FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Desenvolvimento de Novos Serviços Interativos de Vídeo para Smart Homes

Luís Guilherme Ribeiro de Castro Silva Martins

Mestrado Integrado em Engenharia Informática e Computação

Orientadora: Prof. Maria Teresa Magalhães da Silva Pinto de Andrade

14 de Julho de 2015

Desenvolvimento de Novos Serviços Interativos de Vídeo para Smart Homes

Luís Guilherme Ribeiro de Castro Silva Martins

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Nuno Honório Rodrigues Flores (Prof. Auxiliar)

Arguente: José Manuel de Castro Torres (Prof. Associado)

Vogal: Maria Teresa Magalhães da Silva Pinto de Andrade (Prof. Auxiliar)

---

14 de Julho de 2015

Resumo

Sendo a televisão o meio de comunicação e de entretenimento mais popular no mundo, foi com naturalidade que surgiu a ligação entre a Internet os conteúdos televisivos que chegam diariamente a mais de um bilião de pessoas a nível global. As estações de televisão estão sempre à procura de novas formas de cativar e captar mais audiência e os fabricantes responderam com as *Smart TVs*, capazes de providenciar aplicações dinâmicas que sejam passíveis de interagir com o utilizador. Outro mercado que está em forte ascensão é o das casas inteligentes, recheadas de sensores e atuadores que visam centralizar o controlo e a realização de tarefas domésticas e, desta forma, aumentar o conforto dos seus habitantes.

Esta dissertação assenta sobre estas duas temáticas, propondo a criação de uma solução que seja capaz de receber a informação de vários sensores instalados numa casa inteligente, injetá-la no sinal de televisão doméstico e culminar no desenvolvimento de uma aplicação para *Smart TVs* capaz de fazer a leitura dos dados e de os mostrar no ecrã de forma simples, intuitiva e dinâmica.

O contributo principal envolve a injeção de dados personalizados no sinal de televisão, exigindo a criação de um perfil de metadados específico para a situação.

Abstract

Being the television the most popular electronic media and entertainment device in the world, its bound with the Internet appeared as a natural step. To take advantage of this integration, TV manufacturers launched in the market the so-called Smart TVs, able to provide dynamic applications with interactive content, offering broadband connections. This opened up the opportunity to TV stations stations to offer their content in new ways, to further engage and grad the audience. As a consequence, television content reaches daily, in a variety of ways, the homes of more than one billion people globally. Another market in strong ascension is the one of smart homes, using a diversity of sensors and actuators installed around the houses, aiming at automating and centralizing the domestic tasks, thus contributing to increasing the user's well-being.

This dissertation builds on these two emergent topics, proposing the creation of a solution capable of receiving information from several sensors installed in a smart home, inject it in the domestic television signal and culminating with the development of an Smart TV application able to read the data from the sensors and show it on the screen in a simple, intuitive and dynamic way.

The main contribution involves the injection of personalized data in the television signal, which requires the creation of a specific metadata profile for this case.

Agradecimentos

Esta dissertação significa para mim o culminar de um período árduo mas absolutamente fantástico e inesquecível de 5 anos, que resulta na conclusão deste curso, o qual me orgulho muito de ter frequentado. Desta forma, gostava de agradecer a toda a minha família pelo apoio e solidariedade que sempre demonstraram, mesmo nos momentos em que tudo parecia um pouco mais longe que o alcançável. Aos meus amigos que conheci ao durante estes (curtos) longos anos, dedico também umas palavras por toda a amizade, espírito de camaradagem e entreajuda demonstrado quando a nada eram obrigados e só por puro altruísmo se mantiveram por perto nas horas de maior aperto. Uma palavra dirijo também às pessoas com quem convivi nestes últimos meses na MOG, por terem sido impecáveis na minha integração e estarem sempre dispostos a apoiar em tudo o que se revelou necessário.

Fiz um esforço para evitar personalizar os meus agradecimentos, contudo houve pessoas que contribuíram diretamente para a conclusão e chegada a bom porto desta dissertação pelo que tenho que agradecer à Prof. Teresa Andrade por me ter sempre orientado com pertinência dando-me liberdade para definir o meu próprio rumo neste percurso; ao Engº Alexandre Ulisses por me ter dado a conhecer este "mundo"e me ter introduzido à equipa, ao tema e ao desafio mais exigente que abracei em toda a minha vida académica; ao Sr. Engº Victor Fernandes pelo seu excelente contributo, enorme vontade de ajudar, boa disposição e otimismo que sempre demonstrou; e por fim aos meus pais, aos quais devo tudo o quanto sei e sou.

Algures nesta dissertação está um bocadinho de todos vós.

Muito obrigado!

Luís Guilherme Ribeiro de Castro Silva Martins

"The path of the righteous man is beset on all sides
by the inequities of the selfish and the tyranny of evil men.
Blessed is he who, in the name of charity and good will,
shepherds the weak through the valley of darkness,
for he is truly his brother's keeper and the finder of lost children.
And I will strike down upon thee with great vengeance and furious anger
those who attempt to poison and destroy my brothers.
And you will know my name is the Lord when I lay my vengeance upon thee!"

Ezequiel 25:17

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

AIT	Application Information Table
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ATSC	Advanced Television System Committee
CAT	Conditional Access Table
CSS	Cascade Style Sheet
DMB	Digital Multimedia Broadcasting
DOM	Document Object Model
DSM-CC	Digital Storage Media - Command and Control
DVB	Digital Video Broadcasting
ETSI	European Telecommunications Standards Institute
FIFO	First In First Out
HbbTV	Hybrid broadcast broadband TV
HTML	HyperText Markup Language
ISDB	Integrated Services Digital Broadcasting
JSON	JavaScript Object Notation
M2M	Machine to Machine
MHEG	Multimedia and Hypermedia Expert Group
MHP	Multimedia Home Platform
MPEG	Moving Picture Experts Group
MVC	Model-View-Controller
MXF	Material eXchange Format
NIT	Network Information table
PAT	Program Association Table
PCR	Program Clock Reference
PES	Packetized Elementary Stream
PID	Packet Identifier
PMT	Program Map Table
PSI	Program Specific Information
REST	Representational State Transfer
RFID	Radio-Frequency Identification
RPC	Remote Procedure Call
SDK	Software Development Kit
SMPTE	Society of Motion Picture and Television Engineers
TS	Transport Stream
VTR	Video Tape Recorder
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSD	XML Schema Definition

Capítulo 1

Introdução

A televisão é o meio de comunicação mais popular em todo o mundo, muito graças à ótima relação de custo/benefício que oferece[eLH13]. É raro encontrar um lar que não tenha pelo menos um televisor, proporcionando uma enorme variedade de conteúdos, para os quais contribuem tanto os canais públicos como os pagos.

Desde muito cedo se tentaram implementar serviços que proporcionassem interatividade entre os telespectadores e o conteúdo que estão a assistir. Inicialmente essa interatividade fazia-se através do telefone mas com o evoluir das tecnologias e da banalização da Internet no ambiente doméstico, esta tornou-se a principal forma de interação, nomeadamente agora que cada vez mais televisores se encontram ligados à rede.

Hoje em dia existe o conceito de *Smart TV*, que consiste na integração da ligação à Internet em televisores e *set-top boxes* [Kov10], alargando as suas potencialidades de forma considerável. Desta forma cria-se uma fusão entre o mundo *broadband* e o *broadcast*, potenciando de forma substancial a criação e a inovação em serviços interativos.

Outro conceito cada vez mais relevante nos dias de hoje são as *smart homes*, relacionado com a automatização da atividade doméstica, vulgarmente designado por domótica. Uma *smart home* caracteriza-se por possuir sensores, controladores e interfaces, todos ligados em rede, com o objetivo de centralizar o controlo e monitorizar o estado atual da casa e dos seus equipamentos.

1.1 Contexto

O tema desta dissertação relaciona-se com serviços interativos de vídeo e domótica, tentando integrar as duas áreas numa única solução.

Esta dissertação surge no âmbito de conclusão do curso de Mestrado em Engenharia Informática e Computação, sendo realizado em parceria com a empresa MOG Technologies, sob a orientação da Professora Maria Teresa Andrade, docente na Faculdade de Engenharia da Universidade do Porto.

1.2 Motivação e Objetivos

A vulgarização da Internet em praticamente todos os dispositivos facilitou imenso o acesso à informação, nomeadamente em aparelhos portáteis, muito mais fáceis de transportar e de interagir. Acontece que mesmo tendo a Televisão toda a popularidade que lhe está associada, esta foi praticamente das últimas tecnologias a receber as vantagens que uma ligação à Internet lhe pode proporcionar, permitindo expandir as suas capacidades e providenciar aos utilizadores uma experiência dinâmica e interativa que complementasse o serviço já existente. É neste contexto que surge esta dissertação, cuja proposta de solução a desenvolver é uma aplicação interativa para *Smart TVs*, capaz de apresentar ao utilizador dados recolhidos através de sensores numa *smart home* enviados através do sinal de televisão doméstico.

Desta forma, o objetivo do trabalho desenvolvido consiste em alcançar as seguintes metas:

- Definir a arquitetura da solução proposta
- Definir requisitos de especificação
- Criar uma rede de sensores que simule o ambiente doméstico numa *smart home*
- Definir um esquema de metadados a transmitir associado à temática dos sensores e ao tipo de informação que estes podem difundir
- Desenvolver uma aplicação que seja capaz de receber o sinal de vídeo e o injete com a informação dos sensores, criando um sinal melhorado
- Criar uma aplicação para *Smart TV* capaz de receber e interpretar o sinal de televisão melhorado, permitindo a sua visualização no ecrã
- Implementar um protótipo funcional da solução proposta

Os objetivos descritos prevêem a definição da arquitetura da solução, assim como a especificação dos requisitos da mesma, com vista à planificação do desenvolvimento e à estruturação inicial do protótipo a conceber. É essencial que ambos sejam bem-sucedidos, de forma a definir limites que permitam manter o foco no âmbito principal do projeto e guiar o processo de desenvolvimento durante a implementação. Há também etapas intermédias tais como a montagem de uma rede de sensores, definição de um esquema de metadados apropriado, desenvolvimento de um mecanismo automático de manipulação do sinal de vídeo e o desenvolvimento da aplicação interativa, que culminam na implementação de um protótipo funcional da solução. Todas estas etapas fazem parte do processo de desenvolvimento do projeto, sendo cada uma essencial para a prova de conceito, que consiste na demonstração da exequibilidade da solução proposta nesta dissertação.

1.3 Estrutura do Documento

Este documento está dividido em 4 capítulos distintos, sendo este primeiro uma introdução à proposta e uma pequena contextualização sobre os temas abordados. O capítulo 2 descreve o

estado da arte relacionado com várias temáticas relevantes para o projeto. O capítulo 3 aborda a especificação e as diferentes fases do processo de desenvolvimento da solução, incluindo e os testes e a validação a que foi sujeito o produto final. Por fim, no capítulo 4 é apresentada a conclusão relativa à implementação, as dificuldades encontradas, a satisfação dos objetivos propostos e ainda algumas melhorias e trabalho futuro a desenvolver no âmbito deste projeto.

Capítulo 2

Estado da Arte

Os temas afetos a esta dissertação são amplamente conhecidos e divulgados pela população mundial, havendo um forte investimento na tentativa de criar normas e padrões que sustenham plataformas a nível global.

A solução a ser desenvolvida, pretende-se não só que que seja eficiente mas também moderna e adaptada às tecnologias existentes e mais atuais no mercado, pelo que as secções que se seguem descrevem algumas tecnologias, conceitos e normas que são usadas pelas diversas indústrias, de forma a identificar quais as melhores opções para tornar a solução abrangente e realmente útil, tanto para o consumidor final como para a indústria televisiva.

2.1 Redes de Sensores

2.1.1 Internet-of-Things

O conceito de *Internet of Things* popularizou-se em 1999, quando investigadores do MIT publicaram uma série de artigos sobre análise de mercado e projetaram uma ideia de todos os objetos do quotidiano estarem identificados por RFID[1] (tecnologia utilizada para identificar produtos nas superfícies comerciais) e ser possível a sua gestão e controlo através de uma unidade central [Rou14].

À medida que a tecnologia foi evoluindo, os dispositivos foram ficando cada vez mais pequenos, baratos e portáteis pelo que o conceito ganhou novos contornos. Atualmente, é possível definir *Internet of Things* como o resultado do progresso tecnológico nas áreas de sistema integrados, computação ubíqua e comunicação máquina a máquina (M2M) de forma a aumentar o conforto, qualidade, produtividade e eficácia automatizando processos que de outra forma requereriam a intervenção humana [Fel14].

[1] *Radio-frequency identification*.

2.1.2 Qeo

Qeo é uma ferramenta de comunicação desenvolvida pela empresa Technicolor que tenciona ligar serviços de *smart homes* e de entretenimento à computação *cloud*. É resultado do investimento de um consórcio de empresas do ramo que pretendem criar uma interoperabilidade entre ecossistemas fechados e permitir que comuniquem sem haver limitações de sistemas operativos, *middleware* ou aplicações incompatíveis entre si, criando uma camada de comunicação flexível e uniforme. Toda a *framework* é disponibilizada online com código aberto e o objetivo principal é facilitar o trabalho dos programadores, evitando chamadas desnecessárias à API, a configuração de redes e dispositivos, a implementação de protocolos de segurança para as comunicações e focando essencialmente o esforço na lógica da aplicação e não na estrutura da rede e troca de mensagens [Tec].

2.1.3 Orion Context Broker (FI-WARE)

O FI-WARE é um projeto financiado pela Comissão Europeia cujo objetivo é criar ferramentas abertas e de livre acesso que aumentem a competitividade dos parceiros europeus no mercado global das tecnologias associado aos sistemas de saúde, telecomunicações, ambiente, entre outros. Um dos focos do vasto catálogo do FI-WARE é exatamente a potenciação da *Internet of Things*, onde apresenta o *Orion Context Broker*. Esta ferramenta é bastante útil pois permite criar interfaces entre um *Context Broker* (intermediário) e vários *Context Providers* (sensores), possibilitando que a informação seja toda reunida num só sítio e gerida a partir daí [eORR14]. Os dados obtidos podem ser requisitados quando necessários ou então enviados espontaneamente quando associados a eventos. Para isso, os *Context Providers* são registados no sistema para que depois qualquer entidade possa fazer pedidos ou subscrever esse mesmo dispositivo. Esta solução é bastante eficiente uma vez que a gestão de registos e pedidos pode ser feita facilmente, sem prejuízo para a performance do sistema. É também bastante escalável relativamente ao número de entidades que se encontram no sistema e como utiliza conceitos abstratos para a implementação de toda a arquitetura, não restringe o tipo de equipamento que pode ser usado. Para além de tudo isto, possui suporte nativo para a utilização de uma base de dados não-relacional, usada para armazenar informações sobre as entidades do sistema que são guardadas em permanência para consulta a qualquer altura [AB13].

