Faculdade de Engenharia da Universidade do Porto
Mestrado em Tecnologia Multimédia

VISÃOWEB – VIGILÂNCIA E CONTROLO REMOTO

António André Jácome Pereira da Silva

Licenciado em Engenharia Electrotecnia e Computadores
pela Faculdade de Engenharia da Universidade do Porto

Dissertação realizada sob a supervisão do

Professor Doutor Eurico Carrapatoso

do Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

Porto – Maio de 2005

Agradecimentos

Dedico esta dissertação à Ândrea, a qual me incentivou a percorrer esta aventura e me acompanhou sempre nos momentos mais difíceis.

Agradeço ao Carlos Silva e Miguel Costa, meus sócios da empresa CentralCasa e grandes amigos de faculdade, que me ajudaram a tornar o projecto VisãoWeb uma realidade.

Pela ajuda e desenvolvimento da programação, agradeço à equipa do CentralCasa, em especial ao Eng. Vítor Pinto. Por todo o apoio no design da aplicação não posso deixar de agradecer à equipa da PontoPR, nomeadamente ao Miguel Silva.

Agradeço ao Eng. Eurico Carrapatoso, a minha "consciência", que sempre me apoiou no desenvolvimento de todo o trabalho.

Por último, agradeço à minha família, sem os quais nada disto seria possível.

Resumo

A revolução da Internet, no final da década passada, transformou as nossas habitações em autênticos lares digitais, ligando-as ao mundo através das redes IP. Esta mudança veio potenciar a domótica – automação doméstica, que, apesar de já existir há duas décadas, só agora começa a dar os primeiros passos. Actualmente, existem já alguns promotores imobiliários que apostam em equipar os imóveis com dispositivos inteligentes que comunicam e interagem entre si.

É inegável o enorme papel desempenhado pelos telemóveis na nossa vida quotidiana. Estes conferem-nos uma mobilidade nunca antes atingida, permitindo-nos estar presentes em vários locais ao mesmo tempo.

Pretendia-se para esta dissertação a definição, o planeamento e o desenvolvimento de um sistema distribuído de vigilância e controlo remoto. Esta solução devia ser orientada tanto para o mercado doméstico como também devia dispor de funcionalidades que a tornassem atractiva para o sector empresarial.

Neste documento são abordados inicialmente, alguns dos principais protocolos de domótica, com especial destaque para o X10 e EIB, seguindo-se uma breve análise às actuais tecnologias de programação distribuída, incluindo o recente .NET. Antes de se iniciar a definição dos requisitos e da arquitectura da aplicação, são ainda analisados os pontos fortes e fracos das actuais soluções de vigilância e de controlo de redes de domótica.

No levantamento dos requisitos são apresentados vários diagramas e *Uses Cases* usando modelos UML. É feita uma apresentação de toda análise e especificação do problema proposto e definida uma arquitectura de hardware para a implementação desta solução num ambiente real. A implementação do sistema foi efectuada com recurso à linguagem C#, da Microsoft, sobre a plataforma .NET. Foi também projectada e implementada uma base de dados em MS SQL Server 2000 e desenvolvida a comunicação com uma série de dispositivo pré-escolhidos.

Na parte final do projecto, o serviço foi colocado em funcionamento e testado com um grupo de clientes-piloto. Foi feita uma análise para avaliar o comportamento e os hábitos dos utilizadores finais e avaliar o desempenho do sistema. Foram também efectuados testes de stress, simulando o acesso de 1000 utilizadores em simultâneo.

Palavras Chave: Domótica, Casas Inteligentes, X10, EIB, vigilância, segurança, mobilidade, Internet, sistemas distribuídos

Abstract

The Internet revolution, over past decade, brought the digital world to our houses, enabling them to be viewed and controlled from every place, at every time. Along with this evolution, the Home Automation market, even being in its early stage, is growing. Nowadays there are some constructors which already started offering houses equipped with alarm systems and intelligent devices which can communicate and interact.

It's also undeniable the importance of the mobile communication in our lives, allowing us to reach the long awaited mobility, increasing the need to be present in different places at the same time.

The main goal of this work was the definition, design and development of a distributed system that allows the remote control and surveillance. This solution should be targeted to the residential houses, but should have more advanced functionalities allowing it to have a more professional use.

In this document we make an initial approach to some of the home automation protocols, with special attention to X10 and EIB, followed by a brief analysis of the most recent distributed programming languages, including the Microsoft .NET platform. Before laying down the main requirements for the application, some of the available products in the market are analysed, namely their strong and weak sides.

In the requirement definition some diagrams and *Use Cases* are presented, using the UML modelling language. These diagrams are used to create the software and hardware architectures so that they may be deployed in a real world scenario The implementation is done mainly using the C# language over the .NET platform A data base, using a MS SQL Server 2000, was also designed and developped as well as the communication module to interact with all the remote devices (camera, alarm, devices, etc.).

In the final stage we deployed and tested the application with a limited number of clients. We could then analyse the behaviour and habits of end-users and evaluate the overall system performance. There are also made some stress tests, simulating the use of 1000 simultaneous users.

Keywords: Home automation, X10, EIB, surveillance, security, mobile, Internet, distributed applications

Extracto

La revolución de Internet, en la última década, ha introducido el mundo digital en nuestras casas, permitiendo ver y controlar desde cualquier lugar o cualquier hora. Junto con esta evolución, la domótica, a pesar de surgir hace mucho tiempo, es ahora cuando se empiezan a dar los primeros pasos. En la actualidad, algunos constructores empiezan ya a ofrecer las casas equipadas con los sistemas de alarma y con dispositivos inteligentes con los que poderse comunicar.

Es también innegable la importancia de la comunicación móvil en nuestras vidas, permitiendo que alcancemos una gran libertad como consecuencia de la necesidad de estar presente en diversos lugares al mismo tiempo.

La meta principal de este trabajo es la definición, el planeamiento y el desarrollo de un sistema distribuido que permita el telecontrol y la vigilancia. Esta solución debe estar orientada a casas residenciales, pero también debe tener funciones avanzadas que permitan un uso más profesional.

