 // This is an array of news_item
 public var array_pharmacies:Array;

 private var nivo_vars:NiVo_Variables;

 private var alert_message:Alert_Message;

 // This is to limit the maximum number of pharmacies,
 // otherwise, if the number is bigger the TV crashes
 // When the parse is made it gets a blank value
 // intermediate, in this case, the limit number of
 // pharmacies is 36 so, the double is 72.
 // But the first value is always blank, so 72 + 1 = 73

 private var max_number_of_pharmacys:Number = 73;

 public function init(Nivo_vars:NiVo_Variables)
 {
       this.nivo_vars = Nivo_vars;
 }

 public function load_pharmacies(external_object:Object)
 {
       this.external_url = this.nivo_vars.backend +
          "/nivo_adapters/pharmacies.xml";

       var xml_pharmacies:XML = new XML();

       alert_message = new Alert_Message(Utils.
          getCurrentListener() , -1, -1, Translator.
          get_text_per_language("txt_loading_message"),
          "all", 600, 300, Utils.COLOR_MIDDLE_BAR, 90,
          Utils.THICKNESS, Utils.COLOR_BROWN_GOLD, 30,
          Translator.get_text_per_language("txt_pharmacies
          "), this, "Ok");
```

```
        alert_message.loadingMessageLayout();
        alert_message.show();

        // Load xml file from BE
        xml_pharmacies.load(external_url);

        var helper:pharmacies = this;

        xml_pharmacies.onLoad = function (success)
        {
                if (success)
                {
                        // Process xml
                        helper.parse_pharmacies(xml_pharmacies);
                        helper.alert_message.destroy();
                        external_object.show();
                }
                else
                {
                        xml_pharmacies.load("content/
                           nivo_adapters/pharmacies.xml");
                }
        }
}

private function parse_pharmacies(xml_pharm:XML)
{
        var helper:pharmacies = this;
        var xml_node:XMLNode;
        var pharmacy:pharmaciesItem;
        var aux_str:String;
        var aux_num:Number = 0;

        array_pharmacies = new Array();

        if (xml_pharm.childNodes[1].childNodes.length <
           max_number_of_pharmacys)
        {
                aux_num = xml_pharm.childNodes[1].
                   childNodes.length;
        }
        else
        {
                aux_num = max_number_of_pharmacys
        }

        for (var i:Number = 0; i < aux_num; i++)
        {
                xml_node = xml_pharm.childNodes[1].
                   childNodes[i];
                if (xml_node.nodeName == "pharmacy")
                {
                        pharmacy = new pharmaciesItem();
                        for (var j:Number = 1; j < xml_pharm.
                           childNodes[1].childNodes[i].
                           childNodes.length; j=j+2)
                        {
                                aux_str = xml_pharm.childNodes[1].
                                   childNodes[i].childNodes[j].
                                   firstChild.nodeValue;

                                switch(xml_pharm.childNodes[1].
                                   childNodes[i].childNodes[j].
                                   nodeName)
                                {
```

```
                                    case "classe":
                                       pharmacy.classe = aux_str;
                                       break;
                                    case "icon":
                                       pharmacy.icon = aux_str;
                                       break;
                                    case "marker":
                                       pharmacy.marker = aux_str;
                                       break;
                                    case "type":
                                       pharmacy.type = aux_str;
                                       break;
                                    case "phone":
                                       pharmacy.phone = aux_str;
                                       break;
                                    case "name":
                                       pharmacy.name = aux_str;
                                       break;
                                    case "municipality":
                                       pharmacy.municipality=
                                       aux_str; break;
                                    case "street":
                                       pharmacy.street = aux_str;
                                       break;
                                    case "zipcode":
                                       pharmacy.zipcode = aux_str;
                                       break;
                                    case "parish":
                                       pharmacy.parish = aux_str;
                                       break;
                                    case "lat":
                                       pharmacy.lat = aux_str;
                                       break;
                                    case "lon":
                                       pharmacy.lon = aux_str;
                                       break;
                                    case "distance":
                                       pharmacy.distance = aux_str;
                                       break;
                                }
                            }

                            array_pharmacies.push(pharmacy);
                            pharmacy = null;
                        }
                    }
                }
            }
```

pharmaciesItem.as

```
class nivo_adapters.pharmaciesItem
{
  public var classe:String;
  public var icon:String;
  public var marker:String;
  public var type:String;
  public var phone:String;
  public var name:String;
  public var municipality:String;
  public var street:String;
  public var zipcode:String;
  public var parish:String;
  public var lat:String;
  public var lon:String;
  public var distance:String;
}
```