2.1.4 Arduino

O Arduino é uma plataforma de prototipagem para desenvolver de forma rápida dispositivos dotados de alguma capacidade de processamento, sem exigir ao *developer* conhecimentos muitos avançados no que diz respeito à componente de programação. Normalmente o Arduino está associado a algumas placas de desenvolvimento com um micro-controlador programável, que permitem a ligação de diversos equipamentos (inclusive sensores), abrindo um leque de possibilidades praticamente infinito. [Ard15b]

Os programas desenvolvidos para o Arduino são escritos em C/C++ e executados através de uma biblioteca chamada Wiring que é muito utilizada para operações de leitura e escrita no controlador. Para expandir as funcionalidades do Arduino, são utilizadas placas designadas de *shields* que literalmente se encaixam através de pinos I/O[2] e permitem alargar a conetividade do Arduino com ligações à rede, aumento do número e variedade de portas disponíveis, entre outros. É ainda disponibilizado um IDE[3] que permite criar e editar os programas para o Arduino, assim como também permite interagir com as diferentes placas que estiverem ligadas ao computador através da linha de comandos, proporcionando desta forma uma interação durante a execução do código instalado. [Ard15c]

2.2 Televisão Digital

O conceito de televisão nasceu em França no início do século XX a partir da palavra *télévision*, cuja etimologia reside na conjugação da palavra grega *tele*, que significa "à distância" e do latim *visio*, que significa visão. A sua definição estende-se a todos os processos que estão adjacentes à emissão televisiva, incluindo filmagens, produção, edição, transmissão e existe até quem chame, erradamente, televisão ao aparelho televisor que possui em casa [eLH13].

Desde a sua criação, a televisão tem sofrido uma enorme evolução, muito graças à evolução da tecnologia nos equipamentos eletrónicos que em poucas dezenas de anos, potenciaram um crescimento enorme em todas as áreas afetas à emissão televisiva, nomeadamente a "revolução digital" [eLH13].

A televisão digital é o formato utilizado atualmente para a transmissão do sinal televisivo, substituindo o extinto sinal analógico. Este recorre ao sinal de som e imagem processados digitalmente para oferecer aos consumidores uma experiência audiovisual de maior qualidade.

Existem atualmente 4 normas utilizadas em larga escala e que normalmente correspondem a uma localização geográfica ou a um consórcio de países que se uniram para apoiar uma tecnologia.

- DVB - o *Digital Video Broadcast* foi criado em 1998 e é o conjunto de padrões de televisão digital mais usado em todo o mundo, nomeadamente na Europa, grande parte de África, Oceânia e ainda muitos países asiáticos como Índia e Indonésia. Utiliza as normas MPEG-2 ou MPEG-4 para compressão e transmissão dos conteúdos de vídeo.

- ATSC - o padrão da *Advanced Television Systems Committee* foi criado no início dos anos 90 e é usado essencialmente na América do Norte, América Central e Coreia do Sul. A norma de transmissão foi atualizada recentemente para MPEG-4.

- ISDB - a norma *Integrated Services Digital Broadcasting* é japonesa e diz respeito a televisão e rádio. É usada principalmente no Japão e em praticamente todos os países da América do Sul. Utiliza as normas MPEG-2 ou MPEG-4 para compressão e transmissão dos conteúdos de vídeo.

[2] *Input/Output*
[3] *Integrated Development Environment*

- DMB - a norma *Digital Multimedia Broadcasting* foi criada na Coreia do Sul, contudo é usada principalmente na China. Como é uma tecnologia mais recente que as anteriores, suporta nativamente a norma de transmissão MPEG-4.

2.2.1 MPEG-2/MPEG-4

MPEG é acrónimo para *Motion Picture Experts Group* e é um grupo de especialistas que definiu as mais utilizadas normas na reprodução de vídeo, áudio e metadados associados, sendo as mais conhecidas a MPEG-1, MPEG-2, MPEG-4, MPEG-7 e MPEG-21 [Gal91].

As normas MPEG-2 e MPEG-4 são utilizadas para transmitir imagem e som de elevada qualidade reduzindo a ocupação de largura de banda, graças a algoritmos de elevada compressão [Gal91].

A transmissão do sinal é feita por pacotes através de um fluxo de informação denominado de *Transport Stream*. Os dados podem ser separados em três tipos: vídeo, áudio e dados. O objetivo é utilizar o mesmo canal para transmitir vários programas, vulgarmente designados por "canais" de televisão.

2.2.1.1 Transport Stream

O MPEG *Transport Stream* (TS) é o formato usado pelas normas MPEG para transmitir informação relativa a vídeo, áudio e dados associados. É utilizado um protocolo que assegura que a informação é enviada e recebida corretamente, garantindo a sincronização dos conteúdos.

Os pacotes de TS (Fig. 2.1) têm um tamanho de 188 bytes e podem dividir-se essencialmente em três partes: o cabeçalho, o campo de adaptação (*adaptation field*) e o conteúdo do pacote (*payload*) [Urb11]. O cabeçalho tem um tamanho de 4 *bytes* e os campos mais importantes são: o *Sync Byte* que serve para sincronizar os pacotes e definir onde começa e onde acaba o pacote; o PID para identificar o pacote; e o *Continuity Counter* que ajuda a identificar falha na transmissão e perda de pacotes [Tek00]. O campo de adaptação é opcional na medida em que nem sempre precisa de ser enviado e serve essencialmente como extensão do cabeçalho para sincronizar os conteúdos com a hora do descodificador. Para isso, é usado o *Program Clock Reference* (PCR) que ajuda a eliminar as latências provocadas pelo processo de transmissão e permite reproduzir o conteúdo audiovisual tal como foi codificado [Urb11]. O resto do pacote de TS transporta todo o conteúdo útil para a transmissão que são designados por *packetized elementary stream* (PES), que são responsáveis por armazenar informação relativa ao áudio, vídeo, dados e PSIs de cada programa ("canal").

De forma a identificar no fluxo de dados que partes são relativas a cada programa ("canal"), está definida no TS um conjunto de tabelas designado por *Program-specific Information* (PSI). O PSI é composto por 4 tabelas, sendo que apenas duas têm uma estrutura definida:

- *Program association table* (PAT) – Esta tabela possui sempre *Packet Identifier* (PID) 0x0000. Ela é responsável por listar todos os programas ("canais") que são enviados pelo TS, identificando para cada um deles os PIDs das PMTs correspondentes.

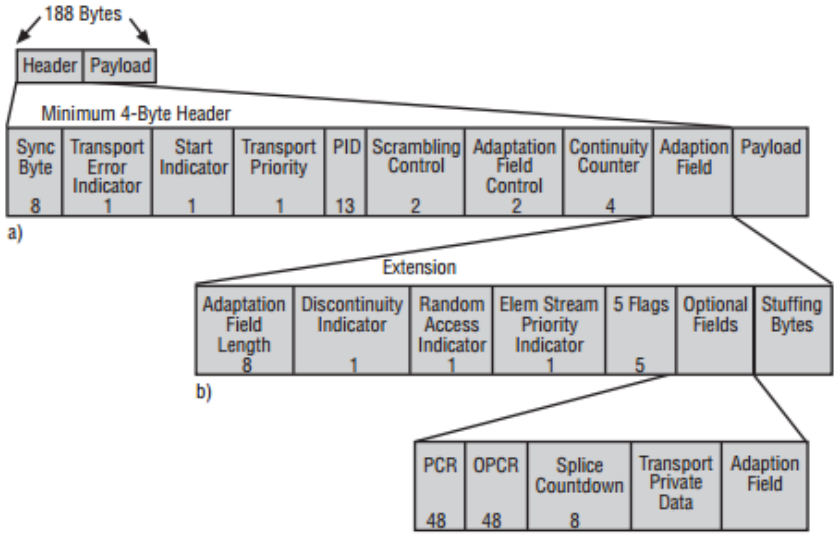

Figura 2.1: Arquitectura de um pacote TS
[Tek00]

- *Program map table* (PMT) – Esta tabela identifica para cada programa, quais os pacotes que contém o áudio e vídeo e quais os seus PIDs.

As restantes tabelas são a *Conditional access table* (CAT) e *Network information table* (NIT) que não têm um formato especificado pela norma MPEG-2. A figura 2.2 mostra como funciona o mapeamento dos PES através da PSI.

2.2.1.2 DSM-CC / Data Carousel

DSM-CC significa *Digital Store Media – Command and Control* e foi inicialmente concebido para permitir dar algum controlo a dispositivos como VTRs[4] remotas mas com os avanços na televisão digital, esta tecnologia começou a ganhar funcionalidades que lhe permitiram dar início ao que seria a verdadeira televisão interativa [Mor11].

O DSM-CC tinha como objetivo criar uma espécie de RPC[5] onde seria possível executar operações no servidor remoto, possibilitando o pedido de informação sempre que tal fosse necessário. Contudo, o canal de transmissão é unidirecional, o que significa que os dados só viajam num sentido (servidor $\rightarrow$ cliente) pelo que para contornar esse problema usa-se a técnica de *Data Carousel*. O nome provém da forma cíclica como os dados são enviados pelo transmissor para um ou mais recetores que para receberem os dados que pretendem apenas têm que esperar que o módulo que contenha a informação que eles pretendem seja enviado. Esta solução elimina não só a necessidade do cliente dar *feedback* relativamente aos dados que está a receber como também resolve a situação do receptor não receber o *stream* desde o início. Um objeto carrossel pode ser enviado

[4] *Video Tape Recorder*
[5] *Remote Procedure Call*

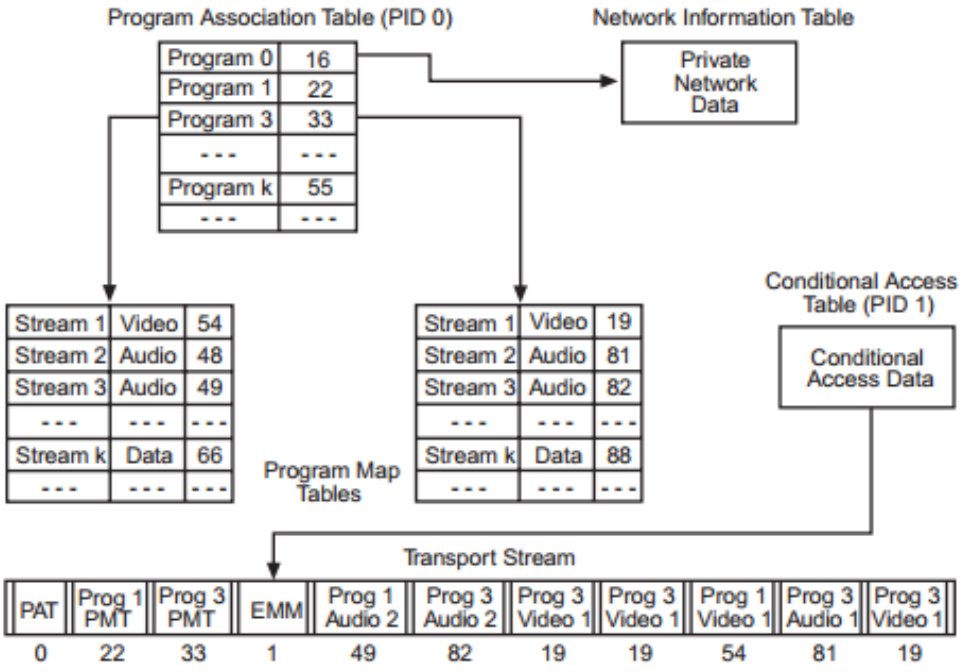

Figura 2.2: Mapeamento dos PES na PSI
[Tek00]

através de um PES, sendo possível enviar mais informação no TS que não apenas áudio e vídeo, inclusive criar programas à parte só para o envio de informação [Mor11].

As aplicações interativas podem tirar partido desta forma de enviar informação, uma vez que a maior parte dos equipamentos recentes possui *middleware* capaz de fazer o *parsing* da informação que é transportada pelo TS, dando oportunidade ao utilizador de interagir com esses mesmos dados.

2.3 Anotação de Conteúdos / Metadados

A anotação de conteúdos para dados multimédia é feita sob a forma de metadados. O significado etimológico da palavra metadados remete à composição das palavra grega *metá* e da palavra latina *datus* e exprime a reflexão dos dados sobre eles próprios, ou seja, são dados que complementam a informação que os acompanha. A sua utilidade é vasta e reconhecida, uma vez que apoiam os sistemas que lidam com a informação a organizá-la, acedê-la e interpretá-la [SS06]. Esta utilidade é ainda mais decisiva no caso dos dados multimédia, uma vez que a informação em grande parte dos casos não é inteligível, ou seja, não permite uma classificação e interpretação automática dos conteúdos.

2.3.1 MXF

A indústria televisiva lida anualmente com quantidades de informação na ordem dos *petabytes* [SS06], proveniente das mais variadas fontes e processados pelos mais diversos equipamentos e aplicações, o que por vezes traz problemas de compatibilidade. Para resolver esse problema, a *Society of Motion Picture & Television Engineers* (SMPTE), definiu um formato de contentor para conteúdos multimédia designado por *Material eXchange Format* (MXF) que não define nenhum tipo de compressão ou codificação específicos, alargando assim as possibilidades e aumentando a intercompatibilidade entre equipamentos e sistemas. Dado que o MXF é um *wrapper*, uma das principais vantagens deste formato é a associação de metadados a conteúdos de diferentes proveniências que de outra forma, não teriam maneira de interoperar [Fer10].

2.3.2 MPEG-7

Ao contrário das normas anteriormente referidas no capítulo 2.2.1, a norma MPEG-7, também conhecida como *Multimedia Content Description Interface*, especifica uma camada de anotação para conteúdos multimédia. Esta interface fornece um conjunto de ferramentas *standard* tais como descritores, esquemas de descrição e linguagem de definição de dados, que permitem uma descrição detalhada de conteúdos a diferentes níveis de granularidade e de área. Desta forma pretende-se que esta norma seja o mais abrangente possível, permitindo a compatibilidade com diferentes aplicações, normas de transmissão e tipos de conteúdo [SFC01].

A linguagem de definição de dados proposta pela norma MPEG-7 é baseada no esquema XML[6], sendo composta por um conjunto de Descritores, Esquemas de Descrição para áudio, imagem e multimédia, assim como uma Linguagem de Definição de Dados para definir novos Descritores e Esquemas de Descrição:

- Descritores – relacionam um valor a um determinado atributo.
- Esquema de Descrição – são como classes do conteúdos multimédia, definindo que Descritores pertencem a cada objeto e quais as relações com outros Esquemas de Descrições.
- Linguagem de Definição de Dados – define as regras de sintaxe para usar nos Esquemas de Descrição.

2.3.3 MPEG-21

A norma MPEG-21 foi criada no pressuposto que cada consumidor de conteúdos multimédia faz parte de uma enorme rede mundial, onde estão sempre a surgir conteúdos novos. Todas as normas MPEG têm o objetivo de aumentar a interoperabilidade entre conteúdos e utilizadores, o que por vezes, por força da vasta quantidade de origens de conteúdos, acaba por tornar este conceito um pouco contraditório. Esta norma foi então desenhada para uniformizar o modo como os conteúdos multimédia interagem entre si.