En este documento, hacemos un acercamiento inicial a algunos de los protocolos más comunes de automatización, prestando especial atención a X10 y a EIB, siguiendo con una comparación de los lenguajes de programación distribuidos más recientes, incluyendo la plataforma .NET de Microsoft. Antes de colocar los requisitos principales de uso, algunos de los productos disponibles en el mercado, se prueban y se estudian sus lados fuertes y débiles.

En la definición de los requisitos se presentan algunos diagramas y utilizan casos, usando el UML. Estos diagramas se utilizan para crear el software y la arquitectura de hardware con la intención de desplegarla en un panorama del mundo real, la puesta en práctica se hace principalmente usando el lenguaje C# sobre la plataforma de .NET. También se proyecta y despliega una base de datos, usando un servidor MS SQL 2000, y el módulo de comunicación para trabajar recíprocamente con todos los dispositivos remotos (cámara fotográfica, alarma, dispositivos, etc.).

En la etapa final del proyecto, el servicio ha sido puesto en funcionamiento y probado con un número limitado de clientes. Estos análisis permiten que estudiemos el comportamiento y el hábito de los usuarios finales y que evaluemos el funcionamiento total. También se ha hecho algunas pruebas de tensión, simulando el uso de 1000 usuarios simultáneos.

Palabras claves: automatización residencial, X10, EIB, vigilancia, seguridad, móvil, Internet

Índice

Índice de figuras

1. INTRODUÇÃO

A evolução do serviço de Internet, através da massificação dos serviços de ADSL e de Internet por cabo, assim como o aumento das capacidades das tecnologias móveis, com o amadurecimento do GPRS e aparecimento da 3G, têm aumentado a nossa capacidade de mobilidade. Esta, por sua vez, cria-nos uma necessidade de estar sempre presente, de ver o que se passa no escritório quando estamos em casa, de visualizar e controlar o que se passa na nossa casa quando estamos a trabalhar.

É nesta realidade actual, onde a vídeo-conferência ainda está longe de ter um uso generalizado, a terceira geração de telemóveis está finalmente a dar os seus primeiros passos comerciais e o interesse do público sobre casas inteligentes começa a despertar, que se enquadra a presente tese.

1.1. MOTIVAÇÃO

É inevitável que a habitação como a conhecemos hoje, onde cada dispositivo electrónico desempenha isoladamente a sua função, em vez de comunicar e partilhar recursos com outros aparelhos, venha a evoluir para uma casa inteligente.

Nos últimos três anos, tem-se verificado o aparecimento no mercado nacional, de diversas soluções e projectos na área das casas inteligentes, salientando-se o projecto da casa do futuro interactiva [CasaFuturo, 04]. Este projecto pretende recriar aquela que será a casa de todos nós num futuro muito próximo. Um espaço que reúne, num só ambiente, grande parte das novas tecnologias de ponta e que pretende ser um verdadeiro espaço multimédia e de *home entertainment*.

Se à evolução do conceito da domótica associarmos a crescente procura por melhores serviços de dados, tanto fixos como móveis, torna-se cada vez mais claro que será necessário aproximar os utilizadores das suas casas ou escritórios. Esta nova abordagem na forma de interagir com a habitação, quer local, quer remotamente, foi claramente um motivo de interesse que levou à elaboração desta tese.

A empresa CentralCasa – Desenvolvimento de soluções de domótica, Lda, desde a sua criação, no ano de 2002, com o intuito de desenvolver, fornecer e instalar soluções de domótica em habitações uni-familiares, sentiu uma grande procura por parte dos seus clientes de sistemas para visualizar e gerir as suas casas, não sendo pois de estranhar que foi com grande interesse e expectativa que a empresa acolheu o desenvolvimento deste projecto.

1.2. OBJECTIVOS

O objectivo principal do trabalho descrito nesta dissertação é o estudo, projecto, implementação e teste de uma aplicação distribuída orientada por objectos, que permitisse a visualização e o controlo domótico de um espaço habitacional ou comercial. Através de um qualquer computador ou dispositivo móvel com acesso à Internet, pretende-se que um utilizador possa visualizar e controlar a sua loja ou habitação. A aplicação desenvolvida deve ser executada nos servidores da CentralCasa, sendo disponibilizada como um serviço web, acessível através de um computador com acesso à Internet ou de um dispositivo móvel (telemóvel, PDA, ou outro) com acesso à Internet e ecrã policromático.

Para permitir a vigilância é fundamental instalar no local a vigiar uma ou mais

câmaras, sendo necessário também a instalação de dispositivos domóticos para o seu controlo remoto.

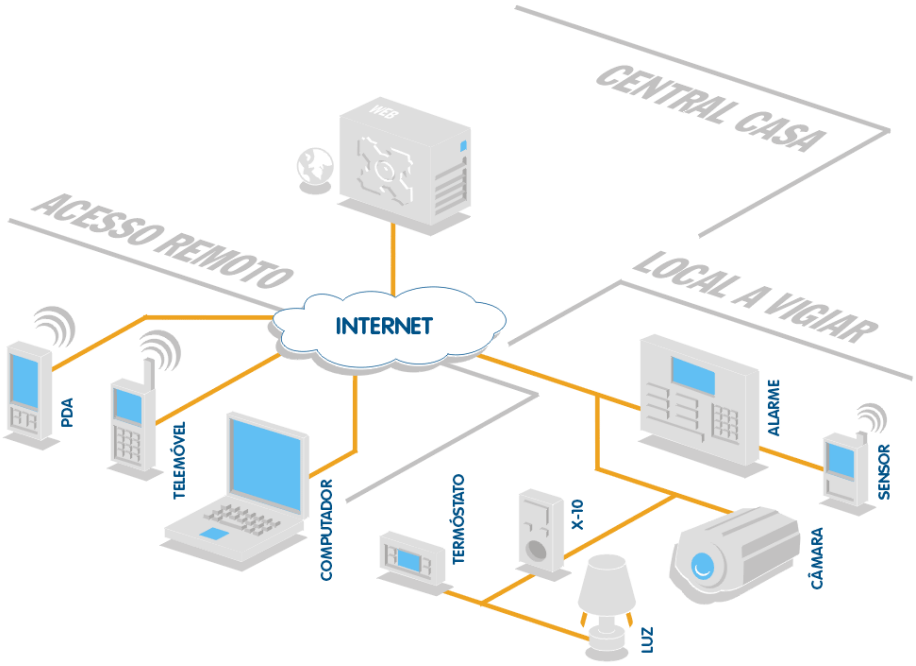

Figura 1.1 - Arquitectura física da aplicação

Os três principais pilares tecnológicos necessários para sustentar o desenvolvimento de uma aplicação que permita ver e controlar remotamente um espaço, são: a vídeo-vigilância, a domótica e a Internet.

A vídeo-vigilância é uma tecnologia à qual já estamos habituados. Usada com o argumento de proteger os utilizadores, já são poucos os espaços públicos e comercias que não disponham de um circuito de vídeo fechado para o controlo de furtos. Esta tecnologia já tem grande aceitação por parte do público que viu ser reduzida a sua privacidade em troca de mais de segurança. Mais recentemente, esta tecnologia está a ser empregue em sistemas para vigiar recém-nascidos, bebés, pessoas idosas ou com incapacidades físicas. Esta última variante utiliza normalmente linhas telefónicas para a transmissão dos sinais de vídeo e áudio para locais remotos.

Apesar de já existir há 2 décadas, a domótica (vulgarmente conhecida pelo termo casas inteligentes), ainda se encontra a dar os primeiros passos, sendo a sua divulgação junto do público ainda muito precária. O termo domótica resulta da junção da palavra *domus* (casa) com telemática (electrónica+informática). São estes dois últimos elementos que, quando utilizados em conjunto, potenciam o

sistema, simplificando a vida diária das pessoas, satisfazendo as suas necessidades de comunicação, de conforto e segurança. Quando a domótica surgiu (com os primeiros edifícios inteligentes, nos anos 80) pretendia-se para além de controlar a iluminação, condições climáticas e a segurança, a interligação entre os três elementos. Apesar de ainda ser pouco conhecida e divulgada, mas pelo conforto e comodidade que pode proporcionar, a domótica promete vir a ter muitos adeptos.

A utilização da domótica pode apresentar inúmeras vantagens, das quais destacamos as seguintes:

- Segurança – permite a detecção e alarme em caso de intrusão, a dissuasão através de sirene de alarme ou de sistema de luzes, a marcação automática após um alarme de um número de telefone exterior, a simulação da presença dos moradores através do controlo de dispositivos eléctricos, o fecho do circuito de água ou gás após a detecção de uma inundação, incêndio ou fuga de gás;
- Gestão energética – optimização do consumo de energia através da programação do aquecimento ou da climatização, tanto no interior como no exterior, selecção das zonas a aquecer consoante os períodos de ocupação, controlo de dispositivos eléctricos, tais como máquinas de lavar ou sistemas de iluminação, de acordo as presenças numa divisão e dos horários de baixa taxação (com evidentes vantagens, quer para consumidores, quer para fornecedores);
- Conforto – programação de macros de acordo preferências pessoais, controlo de luzes de acordo com a luminosidade, programação de rega automática, controlo de aquecimento, de electrodomésticos, de estores, etc.

A Internet permite a convergência de ambas as tecnologias num ambiente distribuído, passível de ser usada em qualquer parte do mundo.

1.3. ESTRUTURA DA DISSERTAÇÃO

A presente dissertação é constituída por 8 capítulos mas a sua estrutura pode ser dividida em cinco partes: uma introdução, dois capítulos sobre as tecnologias analisadas e soluções actuais, dois sobre a aplicação desenvolvida e um último com as conclusões e perspectivas futuras.

Neste primeiro capítulo dão-se a conhecer as razões, a motivação e os objectivos que se pretendem alcançar com esta dissertação e a estrutura da mesma.

No segundo capítulo pretende-se fazer uma abordagem geral sobre os protocolos de domótica existentes os quais permitam, através de um dispositivo central com possibilidade de ligação a uma rede TCP/IP, o controlo de dispositivos eléctricos. Nesta análise dá-se um maior destaque às tecnologias X10 e EIB – *European Installation Bus*, pois são estas que de momento se encontram em forte expansão.

No capítulo seguinte é feita uma análise das tecnologias de programação em ambientes distribuídos, orientadas por objectos, e da plataforma tecnológica onde a aplicação deverá ficar alojada. Serão também analisados os pontos fortes e fracos das soluções actuais na área de vigilância e controlo remoto. Esta análise permitirá, no capítulo quarto, efectuar um levantamento mais rigoroso dos requisitos da aplicação. Estes serão processados, usando um dos modelos de especificação disponíveis, de forma a ser obtida a arquitectura do software.

O próximo passo, capítulo 5, será a implementação do código comportamental da aplicação distribuída, de acordo a estrutura definida anteriormente e de modo a cumprir os requisitos funcionais propostos. Pretende-se ainda, numa fase final, colocar a aplicação desenvolvida em funcionamento numa plataforma real para assim poder ser avaliada relativamente a aspectos como a sua interface de utilizador e o seu desempenho global.

No sexto capítulo apresentam-se algumas conclusões sobre a aplicação desenvolvida no âmbito da dissertação e abordam-se também algumas perspectivas sobre uma evolução futura.

2. PROTOCOLOS DE DOMÓTICA

A domótica baseia-se na interligação de vários sistemas existentes numa habitação, sendo cada sistema composto por um conjunto diversificado de dispositivos domóticos. Estes dispositivos, que poderão ser classificados como controladores, sensores e actuadores, são essenciais para se conhecer o estado da habitação em cada momento e para se poder alterar esse mesmo estado. Todos estes dispositivos existentes numa habitação necessitam de comunicar entre si: os sensores necessitam de reportar as alterações no ambiente (detecção de movimento, temperatura, intensidade luminosa, etc.); os controladores necessitam de enviar comando para alteração de estado e, por sua vez, os actuadores necessitam de receber ordens (ligar aparelho, colocar a luz do tecto a 50%, etc.).