[6] eXtensible Markup Language

Criou-se desta forma o conceito de *Digital Item*, que pretende dar uma identidade a cada conteúdo, reunindo numa *Digital Item Declaration* os dados multimédia e metadados que os identificam. O MPEG-21 definiu também uma especificação para garantir os direitos de propriedade de um determinado item, graças a *Rights Expression Language* que é uma linguagem baseada em XML e que proporciona uma forma de comunicação M2M com vista a definir quais as restrições aplicadas à utilização de conteúdo digital [IB03]. Com isto, especificou-se uma forma inequívoca e segura de transmitir informação de licenciamento.

2.4 Televisão Interativa

O conceito de televisão interativa remonta à época em que a televisão se começou a popularizar como meio de comunicação, em 1920 [Jen08]. Na altura era utilizado um canal bidirecional de áudio para comunicar e desta forma interagir com os conteúdos que estavam a ser transmitidos. Para esta dissertação, é mais interessante focar nas tecnologias atuais e quais os métodos para colocar os utilizadores a controlar os conteúdos que pretendem visualizar. O objetivo é colocar na televisão informação relativa a sensores que se encontrem numa *smart home*, pelo que o foco incidirá sobretudo na visualização de informação dinâmica relacionada com esse tema, permitindo um interação eficaz e intuitiva com os dados demonstrados.

2.4.1 Smart TV

O conceito de *Smart TV*, ou TV híbrida, está relacionado com o facto de estar ligada à Internet e possuir capacidades melhoradas, provenientes dessa mesma ligação [Kov10]. O objetivo passa não só por criar e manter aplicações interativas mas principalmente usufruir dos recursos que esta ligação cria, tal como gerar conteúdos personalizados totalmente *on-demand*, promovendo uma experiência única para cada utilizador [Lev10].

Neste momento existem vários fabricantes a nível mundial, sendo que cada um possui a sua interface e arquitetura, o que dificulta um pouco a tarefa de criar aplicações que possam correr em ambientes variados ou que requeiram compatibilidade de equipamentos. Nos tempos mais recentes, tem havido um esforço para criar ambientes *standard*, de modo a que o mercado evolua no sentido da produção de mais e melhores aplicações interativas, criando uma camada intermédia de software uniforme entre equipamentos. Foram feitas várias tentativas por consórcios de produtoras e associações ligadas ao meio que tiveram mais ou menos sucesso, dentro de uma certa região ou tipo de equipamentos, pelo que o sucesso global não foi ainda completamente atingido. Outra das condicionantes é que estes equipamentos designados por *Smart TVs* são relativamente recentes e até um pouco caros, pelo que a sua aceitação, que será indispensável para o sucesso do projeto, não está assegurada.

2.4.2 Middlewares

Para desenvolver aplicações interativas, muitos fabricantes fornecem *frameworks* próprias de modo a fazer com que os programadores adotem a sua plataforma e desta forma ganhar vantagem sobre a concorrência. Esta medida leva a que haja uma maior segmentação de mercado no que diz respeito a este tipo de *software*, o que dificulta a tarefa de criar aplicações multi-plataforma. Felizmente, muitas empresas já reconheceram a necessidade de uma solução *standard* que se consiga adaptar a praticamente todos os equipamentos televisivos, pelo que nos últimos anos foram criados algumas camadas de *middleware* recorrendo a linguagens de programação conhecidas (C, Java, HTML, etc.) cujo objetivo é uniformizar a criação de conteúdos interativos [eRA05].

2.4.2.1 HbbTV

A norma HbbTV (*Hybrid Broadcast Broadband TV*) foi criada em 2009 com o objetivo de providenciar um sistema aberto e normalizado baseado na tecnologia HTML, que facilite o desenvolvimento fácil e eficiente de aplicações para vários tipos de plataformas, independentemente do equipamento, operadora ou tipo de ligação à Internet [Mer11].

No HbbTV, os conteúdos podem ser transmitidos através do sinal de televisão, permitindo às operadoras o envio de conteúdos para o consumidor final de forma bastante eficiente. É também possível receber e enviar conteúdos através da ligação à Internet. Relativamente à transmissão de conteúdos através do canal de difusão televisivo, o ETSI (*European Telecommunications Standard Institute*) criou uma especificação para o transporte de aplicações HbbTV que faz uso da tecnologia abordada no capítulo 2.2.1.2 relativa ao *Data Carousel* para introduzir os dados relativos às aplicações no fluxo de transmissão e desta forma permitir o seu envio, garantido todos os mecanismos de segurança e eficiência que estão associados às normas de transmissão e codificação de vídeo MPEG.

2.4.2.2 MHEG-5

MHEG é acrónimo para *Multimedia and Hypermedia Experts Group* e MHEG-5 é uma norma criada por essa organização para um *middleware* capaz de enviar e receber sinal de vídeo interativo. É usado principalmente no Reino Unido, na Nova Zelândia, Hong Kong, Austrália, Irlanda e África do Sul. O MHEG-5 utiliza uma linguagem proprietária para as suas aplicações, pelo que a recetividade e escalabilidade das aplicações não é muito elevada [Sof15].

2.4.2.3 MHP

O *Multimedia Home Platform* é uma norma baseada principalmente em Java de um *middleware* de televisão interativa desenvolvido pelo DVB. É utilizado principalmente na Itália, Bélgica e Polónia. O serviço prevê uma forte interatividade com a Internet, havendo três perfis que definem requisitos funcionais para uma grande variedade de aplicações e equipamentos [Pie06]:

- *Enhanced broadcasting*, onde o utilizador apenas necessita de interagir com o recetor e não com a fonte dos conteúdos.
- *Interactive broadcasting*, onde é criado um canal bidirecional que permite ao utilizador interagir com a fonte dos conteúdos televisivos.
- *Internet access*, onde é possível aceder à Internet, nomeadamente a páginas que tenham sido criadas de propósito para serem visualizadas em televisores.

A grande desvantagem do MHP é a utilização de Java para desenvolver as aplicações interativas, o que requer dos equipamentos recursos mais poderosos de forma a fazer face aos requisitos do *middleware*.

Capítulo 3

Especificação e Desenvolvimento

A solução que se pretende desenvolver visa dar ao utilizador a possibilidade de usufruir de conteúdos dinâmicos e atualizados sobre sensores domésticos na sua *Smart TV*. Posto isto, é necessário abstrair qual a infraestrutura física e lógica necessária para implementar tal solução sendo que surgem logo à partida os seguintes desafios:

- Definir como estão ligados e como é transmitida a informação dos sensores;
- Definir como é feita a recolha, a uniformização e o armazenamento dos dados recolhidos pelos sensores;
- Definir a camada de transporte da informação no sinal de vídeo (metadados);
- Criar um modelo de interatividade que defina de forma simples, intuitiva e o menos obstrutiva possível a informação que se pretende transmitir.

Focando estes tópicos aspetos tão diversos da fase de desenvolvimento, é necessário criar uma ligação capaz de os unir a todos de forma a dar uma experiência fluída e abstraindo os pontos em que a intervenção do utilizador não é de todo requerida.

3.1 Modelo de interatividade

O conceito de *Smart TV*, ou televisão conectada, abriu um número praticamente infinito de possibilidades no que toca à componente interativa da televisão pelo que é necessário definir o âmbito do desenvolvimento da aplicação e quais a suas limitações no que à aplicação diz respeito. A interatividade via Internet é uma grande mais-valia disponibilizada pelas tecnologias mais recentes, incluindo a HbbTV, contudo, o foco da dissertação reside na injeção da informação no sinal de televisão que é unidirecional e não permite que haja troca de informação entre o cliente e o fornecedor de conteúdo. Para contornar esta limitação, o que se faz é enviar toda a informação

de uma vez, sempre que os dados são atualizados. Desta forma, os dados necessários estão sempre do lado do cliente, sendo que este apenas escolhe que informação visualizar.

Pretende-se então oferecer ao utilizador uma vista simples e limpa da informação disponibilizada pelos sensores que nunca perturbe de forma significativa a programação que o utilizador já se encontra a visualizar. É desta forma uma aplicação que corre em paralelo com outra programação e que pode a qualquer momento ser consultada e minimizada sem prejuízo da experiência televisiva para o utilizador.

A navegação é por sua vez feita através dos botões de navegação do comando, permitindo escolher que informação ver e com que nível de detalhe, personalizando a sua experiência e aumentando o controlo do utilizador sobre a aplicação.

3.2 Requisitos do sistema

O desenvolvimento da solução foi baseado no levantamento inicial de requisitos que seriam o pilar estrutural da aplicação e sobre os quais cairiam todas as decisões relativamente à gestão do projeto. Para além disso, usaram-se dois pressupostos iniciais: todo o projeto seria desenvolvido tendo em vista o utilizador final e dessa forma fazer com que o processo de transmissão e visualização dos dados fosse o mais homogéneo possível.

Dito isto, de seguida enumeram-se os requisitos de sistema para a aplicação com vista um único ator, o utilizador.

Tabela 3.1: Requisitos de um utilizador

ID	Descrição
1	Um utilizador deve poder escolher quando ter a aplicação aberta.
2	Um utilizador deve poder escolher que divisões visualizar.
3	Um utilizador deve poder escolher que sensores visualizar.
4	Um utilizador deve poder escolher que intervalo de tempo consultar.
5	Um utilizador deve poder consultar a informação graficamente.
6	Um utilizador deve poder consultar a aplicação enquanto vê outro programa televisivo com o mínimo distúrbio possível.
7	Um utilizador deve poder receber informação atualizada automaticamente.
8	Um utilizador deve poder receber alertas de situações estranhas/fora do normal.
9	Um utilizador deve poder adicionar sensores à plataforma.
10	Um utilizador deve poder remover sensores da plataforma.
11	Um utilizador deve poder gerir o histórico de dados armazenados.

3.3 Arquitetura funcional do sistema

O foco desta dissertação é criar uma solução que permita aos utilizadores finais terem acesso de forma simples e intuitiva à informação que de outra forma não seria tão inteligível ou requereria mais recursos (físicos e/ou intelectuais) ao utilizador, pelo que a plataforma tem que abstrair e

tornar o mais automático possível os processos intermédios dos quais resultam o produto final. Tal como foi enunciado no capítulo anterior, todo o desenvolvimento visa o utilizador como único interveniente num ambiente doméstico, muito embora a solução especificada preveja com muito pouca ou nenhuma adaptação o suporte para outro contexto.

Do ponto de vista do utilizador, existirão duas referências imprescindíveis que são o servidor e a televisão. O primeiro é responsável por recolher, uniformizar e armazenar os dados recolhidos pelos sensores, sendo que posteriormente esses mesmos dados serão disponibilizados no sinal de televisão como forma de transportar a informação. Ainda relativamente a esta parte, será necessário que os sensores instalados tenham alguma capacidade de processamento associada, de forma a executarem simples rotinas de análise e envio dos dados recolhidos para o servidor, que de seguida fará o pós-processamento e a normalização da informação. O passo seguinte consiste em definir essa mesma informação numa camada de metadados que será responsável pelo transporte da informação através do canal de comunicação do sinal de vídeo. Estes dados serão enviados já normalizados, organizados sensivelmente da mesma forma que são demonstrados, ou seja, por divisão, por sensor e por espaço temporal ao qual correspondem os dados recolhidos.

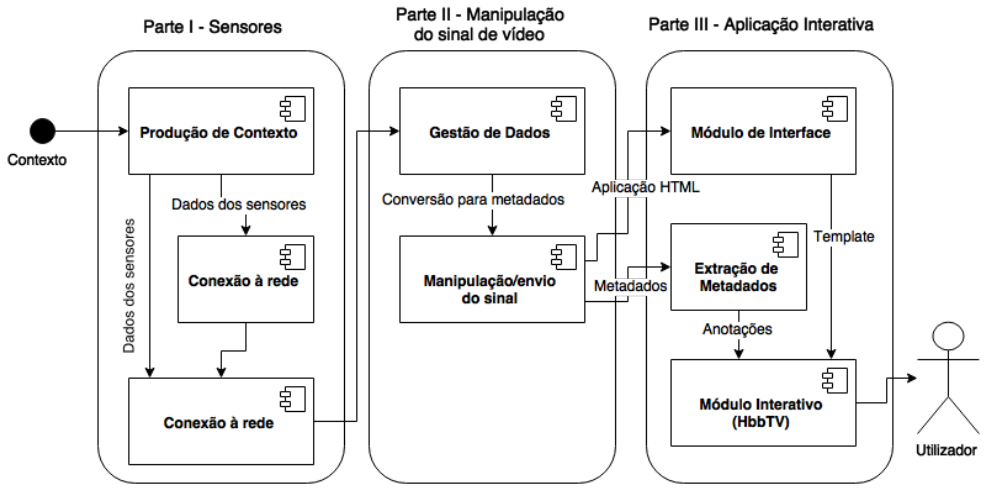

Figura 3.1: Diagrama da arquitetura funcional da solução

3.4 Metadados

Sendo os metadados informação complementar sobre determinado conteúdo que está a ser transmitido, estes são enviados ao mesmo tempo que o sinal multimédia é transmitido. Neste caso de estudo em particular, os dados precisam de ser enviados de uma forma dinâmica, de forma a que a informação que transportam seja sempre atualizada. A particularidade de serem exibidos os dados que o utilizador pretende visualizar no momento em que acede à aplicação, exige que toda a informação esteja disponível em tempo real no lado da televisão.

Este facto requer que a informação seja enviada de forma o mais organizada possível, recorrendo dessa forma a um perfil de metadados que estende o MPEG-7 e serve esse mesmo propósito.

A solução foi desenhada especificamente para o problema dos dados recolhidos através de sensores pelo que as etiquetas definidas cumprem as mesmas regras da estrutura de dados do problema, sendo que houve a preocupação de ser ambíguo na definição relativa ao tipo de variáveis para absorver informação proveniente de diferentes plataformas e tipos de sensores.

A figura seguinte ilustra o modelo conceptual da camada de metadados:

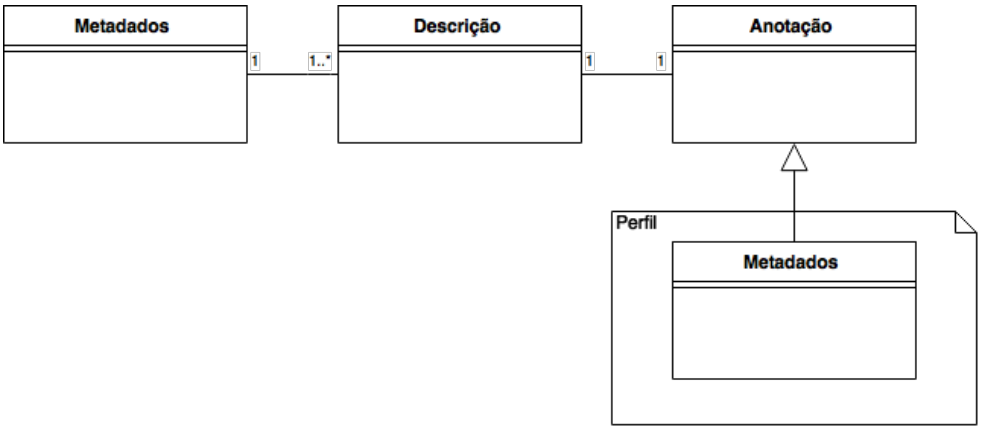

Figura 3.2: Modelo conceptual da camada de metadados

A informação será enviada através de anotações, que são habilitadas na camada de metadados graças à extensão que foi desenvolvida no âmbito da dissertação. Desta forma é possível adicionar ao conteúdo visual a informação que se pretende transmitir num formato compatível numa norma aberta e reconhecida, de forma a que possa ser desenvolvida e adaptada por alguma comunidade que deseje debruçar-se sobre esta temática.