Neste capítulo serão analisadas as características que definem um protocolo de domótica, sendo de seguida apresentados alguns dos protocolos existentes actualmente no mercado. Nesta análise será dada uma maior ênfase ao X10 e EIB.

2.1. CARACTERÍSTICAS DE UM SISTEMA DOMÓTICO

Um sistema domótico assenta sobre três conceitos técnicos:

1. Tipo de arquitectura
2. Meios de transmissão
3. Protocolo de comunicação

Tipo de arquitectura

A arquitectura de um sistema domótico especifica o modo como os diferentes elementos de controlo do sistema se interligam. Neste campo podem-se encontrar duas arquitecturas fundamentais: a arquitectura centralizada e a distribuída. A primeira é caracterizada por possuir um elemento central, pelo qual passa toda a informação. Este tipo de arquitectura é normalmente mais económico pois retira a capacidade de processamento dos diversos dispositivos e centraliza tudo num único dispositivo. Uma arquitectura descentralizada não necessita de nenhum elemento central, sendo desta forma mais flexível e imune a falhas, já que a falha de um dos elementos apenas compromete o funcionamento desse mesmo elemento.

Meios de transmissão

O meio de transmissão é o suporte físico onde circula a informação trocada entre os diversos dispositivos da rede de domótica. Normalmente cada sistema de domótica suporta vários meios de comunicação, usando *gateways* para os ligar entre si.

Os principais meios de transmissão de dados são: cablagem metálica, rede eléctrica, fibra óptica, infra-vermelhos e radiofrequência.

a. Transmissão por cablagem metálica

Utiliza uma cablagem metálica própria como suporte para a transmissão dos sinais de controlo (e por vezes também para alimentação dos módulos e transmissão de sinais de voz). Este meio de transmissão é sobretudo utilizado nas redes telefónicas, na distribuição de sinais audio-vídeo, som de alta-fidelidade e dados. Normalmente são usados pares de cobre entrançados (redes Ethernet ou barramento EIB), podendo também ser usados cabos coaxiais para permitir maiores velocidades de transmissão. Como principais vantagens podemos destacar a grande fiabilidade e uma boa velocidade de transmissão. Em contrapartida este tipo de transmissão implica um grande custo de implementação, principalmente em habitações já construídas.

No caso do uso de cablagem metálica como meio físico de comunicação, deve ter-se em conta a topologia da rede. Esta pode ser, conforme indicado na Figura 2.1, uma topologia em barramento, estrela, anel, árvore ou mista.

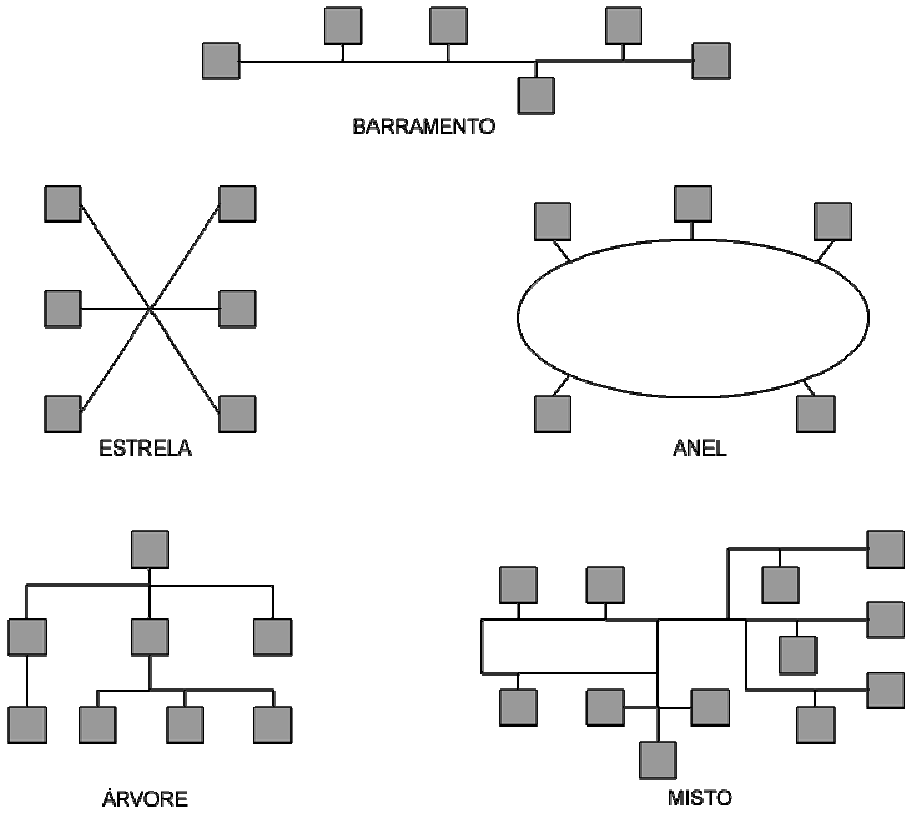

Figura 2.1 - Topologias de ligação

b. Transmissão por rede eléctrica

Utiliza a rede eléctrica já existente numa habitação. Através da inserção de altas-frequências conseguem-se modular sinais na actual rede eléctrica sem interferir com os habituais aparelhos eléctricos. Este tipo de meio tem como principal vantagem o baixo custo da instalação, sendo facilmente usado em casas já construídas. A sua maior desvantagem encontra-se na baixa velocidade de transmissão associada a uma fiabilidade reduzida.

c. Transmissão por fibra óptica

Utiliza uma combinação de tecnologias de semicondutores e de fibras ópticas. A fibra óptica é um meio de transmissão que apresenta grande fiabilidade na transferência de dados e imunidade a interferências electromagnéticas. Apresenta velocidades de transmissão elevadas associadas a um custo elevado dos cabos, ligações e equipamento. Este meio é pouco usado nos sistemas de domótica

actuais.

d. Transmissão por infra-vermelhos

A transmissão por infra-vermelhos está amplamente difundida hoje nos equipamentos áudio e vídeo. Apresenta enorme comodidade e flexibilidade nas aplicações e grande imunidade a interferências electromagnéticas. Tem como principal desvantagem o facto de ser necessário o "estar à vista" entre o transmissor e o receptor.

e. Transmissão por radiofrequência

A transmissão por radiofrequência apresenta uma maior flexibilidade no controlo à distância podendo o sinal ultrapassar paredes e outros obstáculos. Este tipo de transmissão é bastante sensível às interferências electromagnéticas e apresenta uma velocidade de transmissão muito baixa sendo normalmente usada em controlos remotos que necessitem de grande mobilidade.