O ficheiro de metadados é então um ficheiro *xml* que começa com uma etiqueta *Mpeg7* que por sua vez pode conter uma ou mais etiquetas *Description*. Neste caso em concreto teremos apenas uma descrição, que por sua vez conterá uma etiqueta do tipo *MultimediaContent*, especificado como atributo *type* da etiqueta anterior como sub-tipo de *ContentEntityType*.

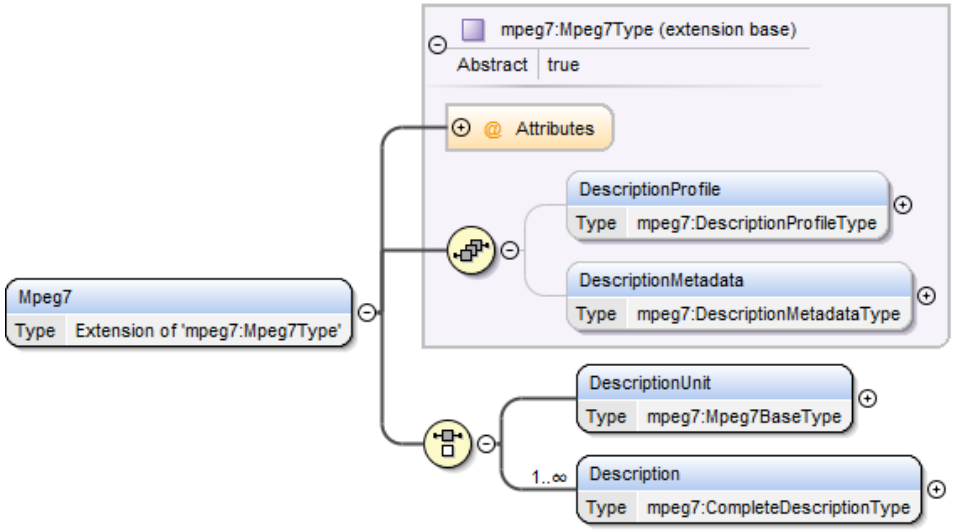

Figura 3.3: Anotação MPEG-7

Dos tipos disponíveis, escolheu-se o *VideoType* pois o conteúdo multimédia a anotar é vídeo. Dentro do *VideoType*, decidiu-se criar uma extensão para a etiqueta *VideoSegmentType* que conterá as anotações *Annotations*, tipo criado especificamente para albergar as anotações *Annotation* relativas aos dados dos sensores. Para além dessa informação muito exclusiva, é possível estender a funcionalidade do tipo *Annotation* de forma a transportar outro tipo de informação num formato mais tangível de forma simples e associado ao mesmo conteúdo que o resto dos dados.

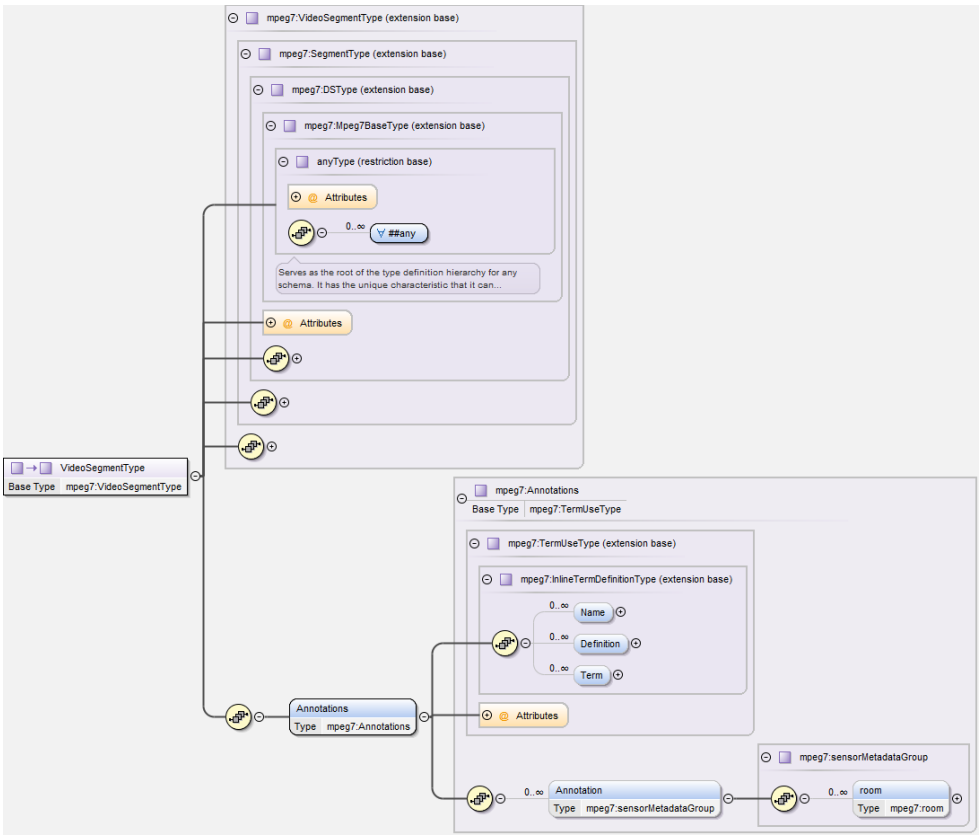

Figura 3.4: Extensão da etiqueta *VideoSegmentType*

3.4.1 Perfil de metadados

Tal como foi indicado previamente, a solução que se pretendia desenvolver levou à necessidade de se criar um perfil de metadados específico de forma a aumentar a compatibilidade com outras aplicações desenvolvidas no mesmo âmbito através de uma estrutura baseada no standard MPEG-7 e as regras definidas dessa norma.

O perfil define uma estrutura hierárquica de dados que se ajustam ao tipo de informação que a aplicação interativa necessita para mostrar os dados recolhidos pelos sensores. Na pesquisa efetuada, não foi encontrado nenhum perfil público que se adaptasse ao tipo de dados a tratar, pelo que de seguida fica o modelo conceptual relativo ao perfil de dados dos sensores:

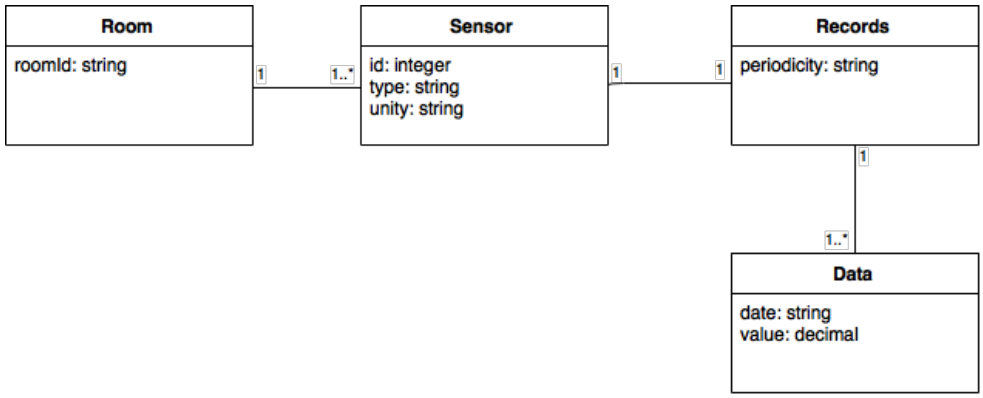

Figura 3.5: Modelo conceptual do perfil dos sensores

Será no tipo *Annotation* que estará a definição da estrutura de dados definida para este problema, sendo que este último pode conter um número de elementos do tipo *Room*, que se caracterizam por possuirem um atributo *roomId*, responsável por identificar a divisão que vai armazenar os dados dos sensores *Sensor* que lá se encontram.

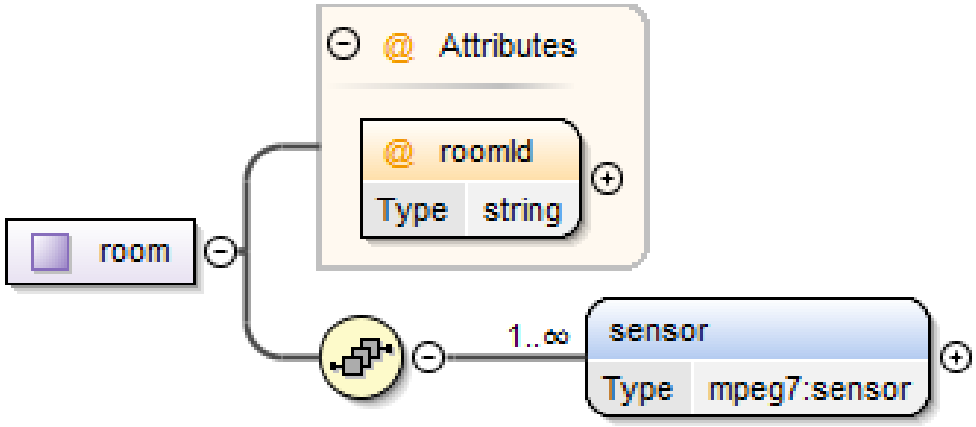

Figura 3.6: Etiqueta *Room*

Os sensores por sua vez vão conter informação relativa à sua identificação *id*, tipo de dados que medem *type*, unidade da informação *unity* e outra etiqueta chamada *records*, responsável por guardar os dados dos sensores.

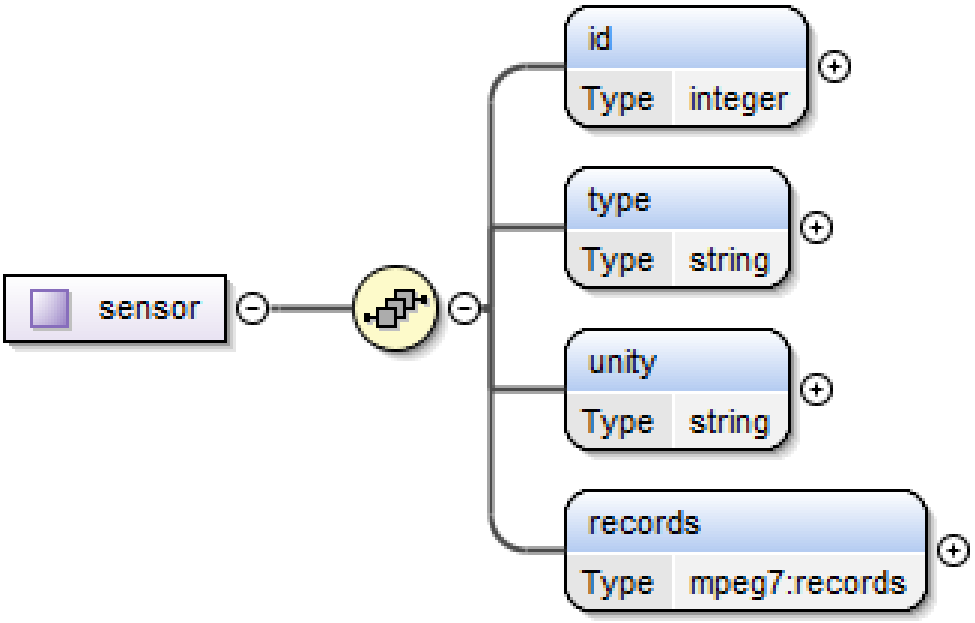

Figura 3.7: Etiqueta *Sensor*

A etiqueta *records* vai albergar dados *data* que correspondem a uma determinada periodicidade *periodicity* (p.e. um determinado conjunto de dados corresponde à informação recolhida numa semana) pelo que este último elemento terá obrigatoriamente que aparecer somente uma vez. O tipo *data* corresponde a um duplo que correlacionará um valor medido *value* com a data da sua medição *date*, ambos obtidos diretamente do sensor.

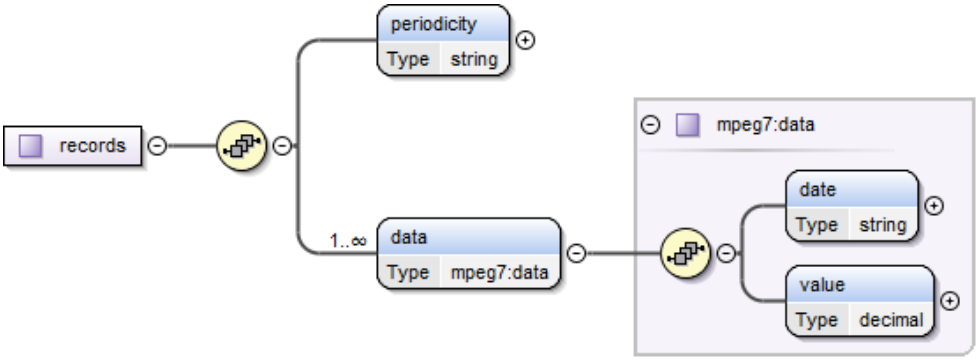

Figura 3.8: Etiquetas *Records* e *Data*

3.5 Implementação

Neste subcapítulo constam todos os detalhes relativos à implementação do projeto, desde características da arquitetura a pormenores técnicos, incluindo decisões de funcionamento das aplicações desenvolvidas, problemas encontrados ao longo do desenvolvimento e os respetivos métodos utilizados para os contornar.

3.5.1 Arquitetura da solução

Uma vez que o tema da dissertação envolve áreas de estudo tão diversas, faz sentido dividir o desenvolvimento da solução pela temática sobre a qual se debruçam. Desta forma, são criados três módulos que abordam os seguintes temas: redes de sensores, manipulação do sinal televisivo e aplicação interativa para *Smart TV*. A figura 3.9 ilustra como se separariam os diversos módulos enumerados na globalidade da solução.