Protocolo de comunicação

Para a comunicação entre dois dispositivos não é apenas necessário interligá-los através de um meio de comunicação. Também é necessário que ambos falem a mesma língua, que usem o mesmo protocolo de comunicação. O protocolo indica o formato das mensagens, a linguagem comum a todos aparelhos para que se entendam entre si. Entre os protocolos existentes, faz-se a seguinte distinção:

Protocolos normalizados – são os protocolos que estão definidos em normas, habitualmente utilizados por diferentes empresas que fabricam produtos que são compatíveis entre si.

Protocolos proprietários – são aqueles que são desenvolvidos por uma única empresa e apenas essa empresa, ou outras devidamente licenciadas, fabricam produtos capazes de comunicar entre si.

2.2. X10

A tecnologia X10 foi desenvolvida entre 1976 e 1978 pela empresa Pico Electronics Ltd, em Glenrother, Escócia, com o objectivo de transmitir dados através da linha eléctrica a baixa velocidade (50 bps na Europa e 60 bps nos EUA). O nome X10 deveu-se ao facto de este ser o décimo projecto da referida empresa [Rye, 97].

A patente foi posteriormente adquirida pela empresa X10 Ltd (que adoptou o mesmo nome que o protocolo) e mantida até ter expirado em 1997. O X10 é actualmente um protocolo aberto, sendo várias as empresas que comercializam produtos baseados nesta tecnologia. A enorme variedade de módulos e interfaces X10, a utilização da rede eléctrica existente como meio de transmissão, os preços baixos e a facilidade na instalação (a maior parte dos módulos são *plug&play*) contribuíram para o rápido sucesso do X10 nos EUA e Europa.

A tecnologia X10 usa a rede de distribuição de energia eléctrica como o principal meio de comunicação entre os vários dispositivos. Este é um aspecto chave desta tecnologia e a sua maior vantagem face a outros protocolos de domótica. Esta tecnologia usa uma arquitectura descentralizada, não necessitando de nenhum elemento central para o seu funcionamento.

Um sistema X10 pode ser simplesmente constituído por um conjunto de dispositivos que são comandados directamente pelo utilizador usando, por exemplo, um comando remoto. Neste caso, através do uso de um telecomando de rádio-frequência, para aumentar o seu alcance, o qual deixa de estar restrito a uma única divisão, o utilizador poderá enviar um comando. Este comando será recebido pelo receptor X10 de RF que, por sua vez, o encaminha através da rede eléctrica de energia para os respectivos destinatários. Dado que cada módulo consegue receber todos os sinais que circulam na rede eléctrica, o sistema tem de ser capaz de endereçar cada mensagem. Para solucionar este problema o protocolo X10 implementou um sistema simples de endereçamento que usa 16 códigos de casa (usando as letras de A-P) e 16 códigos de aparelho (1-16), permitindo endereçar univocamente 16x16=256 aparelhos.

A atribuição de endereços aos vários dispositivos X10 é feita manualmente nos próprios dispositivos, actuando sobre dois selectores rotativos. Num deles é escolhido o código da casa e no outro é escolhido o código do dispositivo, cabendo ao utilizador assegurar-se de que não existem dispositivos com endereços

repetidos. Caso existam dois ou mais dispositivos com o mesmo endereço ambos respondem aos comandos enviados.

Esta limitação não implica que apenas possam ser usados 256 aparelhos numa rede X10, pois podem existir vários módulos a usar o mesmo endereço. Neste caso, um sinal enviado para esse mesmo endereço irá ser lido e executado por todos os módulos configurados com esse endereço.

Para além de sistemas de actuação directa é possível construir sistemas mais complexos recorrendo a controladores X10 específicos. Estes controladores permitem, através de uma interface RS232 (série), comunicar com um PC do qual recebem, por exemplo, programações horárias para ligar ou desligar dispositivos e conjuntos de acções a desencadear em determinadas circunstâncias. O PC poderá ser usado apenas na programação do controlador, pois, após a criação de um conjunto de acções simples, o controlador poderá funcionar de uma forma autónoma, ligando e desligando dispositivos consoante as programações horárias definidas e permitindo desencadear múltiplas acções pela simples actuação de uma tecla.

De forma a facilitar o desenvolvimento de novos produtos baseados no protocolo X10, foram desenvolvidas duas interfaces PLC's que permitem a qualquer fabricante incluir um módulo de comunicação X10 nos seus produtos. Esta solução permitiu o aparecimento de vários equipamentos (consolas de alarme, aparelhos eléctricos, etc.) que são compatíveis com a norma X10.

A alimentação dos dispositivos deste sistema é feita através do próprio meio de comunicação. Cada dispositivo contém um transformador responsável por transformar o sinal proveniente da rede eléctrica 220VAC num outro sinal de 5VDC para se conseguir alimentar os circuitos integrados, que normalmente funcionam a tensões baixas e com correntes contínuas.

Os vários elementos X10 podem ser ligados, através de uma rede eléctrica de energia, usando qualquer tipologia, permitindo maior flexibilidade à aplicação. A dimensão do sistema vai depender das condições específicas de cada casa. A distância máxima que assegura uma comunicação sem grandes problemas, é de 200m, devido a grandes ruídos existentes na própria rede de comunicação. Para melhorar a qualidade da transmissão e, obviamente aumentar a distância máxima de comunicação, poderão ser usados repetidores de sinal.

2.2.1. Modo de transmissão

A comunicação X10 baseia-se na injecção de sinais de alta-frequência (120 KHz) sobre a rede de 220Vac, representando sinais binários (0 ou 1). O sinal é inserido logo a seguir à passagem da onda sinusoidal de 50Hz pela origem (*zero-crossing*), com um atraso máximo de 200 microsegundos. Esta particularidade é usada pelos receptores para saberem quando devem escutar a linha. O sinal é enviado através da rede eléctrica de energia até aos receptores X10, ligados à rede.

Para permitir o uso em instalações eléctricas trifásicas, os sinais de 120 KHz são emitidos três vezes em cada ciclo, em instantes que coincidem com a passagem por zero da tensão de cada uma das fases – ver figura 2.2. Deste modo, e recorrendo a acopladores próprios, torna-se possível comunicar com qualquer dispositivo, independentemente da fase em que esteja instalado. Por uma questão de simplificação da explicação, este aspecto será omitido na continuação do texto, referindo-se apenas os sinais relativos a uma única fase.

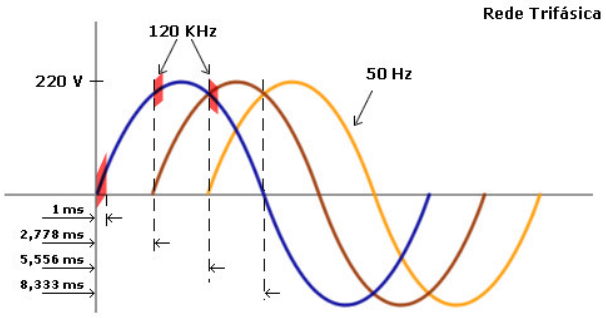

Figura 2.2 - Onda sinusoidal com a injecção de um sinal X10

Dado que o meio de distribuição de energia é muito ruidoso electricamente, foi adoptada uma política em que um bit nunca é enviado isoladamente, sendo sempre enviado o bit e o seu complemento. Isto significa na prática que, sempre que se pretende enviar o bit 1, envia-se um 1 (sinal de 120kHz na origem) seguido de um 0 (ausência de sinal). O envio do bit 0 corresponde a enviar um 0 (ausência de sinal) seguido de um 1 (frequência de 120 KHz na origem), como se encontra ilustrado na figura 2.3. Este cuidado visa minimizar a probabilidade de um ruído eléctrico poder ser confundido com um sinal válido. Contudo, tem como aspecto negativo reduzir o ritmo de transmissão, que fica assim limitado a uns meros 50 bps (é enviado um bit por cada ciclo da rede eléctrica).

É importante salientar que, de forma a simplificar a constituição dos módulos receptores, estes não confirmam a recepção dos comandos.

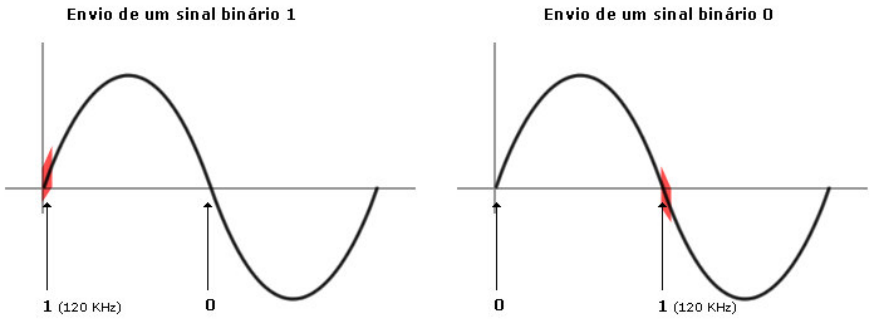

Figura 2.3 - Envio de sinais binários 1 e 0

Uma transmissão completa de um comando X10 engloba a transmissão de quatro campos que ocupam onze ciclos da onda eléctrica. O primeiro campo (2 ciclos) representa o *Start Code* – sequência de bits (1 1 1 0). De salientar que esta sequência é exactamente a indicada, não se verificando a regra de cada bit ser seguido pelo seu complemento. O campo seguinte, representado por 4 ciclos, contem o código da casa e seus respectivos complementos. Seguem-se mais 4 bits, que ocupam 4 ciclos e representam o código do aparelho ou o código da função. Para distinguir este último campo é enviado um último bit (e respectivo complemento) que identifica se o campo anterior se refere ao número de um dispositivo (bit = 0) ou ao código de uma função (bit = 1).

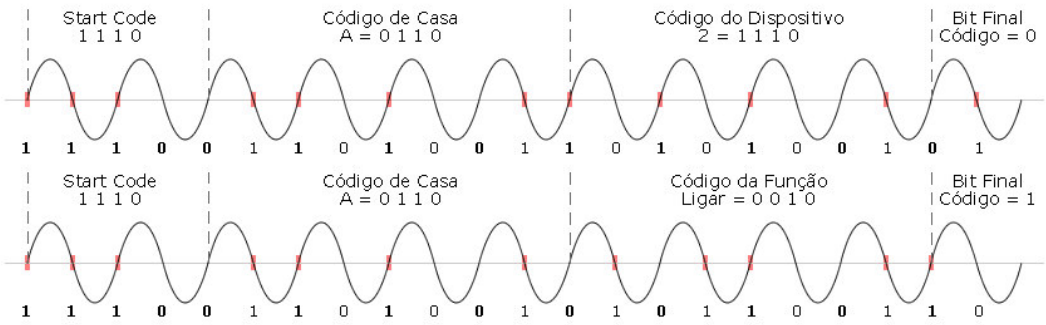

Figura 2.4 - Exemplo da Transmissão de um comando A2 ON

Cada pacote deve ser enviado em dois grupos (o primeiro indica o aparelho e o segundo a função a ser executada) com, no máximo, três ciclos da onda alternada sinusoidal entre cada grupo. Os comandos *dim* e *bright* são excepções a esta regra e devem ser transmitidos continuamente sem nenhum ciclo de intervalo entre eles.

Um comando X10 envolve normalmente duas acções: activar um dado dispositivo (mensagem com indicação de código de dispositivo) e, de seguida, enviar a função a executar (mensagem com código de função). De notar que, após a activação de um dado dispositivo, ele permanece activo até outro ser referenciado sendo

possível enviar-lhe múltiplos comandos.