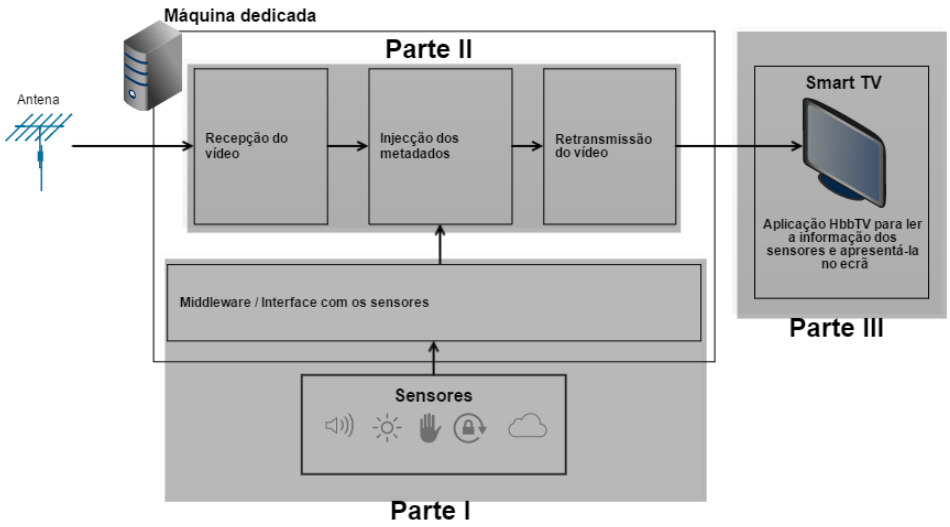

Figura 3.9: Proposta de arquitetura da solução

Sendo a amplitude do problema tão extensa, houve uma necessidade de utilizar tecnologias abertas e reconhecidas na respetiva área de estudo, de forma não só a facilitar o desenvolvimento e manutenção do código produzido mas também para criar uma solução final que utilizasse tecnologias atuais para que o seu desenvolvimento futuro e eventual utilização correspondesse às necessidades, hábitos de consumo e nível de qualidade que se espera que as aplicações interativas possuam nos dias de hoje. Essa busca é notória nas várias partes que compõem a dissertação, sobretudo nas aplicações que possuem interface com o utilizador.

3.5.2 Parte I - Rede de sensores

O objetivo do primeiro módulo é emular uma rede sensores domésticos, sendo que para isso é necessário criar uma interface capaz de fazer o pedido dos dados e o tratamento dos mesmos, que poderão vir nas mais diferentes formas e cadências, consoante o tipo de sensor que envia a informação.

Uma das características essenciais para este módulo é que seja o mais escalável possível, providenciando compatibilidade com uma grande variedade e número de sensores. Outra característica importante é a resiliência dos serviços, mantendo sempre a eficiência e segurança nas comunicações.

Por estes motivos enunciados, a solução que sobressaiu foi o *Orion Context Broker* (capítulo 2.1.3) uma vez que garante uma flexibilidade na arquitetura da rede de sensores que é exatamente o pretendido. Outra vantagem desta ferramenta é a integração com a base de dados não-relacional, pois facilita uma das funcionalidades da aplicação da *Smart TV* que é a disponibilização de um histórico relativamente à informação dos sensores.

3.5.2.1 Produção de contexto

Os sensores por si só não têm capacidade de processamento que lhes permita fazer a gestão da informação que eles próprios produzem, sendo necessário ligá-los a alguma plataforma capaz de lhes "dar vida"e permitir que a comunicação com o *Context Broker* seja efetuada utilizando protocolos previamente estabelecidos e que mais à frente serão detalhados. Esta necessidade de "inteligência"por parte dos sensores foi resolvida recorrendo ao Arduino, que graças ao seu CPU[1], permite controlar a transmissão da informação recolhida, a sua cadência e ainda um conjunto de outras informações (p.e. temporais) que complementam os dados recolhidos. O Arduino é uma plataforma de desenvolvimento de ferramentas eletrónicas, possuidor de um microprocessador programável numa linguagem inspirada em C/C++ [Ard15a].

Desta forma, criou-se uma aplicação simples, normalmente chamada *sketch* [Ard15d], que tem como objetivo ler os dados que são capturados nos sensores, transformá-los num formato compatível com o aceite pelo *Context Broker* e de seguida enviá-los. Este último passo não foi possível ser realizado porque o equipamento fornecido para testes não possui um *network shield* capaz de conectar o Arduino a uma rede local (secção 3.5.2.2). Os dados são recolhidos da porta digital à qual está ligado o sensor e são embebidos na *string* JSON que é enviada para ser armazenada.

3.5.2.2 Ligação à rede

A maior parte dos Arduinos comercializados não possuem nativamente uma porta *Ethernet* ou *Wi-Fi* pelo que para que possam ser ligados a uma rede local ou à Internet necessitam de um equipamento adicional denominado de *network shield* ou *WiFi shield* que tem como objetivo expandir as funcionalidades do Arduino e aumentar a sua conectividade.

[1] *Central Processing Unit*

Figura 3.10: *Network shield*

Para contornar esse obstáculo, criou-se um módulo em *Python* que estaria responsável por receber a *string* JSON a enviar pelo Arduino através da porta de série/USB[2]. Este módulo estaria depois responsável por comunicar com a API REST do *Orion Context Broker*, enviando o pedido com a informação que atualiza a base de dados do *Broker*. É importante ressalvar que este módulo é uma medida de recurso para contornar o problema já descrito, pelo que não faz parte da solução tal como ela foi desenhada. O Arduino é programado para produzir a *string* JSON no formato correto e, na eventualidade de ser disponibilizada a ligação à rede, com mínimas alterações, este módulo seria dispensável.

3.5.2.3 Configuração do *Context Broker*

A escolha da interface de gestão dos sensores recaiu na implementação do *Orion Context Broker* por encaixar perfeitamente nas necessidades do projeto, sendo que a sua instalação requereu algumas especificações exigidas pelos *developers*. Tal como consta no guia de instalação e configuração do *Broker* [Gal15], este foi desenvolvido e testado num ambiente *CentOS*, pelo que houve a necessidade de configurar uma máquina com esse mesmo sistema operativo de forma a minimizar conflitos de sistemas e de compatibilidade que não estariam de todo no âmbito deste projeto. Foi então instalada numa máquina a versão 7 do sistema operativo supracitado, à qual foi necessário configurar uma instalação do *MongoDB*, um sistema de gestão de base de dados

[2] *Universal Serial Bus*

não-relacional sobre o qual assenta o *Orion Context Broker*. Uma vez instalado o *Broker*, após a sua execução, fica ativo um *web service* que responde no endereço local *localhost* e pedidos REST com a seguinte estrutura:

Pedido ao servidor REST para atualizar o *Context Broker*:

```
{
    "contextElements": [
        {
            "type": "Sensor",
            "isPattern": "false",
            "id": "Sensor1",
            "attributes": [
            {
                "name": "temperature",
                "type": "float",
                "value": "21"
            }
            ]
        }
    ],
    "updateAction": "UPDATE"
}
```

Resposta do servidor REST ao pedido de atualização de informação:

```
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "temperature",
                        "type": "float",
                        "value": ""
                    }
                ],
                "id": "Sensor1",
                "isPattern": "false",
                "type": "Sensor"
            },
```

```
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ]
}
```

Pedido ao servidor REST para receber informação do *Context Broker*:

```
{
    "entities": [
        {
            "type": "Sensor",
            "isPattern": "false",
            "id": "Sensor1"
        }
    ]
}
```

Resposta do servidor REST ao pedido de informação:

```
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "temperature",
                        "type": "float",
                        "value": "21"
                    }
                ],
                "id": "Sensor1",
                "isPattern": "false",
                "type": "Sensor"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
```

```
19                          }
20                  }
21          ]
22  }
```

3.5.3 Parte II - Injeção dos dados

No segundo módulo, não há exatamente uma escolha a fazer, uma vez que as normas utilizadas são padrões da indústria televisiva, o que faz com que seja a solução a adaptar-se ao mercado e não o contrário. Para efeitos de desenvolvimento, a aplicação que vai injetar o sinal terá que receber o *stream* de vídeo, interpretá-lo, introduzir os dados referentes ao sensores através da camada de metadados e de seguida reencodificá-lo e retransmití-lo.

Em termos práticos, vai ser necessário analisar o TS do sinal de vídeo, analisar na PAT quais são os programas ("canais") que estão presentes e a que PES se referem os PIDSs que nela constam. De seguida será necessário criar um programa adicional que contenha os dados dos sensores, recorrendo à funcionalidade de *Data Carousel*, referida no capítulo 2.2.1.2. Este novo programa terá somente dados (sem áudio nem imagem) e exigirá a criação de um esquema de metadados que se adeque à informação remetida pelos sensores. Através do Objeto Carrossel será possível enviar o esquema criado para o efeito, os metadados propriamente ditos com a informação relevante e ainda outros ficheiros que sejam necessários ou interessantes para o funcionamento da aplicação. Para finalizar esta secção, será necessário alterar a PAT do TS recebido e atualizá-la com a localização dos novos programas e respetivos PIDs. De seguida, é necessário reempacotar tudo e enviar o novo fluxo de vídeo.

Por motivos de tempo, não se conseguiu efetuar testes com sinal de vídeo real, pelo que para efeitos de experimentação, foram apenas usados ficheiros TS locais. É perfeitamente compreensível que para que a solução seja economicamente viável, esse requisito seja imprescindível, contudo esta dissertação visa criar uma prova de conceito pelo que tal funcionalidade não teve a prioridade máxima atribuída inicialmente.

3.5.3.1 Gestão dos dados

Uma vez armazenada a informação proveniente dos sensores, é necessário decidir qual o tratamento a dar esses dados para a sua posterior transmissão em conjunto com o sinal de televisão. De forma a tornar todo o processo o mais automático possível, foi desenvolvida uma aplicação que faz o levantamento dos dados armazenados na base de dados do *Broker* e os armazena em variáveis locais, para posteriormente os estruturar em objetos definidos também na aplicação. Isto vai facilitar o manuseamento dos dados para a conversão em metadados e consequentemente no envio da informação no sinal de vídeo.

O código, desenvolvido em Python por uma questão de consistência e coerência com a aplicação desenvolvida anteriormente e homogeneidade do projeto, contém a definição das classes segundo do diagrama que se segue:

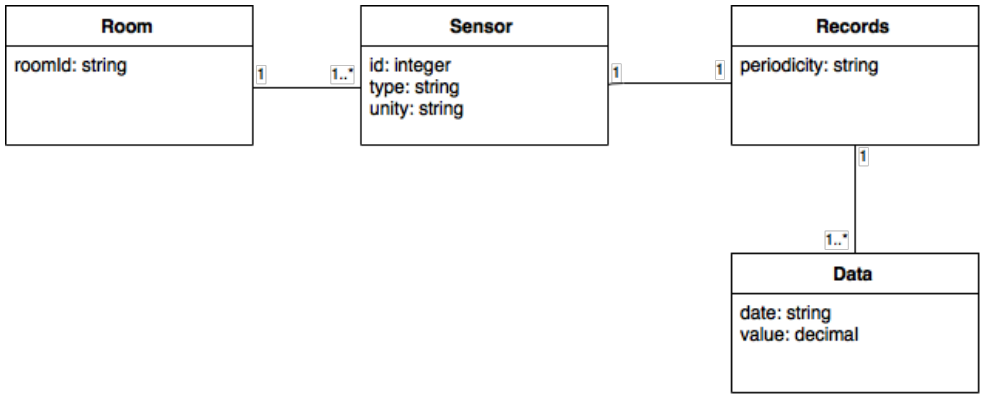

Figura 3.11: Diagrama de classes do módulo Python

Para efeitos da prova de conceito, e porque não faz parte do âmbito da dissertação a adaptação da solução a um contexto real de utilização numa casa inteligente, a adição de novos sensores, bem como a sua remoção, apesar de planeadas, não foram implementadas. Desta forma os sensores a ser utilizados estão definidos no código (*hardcoded*) e as chamadas à API do *Broker* são feitas especificamente para cada um deles para atualizar os respetivos objetos *Sensor* e todas as informações associadas.

Essas informações, tal como consta no diagrama de classes acima demonstrado, todos os registos de informação são armazenados num objeto *Record*, que está diretamente associado a um sensor, de forma a que um sensor possa ter inúmeros registos e cada registo pertença a um só sensor. Com isto, é garantida a integridade e independência das várias medições ao longo do tempo. À semelhança dos sensores, mas numa escala completamente diferente, também as várias instâncias *Record* são armazenadas num *Vector* que posteriormente será corrido e após um processo de transformação, será convertido no formato de metadados desenvolvido e enunciado no capítulo 3.4.

3.5.3.2 Manipulação/envio do sinal

Uma vez convertidos os dados no formato de metadados, é necessário tratar do sinal de televisão para lhe incrementar a informação adicional recolhida nos sensores. O método a utilizar, tal como foi referido anteriormente, consiste na análise do TS do sinal televisivo e a respetiva reconstrução, incluindo no mesmo a aplicação HbbTV que é suposto ser enviada e os metadados num programa adicional que é adicionado ao fluxo do sinal multimédia, cuja PAT é respetivamente atualizada. Tal como tinha sido previsto inicialmente, devido a falta de equipamento próprio, não foi possível conceber uma solução que fosse capaz de obter o sinal de televisão real para de seguida o retransmitir para o televisor, pelo que houve a necessidade de reajustar o projeto para que o conteúdo a enviar fosse gerado diretamente pelo dispositivo responsável pela transmissão do sinal. Note-se que esta alteração foi prevista inicialmente, pelo que em nada foram comprometidos os objetivos desta dissertação.

Para efetuar a transmissão dos dados, foi utilizado o equipamento UT-100C da HiDes, Inc. [HiD15] que é um modulador de sinal televisivo compatível com as normas MPEG e DVB e que se liga ao computador via USB, porta pela qual é também alimentado.

Figura 3.12: Modulador UT-100C

Juntamente com a aplicação *Opencaster* [Srl13], é possível criar, modificar e enviar conteúdo multimédia compatível com as normas DVB de um computador para um ou vários televisores. A Avalpa, empresa que desenvolveu o *Opencaster*, disponibiliza um SDK[3] para utilizar o modulador, providenciando funções capazes de alterar as propriedades do TS que vai ser enviado.

Desta forma, foi desenvolvido um *script bash*[4] que é usado para automatizar o processo de transmissão do TS, utilizando os comandos disponibilizados para gerar e multiplexar[5] o TS, assim como efetuar o envio da aplicação HbbTV como um objeto DSM-CC, tal como foi especificado previamente. Para além disso, foi ainda desenvolvido um módulo *Python* que define as tabelas e secções que vão constituir o TS aquando da sua geração, que são modificadas segundo a documentação oficial do *Opencaster* para cumprir os propósitos da aplicação a desenvolver.

O procedimento é então constituído pelas seguintes etapas:

1. No primeiro passo é necessário criar o ficheiro TS com a aplicação onde vai estar o objeto *Carousel*. Para isso usa-se a ferramenta *oc-update.sh* que usa a pasta onde se encontra a aplicação interativa a usar e cria um ficheiro *.ts com o mesmo nome.

2. De seguida é necessário criar as diferentes tabelas que constituem o fluxo de informação que vai ser enviada. De acordo com as normas DVB, o TS tem que conter obrigatoriamente as seguintes tabelas: PAT, PMT, SDT e a NIT. Todas estas tabelas e respetivo funcionamento foram referenciados no capítulo 2.2.1. A sua definição é feita no módulo *Python*, de acordo com os moldes definidos no manual que acompanha o *Opencaster* produzido pela Avalpa.