Tabela 1 – Lista de comandos X10

Código	Função	Descrição
0 0 0 0	*All Units Off*	Desliga todos os dispositivos com o código de casa indicado na mensagem
0 0 0 1	*All Lights On*	Liga todos os dispositivos de iluminação (com capacidade de controlo de brilho)
0 0 1 0	*On*	Liga um dispositivo
0 0 1 1	*Off*	Desliga um dispositivo
0 1 0 0	*Dim*	Reduz a intensidade luminosa
0 1 0 1	*Bright*	Aumenta a intensidade luminosa
0 1 1 1	*Extended Code*	Código de extensão
1 0 0 0	*Hail Request*	Solicita resposta do(s) dispositivo(s) com o código de casa indicado na mensagem
1 0 0 1	*Hail Acknowledge*	Resposta ao comando anterior
1 0 1 x	*Pre-Set Dim*	Permite seleccionar dois níveis pré-definidos de intensidade luminosa
1 1 0 0	*Extended Data*	Dados adicionais (seguem-se 8 bytes)
1 1 0 1	*Status is On*	Resposta ao pedido *Status Request*, indicando que o dispositivo está ligado
1 1 1 0	*Status is Off*	Resposta indicando que o dispositivo está desligado.
1 1 1 1	*Status Request*	Pedido solicitando o estado de um aparelho

A Tabela anterior descreve todos os comandos suportados pela norma X10. Os seis primeiros são os comandos básicos usados com maior frequência. De salientar que o uso das funções *dim* e *bright* não se restringem apenas à regulação da intensidade luminosa, podendo vir a ser usados para outras funções, tais como o controlo da subida e descida de estores ou o controlo de aquecimento.

As funções *hail request* e *hail acknowledge* são usadas para determinar se é possível comunicar com uma casa vizinha. Caso esta situação se verifique, é necessário usar um código de casa diferente ou um filtro que impeça que os sinais X10 circulem para outras habitações. As funções *extended code* e *extended data* são funções introduzidas no X10 para permitir enviar mais comandos ou dados adicionais.

2.2.2. Conclusão

O protocolo X10 é actualmente a tecnologia de comunicação dentro de um ambiente doméstico mais económica e de fácil instalação. A construção de uma pequena rede de domótica é uma tarefa ao alcance de qualquer utilizador comum e permite-lhe realizar pequenas soluções de automação mesmo em habitações já construídas, sem ser necessária a instalação de qualquer tipo de cablagem.

No entanto, o X10 é uma tecnologia com grandes limitações funcionais, já que os comandos suportados restringem-se ao ligar, desligar e regular a intensidade luminosa de lâmpadas. Dada a sua constituição técnica, é um protocolo bastante lento no envio de informação (um simples comando para ligar um aparelho demora cerca de 1 segundo a ser transmitido), é limitado no endereçamento (suporta apenas 256 dispositivos) e é pouco robusto (não existe confirmação da entrega dos comandos).

2.3. EIB

O *European Installation Bus*, seguidamente tratado por EIB, foi desenvolvido como um sistema para controlo de cargas, ambiente e segurança, podendo ser instalado não só em grandes edifícios, como centros comerciais, escolas, hospitais e fábricas, como também em pequenas vivendas ou apartamentos. A sua função é a monitorização e o controlo de serviços como iluminação, aquecimento, ventilação, ar-condicionado e segurança.

A norma EIB foi proposta pela EIBA (*European Installation Bus Association*). Esta associação é formada pelos principais fabricantes europeus de electrónica e automação, tais como a Siemens, ABB, Hager, Jung, etc, contando hoje em dia com mais de 130 empresas. A EIBA tem como principais objectivos a regulamentação e promoção do EIB, através da certificação de centros de formação, da publicação de documentos periódicos informativos e da participação em feiras da especialidade.

Trata-se de um sistema operativo distribuído, baseado no modelo de referência OSI, para controlo de redes, optimizado para o controlo de casas e edifícios [EIBK, 00]. Como principais vantagens do uso da tecnologia EIB, relativamente a uma instalação eléctrica convencional, poderemos destacar as seguintes:

- Incremento na segurança;
- Uso económico e racional da energia no funcionamento da habitação ou edifício;
- Adaptação da instalação aos hábitos dos usuários;
- Maior grau de conforto;
- Compatibilidade de diferentes produtos de diferentes fabricantes.

O sistema EIB permite que os módulos que o compõem sejam alimentados pelo próprio meio de comunicação, seja este um par de cobre entrançado ou a rede eléctrica (220V). Outros módulos poderão, adicionalmente, adquirir energia através de outras fontes, como acontece com os módulos que usam radiofrequência ou infra-vermelhos para a comunicação.

Na figura 2.5 apresenta-se um pequeno exemplo de comunicação e alimentação numa rede EIB. Neste esquema pode-se verificar que existe uma separação clara entre o barramento de potência, que alimenta as cargas, e o barramento de controlo.

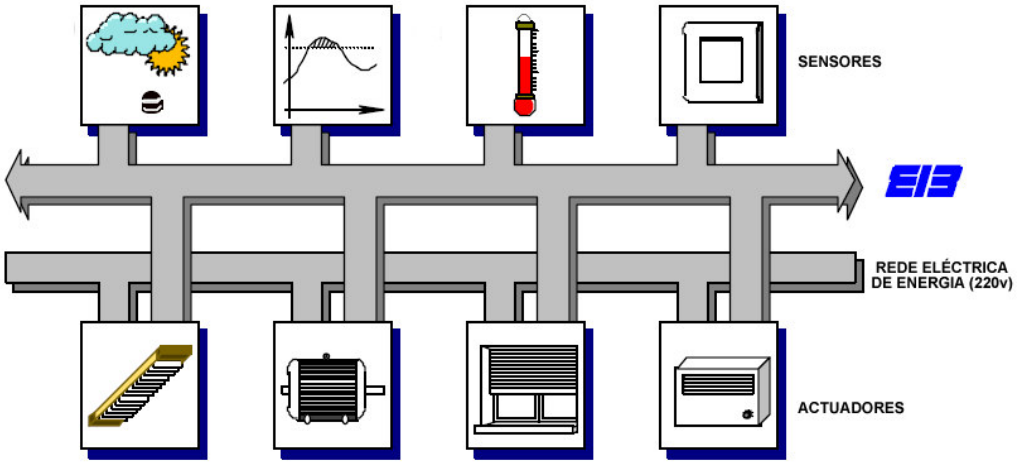

Figura 2.5 - Exemplo de um BUS EIB

2.3.1. Principais características

O EIB foi projectado de forma a permitir um controlo distribuído para a gestão e monitorização de edifícios. Desta forma permite uma comunicação em série entre todos os dispositivos ligados ao barramento.