    - A *Program Map Table* é responsável pela informação relativa a cada programa do TS, pelo que se define uma para o programa que vai ser transmitido, definindo ainda os canais de transporte do vídeo, áudio e um específico para o objeto DSM-CC relativo à aplicação.

[3] Software Development Kit
[4] Linha de comandos utilizada nos sistemas Unix
[5] Processo de combinar vários canais de informação num só

- A *Program Association Table* é a tabela primária do TS pelo que nela tem que constar as PMTs correspondentes a cada programa que é transmitido no sinal.
- A *Network Information Table* é utilizada para transmitir informação relativa à rede usada na estrutura do TS. Neste contexto é adicionado simplesmente por motivos formais relacionados com o cumprimento das normas DVB.
- A *Application Information Table* é responsável por conter variáveis importantes relativas à aplicação, tais como protocolo de transporte, versão da aplicação, visibilidade, *auto-run*, prioridade de execução, entre outros
- A *Service Description Table*, tal como a NIT, é criada por uma questão de cumprimento do protocolo, uma vez que nenhuma das funções específicas são definidas no âmbito deste projeto.

3. Uma vez definidas todas as tabelas e respetivos conteúdos, é necessário efetuar o *marshalling*[6] e encapsulamento dessa informação em ficheiros *.ts, um por cada tabela. Para isso é utilizado o comando *sec2ts* que vai encapsular as secções das tabelas, produzidas nos objetos definidos no módulo Python, num ficheiro de TS independente, para posteriormente serem multiplexados.

4. De seguida é necessário criar uma FIFO[7] que é utilizada pelo *Opencaster* para fazer a gestão da ordem dos pacotes do TS a serem transmitidos.

5. A transmissão do sinal fica então dependente da multiplexação de todos os ficheiros gerados num só, passível de ser enviado segundo as normas MPEG-2 e DVB. Utiliza-se então o comando *tscbrmuxer* que recebe os ficheiros *.ts gerados (inclusive o da aplicação interativa gerado inicialmente), os respetivos *bit rates*[8] e uma FIFO criada para então modificar este último com a ordem dos pacotes de TS a serem enviados.

6. Para finalizar, é feito o envio sequencial dos pacotes do TS graças ao comando *tsrfsend*, que utiliza a FIFO populada anteriormente com os conteúdos de vídeo, áudio e da aplicação interativa.

3.5.4 Parte III - Aplicação interativa

O último módulo consiste na criação de uma aplicação para *Smart TVs* que seja capaz de ler os dados que foram previamente injetados no sinal de vídeo. Pretende-se desta forma uma aplicação interativa que visa facilitar a gestão das tarefas domésticas pelos utilizadores, pelo que é suposto que a aplicação seja visualmente simples mas apelativa e que permita ao utilizador visualizar os dados como e quando lhe convier. Para satisfazer esta necessidade, a tecnologia que faz mais sentido usar é o HbbTV (capítulo 2.4.2.1) pois está neste momento em expansão e a

[6] Processo de transformação de da representação tangível de um objeto instanciado num formato compatível com a sua transmissão

[7] Estrutura de dados que respeita o método *First In First Out*

[8] Fluxo de transferência de *bits*

ganhar adesão de grande parte do mercado mundial de aplicações interativas para televisão. O facto de usar HTML e Javascript como linguagens de desenvolvimento é também uma vantagem, uma vez que são linguagens abertas com uma enorme aceitação de mercado e da comunidade de programadores.

3.5.4.1 Arquitetura

A arquitetura da solução desenvolvida é modular, reservando a cada componente da aplicação a sua função específica. Desta forma, é possível enumerar os seguintes elementos:

- Módulo de Interface – Este módulo tem como função definir a estrutura da aplicação através do código HTML, assim como o aspeto da mesma com recurso a CSS. Outra importante funcionalidade é a injeção dos metadados recebidos na estrutura previamente definida.

- Módulo de HbbTV – Este módulo é responsável pela interação entre o televisor e a aplicação, tal como da injeção da aplicação no ecrã como um objeto HbbTV. A navegação entre as opções da aplicação é também responsabilidade deste módulo uma vez que este está encarregue de interpretar os botões premidos no comando.

- Módulo de Extração de Metadados – Este módulo está encarregue de recolher a informação enviada como metadados (XML) e fazer a sua conversão para um formato mais fácil de tratar e manusear (JSON).

Para facilitar a compreensão desta estrutura, desenhou-se um diagrama que explicita a relação entre os módulos e ainda o processo de envio da informação com recurso ao DSM-CC.

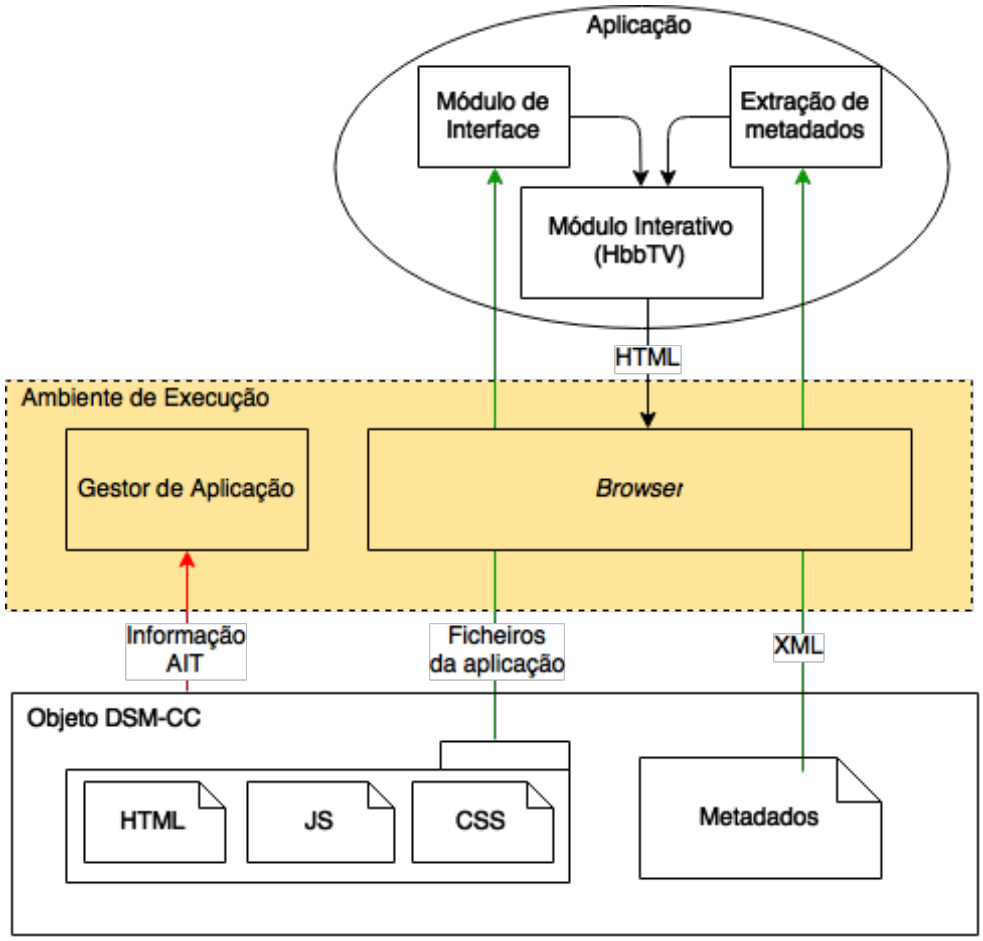

Figura 3.13: Diagrama de relação entre módulos

Nesta imagem está patente o método de transmissão da aplicação HbbTV, assim como os dados dos sensores. Através do DSM-CC, é possível enviar toda esta informação embebida num objeto carrossel que é responsável pelo transporte dos ficheiros HTML, CSS e JS para a aplicação e dos ficheiros XML para os metadados. Para que tal aconteça, na AIT terá de ser especificado qual o método de transporte da aplicação, assim como o caminho da aplicação para que ela possa ser multiplexada em conjunto com o resto da informação.

3.5.4.2 Desenvolvimento e interface

A aplicação HbbTV desenvolvida no âmbito da dissertação utiliza as tecnologias CE-HTML e Javascript que em conjunto permitem criar uma ferramenta tecnológica capaz de cumprir os requisitos definidos inicialmente no que há componente interactiva deste projeto diz respeito. As suas funções primárias são então processar a informação que é enviada sob a forma de metadados (definidos no capítulo 3.4), processá-los e representá-los visualmente no ecrã, duma forma não obstrutiva e complementar ao sinal de vídeo que é enviado. De forma a tornar a aplicação mais robusta e prática, tanto a nível de desenvolvimento como a nível de utilização, usou-se a *framework*

Javascript AngularJS [Goo15] de forma a implementar o padrão de desenvolvimento MVC[9]. Esta alteração permitiu definir componentes que separam virtualmente a maneira como os dados são renderizados, de modo a facilitar o seu manuseamento. Esta divisão, tal como o nome implica, é feita em controladores, vistas e modelos na relação que em baixo de apresenta:

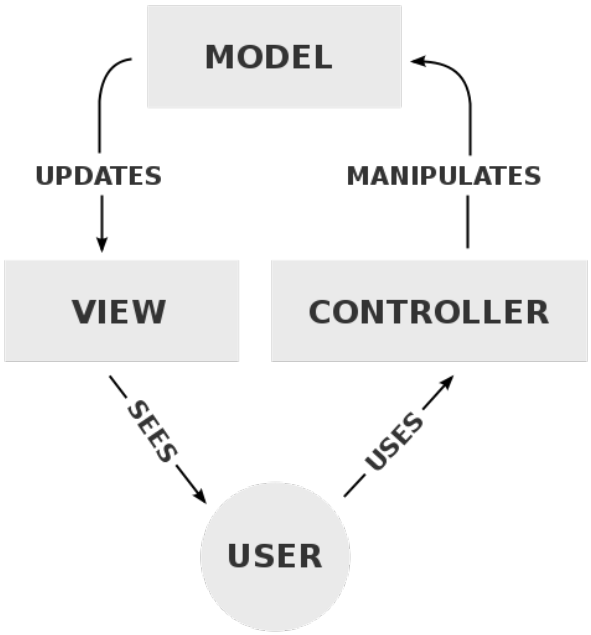

Figura 3.14: Relação entre os componentes no MVC
[Fre10]

A utilização do Angular na prova de conceito teve o intuito de tornar a aplicação mais sólida e facilitar a sua expansão num futuro desenvolvimento. Há imenso potencial no que diz respeito à gestão e manipulação da informação e sem sombra de dúvidas que é pertinente a criação de uma base sólida de apoio ao manuseamento dos dados vindos dos sensores. A nível prático, o Angular serviu como plataforma para a gestão dinâmica do conteúdo da aplicação, sendo usadas as ferramentas de *data-binding* para atualizar automaticamente a informação disponibilizada na vista da aplicação. Esta funcionalidade é deveras interessante pois "liberta" o programador da responsabilidade de manipulação do DOM, dando liberdade para gerir a estrutura da aplicação com maior flexibilidade. Outra vantagem está na já referida utilização do Javascript como linguagem de programação pois confere associada às vantagens da implementação do MVC, todo o potencial dessa linguagem e a facilidade com que é possível produzir, reutilizar e testar código praticamente em tempo real.

Uma das preocupações principais, tal como já foi referido anteriormente, foi em manter a aplicação simples e que, mesmo quando estivesse a ser visualizada, mantivesse o ecrã "limpo de poluição"visual e do excesso de informação que por ventura pudesse surgir derivado da quantidade

[9]Model-View-Controller

de informação passível de ser transmitida. Exatamente com este propósito, foram desenhados alguns *mockups*[10] para tentar avaliar qual a melhor opção para criar uma interface que cumprisse todos os objetivos já definidos. Surgiu o primeiro "rascunho"de como seria a interface:

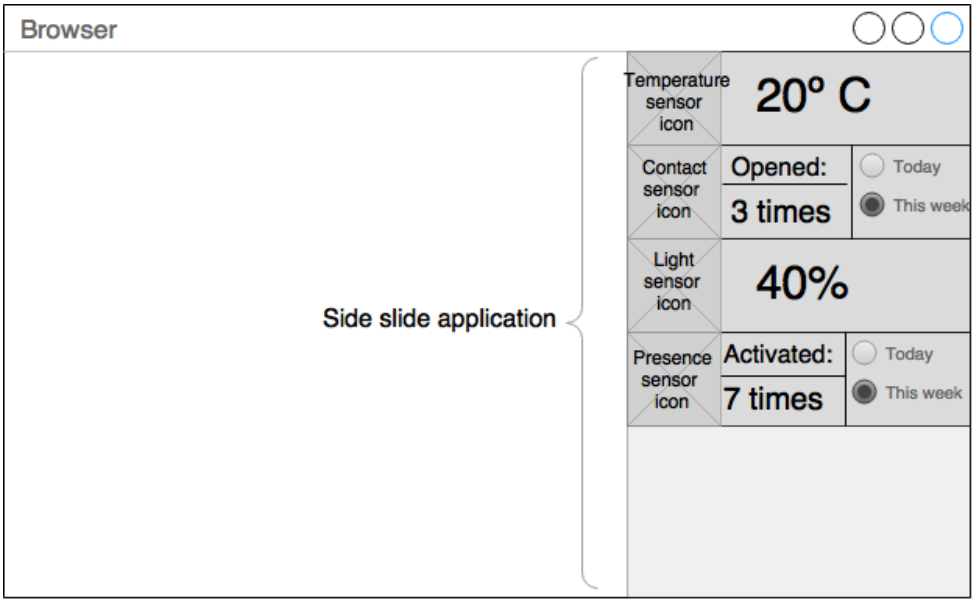

Figura 3.15: *Mockup inicial da interface*

Era de todo essencial que a aplicação fosse navegável recorrendo aos botões do comando, de forma a que o utilizador pudesse percorrer as opções e selecionar qual dos sensores quereria observar em maior detalhe. Houve também uma necessidade importante em transmitir a informação graficamente, uma vez que esta é muito mais simples de interpretar, assim como torna a aplicação muito mais intuitiva e visualmente apelativa. A ideia já patente no *mockup* inicial da aplicaçao "deslizante"foi baseada no trabalho de Matteo Bicocchi, nomeadamente a sua extensão para a biblioteca JQuery chamada *mb.Extruder* [Bic15]. Com recurso as tecnologias HTML, CSS e Javascript, criou-se um protótipo funcional da aplicação em tudo semelhante ao *mockup*, já com informação dinâmica, animações e a navegação completamente implementada, sendo que no final a aplicação ficou com este aspeto:

[10]Modelo de interface para prototipagem

Figura 3.16: Interface real em execução no *browser*

Os dados relativos aos sensores são passados para a aplicação via JSON após serem convertidos dos metadados (secção 3.4), aproveitando a facilidade que proporciona este método de notação quando usada em conjunto com o Javascript. A já mencionada funcionalidade de *data-binding* foi extremamente útil na definição do *template* da aplicação, possibilitando através do uso de chavetas duplas ({{...}}) a injeção de código Javascript diretamente no código HTML , tal como exemplifica o excerto que se segue:

```
1 <div ng-if="sensor.type == 'temperature'">
2     <img class="sensor-icon" src="../images/thermometer.svg">
3     <span class="sensor-value">
4
          ↪ {{sensor.records[1].data[sensor.records[1].data.length-1].value}}
          ↪ {{sensor.unity}}
5     </span>
6 </div>
```

Os dados são acedidos diretamente do formato JSON, o que facilita tanto a nível de produção a consulta em qualquer altura de uma enorme quantidade de informação que está sempre devidamente organizada, como ajuda imenso no processo de desenvolvimento, a nível de *debugging*[11].

3.5.4.3 Extração dos metadados

Os metadados que são enviados juntamente com o vídeo, são definidos na linguagem XML, tal como consta no capítulo 3.4. Apesar deste ser um método prático e eficiente para anotar informação, quando é necessário consultar ou manipular esses mesmos dados, o XML tem algo a

[11] Processo de identificação e remoção de erros em *software*

perder quando comparado, por exemplo, ao JSON, nomeadamente quando se está a usar o AngularJS como *framework*. Exatamente por este motivo, quando os metadados são recebidos, eles são convertidos para formato JSON, que posteriormente serão interpretados pela aplicação.

A W3C[12] definiu em 1995 uma restrição denominada de política de mesma origem [WHA15], que consiste no bloqueio ao acesso de recursos de uma página a outra, se ambas não partilharem o domínio. Por este motivo, os metadados têm que ser enviados em conjunto com a aplicação para que possam ter o mesmo domínio. Os metadados são então obtidos com recurso a pedidos AJAX ao ficheiro XML onde estão definidos para serem então convertidos para o formato JSON, com recurso a uma função que transforma as tradicionais etiquetas dos ficheiros de anotação em objetos JSON devidamente estruturados. De forma a simplificar o manuseamento dos metados, foi utilizada uma biblioteca externa de conversão de ficheiros XML para JSON.

3.6 Validação e testes

3.6.1 Validação

3.6.1.1 Validação da solução

Toda a dissertação teve como propósito principal a criação de uma solução interativa que provasse que o conceito já enunciado é exequível. Para tal, é necessário proceder à validação do produto desenvolvido como um todo, garantido que todas as suas componentes funcionam independentemente e que comunicam entre si com vista o cumprimento dos requisitos definidos inicialmente.

Tendo em conta esta situação, foram definidos testes operacionais que procuram averiguar que as funcionalidades enunciadas estão de facto implementadas e que a solução é funcional:

Tabela 3.2: Casos de teste da solução

ID	Descrição
S1	A aplicação chega ao televisor e é executada.
S2	A aplicação é aberta e minimizada quando escolhido.
S3	Os metadados são importados corretamente.
S4	Os metadados são corretamente convertidos.
S5	A aplicação é navegável com o comando da televisão.
S6	A informação é mostrada textualmente e graficamente de forma conveniente.
S7	É possível visualizar um canal televisivo enquanto se utiliza a aplicação.

Os casos de testes foram definidos tendo em conta os requisitos especificados no capítulo 3.2, pelo que é possível fazer uma associação entre os dois, de forma a averiguar quais são cumpridos. A solução inicialmente proposta continha algumas funcionalidades que por falta de equipamento,

[12]World Wide Web Consortium

tempo e por levarem a um desvio do foco da dissertação, acabaram por não ser implementadas. Na tabela seguinte, os casos de uso que estão relacionados com um ou vários casos de teste estão assinalados com um ponto enquanto os casos em que as funcionalidades não podem ser testadas estão assinaladas com uma cruz.

Tabela 3.3: Relação entre requisitos de utilizador e os casos de teste

		Casos de Teste						
		S1	S2	S3	S4	S5	S6	S7
Requisitos de Utilizador	1	•	•					
	2				•	•		
	3				•	•		
	4				•	•		
	5				•		•	
	6							•
	7			•				
	8	×	×	×	×	×	×	×
	9	×	×	×	×	×	×	×
	10	×	×	×	×	×	×	×
	11	×	×	×	×	×	×	×

3.6.1.2 Validação modular

Uma vez que a solução desenvolvida se divide em diversos módulos, faz sentido de igual forma averiguar quais as condições de funcionamento de cada um dos módulos separadamente, facilitando a correção de erros e favorecendo a qualidade individual de cada componente, o que afeta diretamente a globalidade da solução.

Tabela 3.4: Casos de teste individuais

ID	**Descrição**
I1	Os sensores capturam informação de contexto.
I2	A informação é armazenada no *Context Broker*.
I3	Os dados do *Context Broker* são corretamente convertidos em metadados.
I4	Os metadados são enviados em conjunto com o sinal de televisão.
I5	O conteúdo multimédia chega ao televisor.

Sendo assim, a seguinte tabela apresenta a relação entre os casos de testes individuais e os módulos que lhe são afetos. Uma particularidade assenta no facto desta não possuir nada assinalado na terceira coluna, correspondente à aplicação interativa. Isto sucede-se porque a aplicação interativa é o módulo final, o que implica que todos os casos de teste sejam verificados como testes de solução e não testes modulares.

Tabela 3.5: Relação entre os casos de teste individuais e os módulos da solução

		Módulos		
		1 - Rede de Sensores	2 - Manipulação do Vídeo	3 - Aplicação Interativa
Casos de Teste	I1	•		
	I2	•		
	I3		•	
	I4		•	
	I5		•	

3.6.2 Testes

3.6.2.1 Ambiente de testes

A avaliação do desempenho da solução está dependente do funcionamento dos três módulos independentes e da forma como eles interagem. Para que essa avaliação seja eficiente, é necessário criar um ambiente de testes que permita identificar as condições de funcionamento da solução e localizar eventuais erros ou anomalias para que a sua posterior correção ou melhoria seja o mais eficaz possível. Uma das preocupações durante a fase de desenvolvimento era produzir uma solução cujo desempenho se aproximasse o mais possível de um caso real de utilização, possibilitando, com o número mínimo de alterações, o seu lançamento para o mercado de consumo.

Para testar parte da solução relativa aos sensores, foram instalados dois Arduinos munidos de sensores de temperatura, presença, luminosidade e contacto (porta ou janela). Em cada uma das plataformas de prototipagem, foi instalado o *sketch* produzido para a recolha de informação de contexto (capítulo 3.5.2.1), cujo *output* permite averiguar a veracidade dos dados e se assemelha a este exemplo:

```
1 {"contextElements":[{"type": "Sensor","isPattern": "false","id":
  ↪ "Sensor1","attributes": [{"name": "temperature","type":
  ↪ "float","value": "23.9276638000"}]}],"updateAction": "UPDATE"}
```

De seguida a informação é enviada para o *Context Broker* onde é armazenada. Tal como foi referido no capítulo 3.5.2.3, a solução prevê a configuração de uma máquina com o sistema operativo *CentOS* onde é instalado o pacote do *Orion Context Broker*. De forma a avaliar se a informação enviada corresponde à original, procedeu-se à instalação de todo o *software* necessário numa máquina virtual de forma a emular uma máquina real dedicada para o propósito, que alojava o servidor REST ao qual se fazem os pedidos de atualização e *queries* à base de dados. Se a informação coincidir com a enviada anteriormente, a resposta ao pedido de informação terá o mesmo conteúdo com que foi atualizada, tal como consta no seguinte exemplo:

```
1 {
2     "contextResponses": [
```

```
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "temperature",
                        "type": "float",
                        "value": "23.9276638000"
                    }
                ],
                "id": "Sensor1",
                "isPattern": "false",
                "type": "Sensor"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ]
}
```

Os dados provenientes do *Broker* são então convertidos em metadados cuja validação é possível através do ficheiro XSD desenvolvido aquando da criação do perfil de metadados para os sensores (capítulo 3.4). Em condições normais, os dados enviados deverão ser os mesmos que os recolhidos pelos sensores e teriam aproximadamente esta estrutura:

```
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004"
 ↪ xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
 ↪ xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <Description xsi:type="ContentEntityType">
        <MultimediaContent xsi:type="VideoType">
            <Video>
                <Annotations>
                    <Annotation>
                        <room roomId="1">
              <sensor>
                <id>1</id>
                <type>temperature</type>
                <unity>C</unity>
```

```
13                  <records>
14                    <periodicity>weekly</periodicity>
15                    <data>
16                      <date>2015-04-04 20:00:00</date>
17                      <value>24</value>
18                    </data>
19                                        ...
20                </sensor>
21                          </room>
22                          ...
```

A fase seguinte consiste na utilização dos metadados criados anteriormente e no envio dos mesmos juntamente com o sinal de televisão, de forma a que estes sejam reproduzidos na aplicação interativa instalada no aparelho recetor. Nesta fase existem três processos passíveis de serem avaliados: a execução do *template* da aplicação, a transmissão do sinal de televisão e a transmissão e execução da aplicação no televisor. Para todas estas situações foi criado um ambiente onde o sinal seria enviado pela máquina dedicada para o efeito através do modulador UT-100C e seria recebido por uma televisão Samsung UE65JU6400KXXC. Durante a fase de desenvolvimento, a aplicação interativa foi testada no navegador Google Chrome, com o auxílio das ferramentas de programador disponibilizadas para fazer o *debug* e afinar detalhes relativos ao aspeto do *template*.

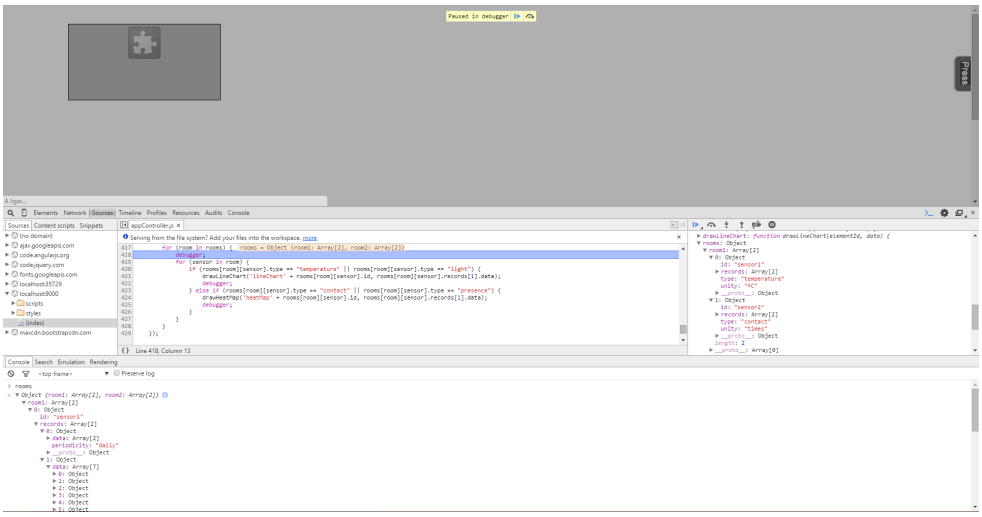

Figura 3.17: Ferramentas de programador do Chrome

3.6.2.2 Resultados

A enunciação dos testes e a criação do ambiente para os executar deram então o mote para a avaliação do desempenho da aplicação. Tal como foi referido na secção 3.6.1, existem dois tipos de testes que visam avaliar a solução como um todo ou cada módulo individualmente. Na

generalidade os testes foram executados com sucesso, com exceção dos testes globais, onde se verificou uma falha na transmissão da aplicação HbbTV e correspondente execução no televisor.

A nível individual, todos os testes modulares foram bem sucedidos, o que permite assumir que a nível estrutural a solução está bem desenvolvida e que a sua granularidade corresponde ao que seria expectável. Os sensores capturam informação de contexto tal como previsto, atualizando o *Context Broker* com a informação que lhe é enviada para o servidor REST. Esses dados são convertidos no formato de metadados com sucesso e são enviados para a aplicação, juntamente com o sinal de vídeo, que por sua vez é transmitido na televisão com áudio e vídeo.

Contudo, no ambiente de testes criado para simular um caso de estudo real, a aplicação não foi tão bem sucedida. A aplicação HbbTV funciona perfeitamente quando executada num *browser* mas quando é enviada para o aparelho recetor, o mesmo já não se sucede. A informação proveniente dos metadados é convertida para JSON sendo que esses dados são lidos corretamente, o que por exclusão de partes remete o problema para o envio da aplicação por parte do modulador utilizado. Foram realizados alguns testes com ficheiros TS fornecidos pela Avalpa juntamente com o *Opencaster* que serviriam para testar o equipamento. Acontece que estes ficheiros também não funcionaram, o que pode significar que o erro está na configuração da televisão e a sua capacidade de suportar HbbTV. O facto dessa funcionalidade vir desativada por defeito e não ser suportada nativamente em todos os países (forçando uma mudança de país nas configurações de televisor) pode ser a origem do problema, pelo que seriam necessários testes mais exaustivos para concluir ao certo qual a origem exata do erro. Infelizmente o equipamento não esteve disponível tão cedo quando desejado pelo que não houve oportunidade de resolver esta situação em tempo útil.

Capítulo 4

Conclusões

Sendo esta dissertação tão multidisciplinar e ao mesmo tempo tão específica em determinados pontos, é normal que os vários temas abordados não sejam tão aprofundados como seria desejável. Desta forma, tentou-se focar nas principais tecnologias e sobretudo nas mais recentes e utilizadas no mercado. O objetivo final da dissertação é produzir uma solução que acrescente algo de valor ao mercado, pelo que faz todo o sentido apostar em tecnologias que são atualmente estado da arte.

O objetivo primário e o resultado mais importante deste projeto é a aplicação interativa para a televisão, o que tornou o foco de praticamente toda a fase de desenvolvimento a interação da solução com o utilizador final. Esta decisão influenciou a forma como foi implementada a solução e de certa forma até alterou a ordem pela qual se desenrolou o desenvolvimento das diferentes partes do projeto. Para isto contribuíram todos os objetivos intermédios de definição de arquitetura e de especificação que visavam estruturar não só a solução a conceber, como o seu próprio método de desenvolvimento.

Inicialmente estava previsto o desenvolvimento da solução no sentido em que "viajam"os dados, ou seja, desde a produção de contexto até à disponibilização da informação no dispositivo recetor. Mas na verdade, o que acabou por acontecer é que no início desenvolveu-se a plataforma de comunicação com os sensores e de seguida passou-se logo para a criação da interface interativa disponível na televisão, passando à frente a a camada intermédia de manipulação do sinal de vídeo. Este facto deveu-se exatamente à necessidade também de desenhar a estrutura de dados à medida da aplicação desenvolvida e ao seu modelo de interatividade, garantido assim que o transporte de informação era o mais completo e eficiente possível.

Houve então necessidade de criar uma rede de sensores baseada nas plataformas Arduino que era responsável pela produção de contexto e fornecer a informação que vai ser tratada e exibida pelos dois outros blocos da solução. Uma das dificuldades sentidas nesta fase de desenvolvimento prendeu-se com a falta de experiência na plataforma Arduino, assim como na falta de bases em eletrónica na assemblagem dos sensores, como se numa rede se tratassem. Houve ainda a adversidade já mencionada relativa à falta do *network shield* que foi contornada da forma já explicada

na secção 3.5.2.2.

A segunda fase era relativa à manipulação de vídeo e exigiu a configuração de uma máquina dedicada para o efeito. O objetivo era a criação de uma aplicação ou de um mecanismo automático que fosse capaz de receber a informação dos sensores e enviá-la juntamente com o sinal de vídeo. Aqui foram introduzidas muitas diversidades, relacionadas com a utilização de *software* criado para funcionar em situações muito características (o que por diversas vezes originou problemas de compatibilidade). O equipamento utilizado na emissão do sinal de vídeo e da transmissão da aplicação HbbTV também criou problemas de compatibilidade e de instalação. A exigência e rigor associado à manipulação das tabelas do TS também foram obstáculos que dificultaram um desenvolvimento mais uniforme e livre de percalços. Os metadados que acompanham o sinal de vídeo também fizeram parte dos objetivos uma vez que a sua correta transmissão estava dependente da criação de um perfil de dados criado de propósito. Este teria de ser capaz de proporcionar uma estrutura de dados adequada, o que se verificou trabalhoso, uma vez que o padrão de metadados MPEG-7 possui uma complexidade e um nível de detalhe muito elevados.

O objetivo final era a criação de uma aplicação interativa para *Smart TVs* capaz de disponibilizar de forma interativa os dados dos sensores aos utilizadores. Esta aplicação, baseada em tecnologias HTML, CSS e Javascript, é responsável pela exibição do *template* da aplicação, tratamento dos metadados e navegação dinâmica dos dados. Uma das dificuldades que surgiu nesta fase foi a adaptação do *template* para a uma aplicação HbbTV, que por ser uma tecnologia relativamente recente e com poucos exemplos públicos, foi difícil de testar e de comprovar o seu funcionamento.

4.1 Satisfação dos objetivos

A satisfação dos objetivos está intimamente relacionada com o sucesso na execução dos testes aos quais a solução desenvolvida foi submetida. Foram criados vários testes de forma a avaliar qual o desempenho da solução nos seus diversos componentes.

Os testes foram realizados de forma a simular um ambiente real, pelo que estão criadas as condições para uma eventual demonstração ao vivo. É desta forma possível personalizar os conteúdos a emitir e ainda alterar o contexto dos sensores que vai ser enviado para a aplicação, uma vez que a solução faz a gestão automática toda essa informação.

Uma vez que os testes não foram conclusivos, não foi possível garantir que todo o fluxo da solução é ótimo. Contudo, os testes modulares permitem assumir que a arquitetura definida é capaz de garantir a exequibilidade da solução, sendo empiricamente possível garantir a viabilidade técnica de uma solução semelhante (com ligeiras alterações/melhoramentos) que possa acrescentar valor a um nicho de mercado que se encontra fortemente sub-desenvolvido.

Sendo a área de desenvolvimento desta dissertação muito específica e limitada, a criação de uma aplicação que permita expandir os serviços disponibilizados nas *Smart TVs* é uma adição importante a um nicho de mercado que necessita claramente deste tipo de impulsos para crescer

e tornar-se parte integrante do mercado de ofertas neste tipo de serviço interativo. É então possível concluir que a solução desenvolvida teria uma elevada aplicabilidade, isto porque o custo de implementação desta aplicação é relativamente baixo, apoiando-se em tecnologias amplamente utilizadas e justificando assim o investimento em tendências que mostram estar cada vez mais a ganhar aceitação do público.

Uma das preocupações principais durante o desenvolvimento da solução é garantir a escalabilidade e flexibilidade da solução, o que se refletiu no produto final. É importante para o sucesso da aplicação que esta seja expansível a nível de funcionalidades e que a sua integração com as tecnologias que atualmente existem no setor das *smart homes* seja facilitado pela criação de uma estrutura sólida e fundado em tecnologias abertas, de modo a facilitar um futuro desenvolvimento.

4.2 Melhorias futuras

Um dos pontos mais focados na secção de desenvolvimento foi que devido à implementação ser uma prova de conceito, algumas das funcionalidades que estavam inicialmente previstas não puderam ser executadas como seria desejável. Tal facto é compreensível pelos motivos já referidos, pelo que se houvesse mais tempo para dedicar a este projeto, uma das melhorias seria implementar as funcionalidades de alertas, gestão de sensores e gestão de histórico de dados dos sensores que não ficaram completas.

Outra das melhorias possíveis seria a otimização da aplicação pois como esta se trata de uma aplicação baseada em HTML, seria conveniente reduzir os tempos de carga, minimizar as latências em transferências e reduzir o tráfego gerado pelo uso da aplicação. Algumas medidas que poderiam contribuir para a otimização seriam refatoração[1] de código, minimização de código e utilização de memória *cache* de forma a gerir a validade dos conteúdos após serem carregados, evitando tráfego desnecessário.

Uma das principais falhas no HbbTV é incapacidade de reiniciar a aplicação através de um pedido remoto, o que de certa forma se incompatibiliza com a maneira como a aplicação carrega a informação dos metadados. A especificação mais recente do HbbTV [Ass15] supostamente corrige esta situação mas como os testes realizados não permitiram testar a aplicação num ambiente real, não é possível concluir se este problema estaria 100% resolvido.

[1] Modificação/otimização de código sem alterar o seu funcionamento

Referências

[AB13] Marylin Arndt e Fano Ramparany Alia Bellabas. New promising service infrastructure for smart iot applications. In *International Congress on 3D IT, Communications, and Convergence*, pages 1017–1019, 2013.

[Ard15a] Arduino. Arduino - faq. Arduino, disponível em `http://www.arduino.cc/en/Main/FAQ`, acedido a última vez a 19 de Junho de 2015, 2015.

[Ard15b] Arduino. Arduino - introduction. Arduino, disponível em `https://www.arduino.cc/en/Guide/Introduction`, acedido a última vez a 19 de Junho de 2015, 2015.

[Ard15c] Arduino. Arduino - software. Arduino, disponível em `https://www.arduino.cc/en/Main/Software`, acedido a última vez a 19 de Junho de 2015, 2015.

[Ard15d] Arduino. Arduino development environment. Arduino, disponível em `http://www.arduino.cc/en/Guide/Environment`, acedido a última vez a 19 de Junho de 2015, 2015.

[Ass15] HbbTV Association. Hbbtv 2.0 specification. HbbTV Association, disponível em `https://www.hbbtv.org/pages/about_hbbtv/HbbTV_specification_2_0.pdf`, acedido a última vez a 19 de Junho de 2015, 2015.

[Bic15] Matteo Bicocchi. Jquery mb.extruder. Pupunzi, disponível em `http://pupunzi.open-lab.com/mb-jquery-components/jquery-mb-extruder`, acedido a última vez a 19 de Junho de 2015, 2015.

[eLH13] Mohammed El-Hajjar e Lajos Hanzo. A survey of digital television broadcast transmission techniques. *IEEE Communications Surveys & Tutorials*, 15(4), Quarto Trimestre 2013.

[eORR14] Boris Moltchanov e Oscar Rodríguez Rocha. A context broker to enable future iot applications and services. In *6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 263–268, 2014.

[eRA05] Alexsandro Paes e Renato Antoniazzi. Padrões de middleware para tv digital. *CEP*, 24210:240, 2005.

[Fel14] Mark Fell. Roadmap for the emerging internet of things. Carré & Strauss, disponível em `http://carre-strauss.com/documents/IoT_Roadmap.pdf`, 2014.

[Fer10] Pedro Ferreira. Mxf – a progress report. *EBU Technical Review*, Terceiro Trimestre 2010.

REFERÊNCIAS

[Fre10] Regis Frey. The model, view, and controller (mvc) pattern relative to the user. Wikimedia Commons, disponível em https://commons.wikimedia.org/wiki/File:MVC-Process.svg, acedido a última vez a 19 de Junho de 2015, 2010.

[Gal91] Didier Le Gall. Mpeg: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4), Abril 1991.

[Gal15] Fermin Galan. Publish/subscribe broker - orion context broker - installation and administration guide. Telefónica Investigación y Desarrollo, disponível em https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Publish/Subscribe_Broker_-_Orion_Context_Broker_-_Installation_and_Administration_Guide, acedido a última vez a 19 de Junho de 2015, 2015.

[Goo15] Google. Angularjs. Google, disponível em http://angularjs.org, acedido a última vez a 19 de Junho de 2015, 2015.

[HiD15] Inc. HiDes. Ut-100c opencaster special edition (tx only). HiDes, Inc., disponível em http://www.hides.com.tw/product_opencaster_eng.html, acedido a última vez a 19 de Junho de 2015, 2015.

[IB03] Keith Hill Jan Borgmans e Fernando Pereira Ian Burnett, Rik Van de Walle. Mpeg-21: goals and achievements. *IEEE Multimedia*, 10(4):60–70, 2003.

[Jen08] Jens F. Jensen. *Changing Television Environments*. Springer Berlin Heidelberg, 2008.

[Kov10] Steve Kovach. What is a smart tv? Business Insider, disponível em http://www.businessinsider.com/what-is-a-smart-tv-2010-12, acedido a última vez em 14 de Fevereiro de 2015, Dezembro 2010.

[Lev10] Carmi Levy. Future of television is online and on-demand. Toronto Star, disponível em http://www.thestar.com/business/2010/10/15/future_of_television_is_online_and_ondemand.html, acedido a última vez em 15 de Fevereiro de 2015, Outubro 2010.

[Mer11] Klaus Merkel. Hybrid broadcast broadband tv, the new way to a comprehensive tv experience. In *Electronic Media Technology (CEMT), 2011 14th ITG Conference*, pages 1–4, 2011.

[Mor11] Steven Morris. How to become an expert in dsm-cc. Disponível em http://www.interactivetvweb.org/tutorials/dtv_intro/dsmcc/, acedido a última vez a 14 de Fevereiro de 2015, 2011.

[Pie06] Jon Piesing. The dvb multimedia home platform (mhp) and related specifications. In *Proceedings of the IEEE*, volume 94, pages 237–247, 2006.

[Rou14] Margaret Rouse. Internet of things (iot). Disponível em http://whatis.techtarget.com/definition/Internet-of-Things, acedido a última vez em 14 de Fevereiro de 2015, Junho 2014.

[SFC01] Thomas Sikora e Atul Puri Shih-Fu Chang. Overview of the mpeg-7 standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6), Junho 2001.

[Sof15] Ocean Blue Software. Mheg-5. Ocean Blue Software, disponível em `http://www.oceanbluesoftware.com/mheg5/`, acedido a última vez a 15 de Fevereiro de 2015, 2015.

[Srl13] Avalpa Digital Engineering Srl. Opencaster 3.2.2: the free digital tv software. Avalpa Digital Engineering Srl., disponível em `http://www.avalpa.com/the-key-values/15-free-software/33-opencaster`, acedido a última vez a 19 de Junho de 2015, 2013.

[SS06] John R. Smith e Peter Schirling. Metadata standards roundup. *IEEE MultiMedia*, 13(2):84–88, Abril-Junho 2006.

[Tec] Technicolor. Sdks in action. Technicolor QEO, disponível em `http://www.qeo-app-development.com/member/sdk-in-action`, acedido a última vez a 14 de Fevereiro de 2015.

[Tek00] Tektronix. *A Guide to MPEG Fundamentals and Protocol Analysis*. Tektronix, Third edition, 2000.

[Urb11] Leon Urban. Mpeg 101 - transport stream demystification. In *Conference for Caribbean and Cable Telecommunications Association members*. Triveni Digital, 2011.

[WHA15] WHATWG. Origin. Web Hypertext Application Technology Working Group, disponível em `https://html.spec.whatwg.org/multipage/browsers.html#origin`, acedido a última vez a 19 de Junho de 2015, 2015.

REFERÊNCIAS

Anexo A

Metadados

Este anexo apresenta o ficheiro XML relativo à definição dos metadados.

A.1 Extensão do esquema MPEG-7 e perfil dos sensores

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <schema xmlns="http://www.w3.org/2001/XMLSchema"
3    xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
4    targetNamespace="urn:mpeg:mpeg7:schema:2004"
5    elementFormDefault="qualified" attributeFormDefault="unqualified">
6      <import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation="http:
           //www.w3.org/2001/03/xml.xsd"/>
7    <include schemaLocation="mpeg7-v3.xsd"/>
8
9    <!-- Redefine schema-->
10   <!-- Add Annotations-->
11   <redefine schemaLocation="mpeg7-v3.xsd">
12     <complexType name="VideoSegmentType">
13       <complexContent>
14         <extension base="mpeg7:VideoSegmentType">
15           <sequence>
16             <element name="Annotations" type="mpeg7:Annotations" minOccurs="1"
                   maxOccurs="1"/>
17           </sequence>
18         </extension>
19       </complexContent>
20     </complexType>
21   </redefine>
22
23   <!-- Extend TermUse-->
24   <!-- Add sensor group-->
25   <complexType name="Annotations">
26     <complexContent>
```

```
27      <extension base="mpeg7:TermUseType">
28        <sequence>
29          <element name="Annotation" type="mpeg7:sensorMetadataGroup" minOccurs="0"
                maxOccurs="unbounded"/>
30        </sequence>
31      </extension>
32    </complexContent>
33  </complexType>
34
35  <!-- Create sensor group-->
36  <complexType name="sensorMetadataGroup">
37    <sequence>
38      <element name="room" type="mpeg7:room" minOccurs="0" maxOccurs="unbounded"/>
39    </sequence>
40  </complexType>
41
42  <!-- Create room type-->
43  <complexType name="room">
44    <sequence>
45      <element name="sensor" type="mpeg7:sensor" minOccurs="1" maxOccurs="unbounded
            "/>
46    </sequence>
47    <attribute name="roomId" use="required" type="string"/>
48  </complexType>
49
50  <!-- Create sensor type-->
51  <complexType name="sensor">
52    <sequence>
53      <element name="id" type="integer" minOccurs="1" maxOccurs="1"/>
54      <element name="type" type="string" minOccurs="1" maxOccurs="1"/>
55      <element name="unity" type="string" minOccurs="1" maxOccurs="1"/>
56      <element name="records" type="mpeg7:records" minOccurs="1" maxOccurs="1"/>
57    </sequence>
58  </complexType>
59
60  <!-- Create records type-->
61  <complexType name="records">
62    <sequence>
63      <element name="periodicity" type="string" minOccurs="1" maxOccurs="1"/>
64      <element name="data" type="mpeg7:data" minOccurs="1" maxOccurs="unbounded"/>
65    </sequence>
66  </complexType>
67
68  <!-- Create data type-->
69  <complexType name="data">
70    <sequence>
71      <element name="date" type="string" minOccurs="1" maxOccurs="1"/>
72      <element name="value" type="decimal" minOccurs="1" maxOccurs="1" />
73    </sequence>
```

```
74    </complexType>
75
76  </schema>
```