O barramento é normalmente implementado como um sistema descentralizado mas, no entanto, permite, sempre que necessário, a implementação de um sistema centralizado, sendo apenas necessária a inserção de um controlador de aplicações (ApC – *Application Controller*) no barramento. A gestão num sistema descentralizado fica a cargo dos diferentes dispositivos que, dependendo da função, podem ser transmissores, receptores ou ambos. Estes dispositivos comunicam directamente entre si sem recurso a uma hierarquia ou a um dispositivo central de controlo, tornando o sistema muito flexível [EIBHandbook, 99].

2.3.2. Meios de comunicação

O protocolo EIB é suportado por diferentes meios, tais como o par de cobre entrançado, a rede eléctrica, a radiofrequência, os infra-vermelhos e as redes IP, através da evolução do EIB.NET. Na tabela seguinte está representado uma pequena comparação entre as várias possibilidades.

Tabela 2 – Meios de comunicação do EIB

EIB.TP	*Twisted Pair* – Cabo de cobre entrançado	É o meio de comunicação mais comum. Apresenta a maior fiabilidade e permite uma velocidade de comunicação de 9600 bps.

EIB.PL	*Power Line* – Rede eléctrica de energia	Menos fiável que o EIB.TP, permite o aproveitamento de uma rede eléctrica já existente. Usa uma modulação SFSK e permite uma velocidade de comunicação de 1200 bps.
EIB.RF	Radiofrequência	Em campo aberto permite uma transmissão até 300m.
EIB.IR	Infra-vermelhos	Está neste momento em desenvolvimento.
EIB.net	Redes IP	A norma EIB.*net* permite a comunicação entre dispositivos EIB através de redes IP. Com o EIB.*net* torna-se possível encaminhar as mensagens através da Internet.

Para além dos meios referidos poderão ser sempre utilizados outros meios através do uso de *gateways* específicas.

No EIB.TP, o sinal binário 0 ocorre quando existe uma variação no valor contínuo da tensão (alta frequência), sendo o bit lógico 1 o oposto. A escolha desta codificação dos sinais dá uma maior prioridade ao sinal 0, pois caso dois dispositivos tentem enviar sinais opostos para o barramento, será o sinal 0 o dominante. O tempo de envio de cada sinal binário é de 104 μs, o que origina uma velocidade de transmissão de 1/104 μs = 9600 bps.

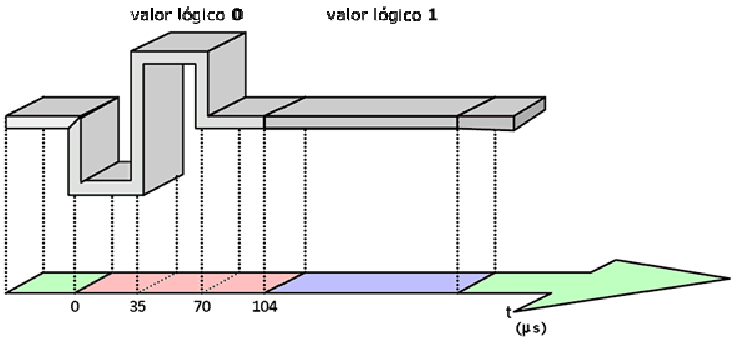

Figura 2.6 - Transmissão de um bit

A transmissão é balanceada e banda base, assíncrona, com uma velocidade de 9600 bps. Para evitar colisões no acesso ao meio é usado o protocolo CSMA/CA – *Carrier Sense Multiple Access, Collision Avoidance*.

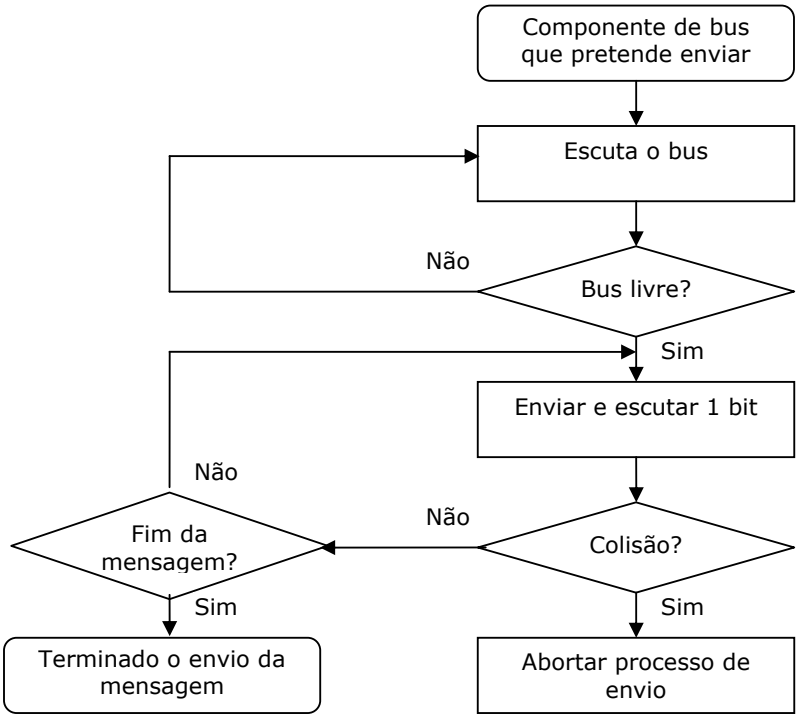

Figura 2.7 - Processo de envio CSMA/CA

Através do CSMA/CA cada dispositivo EIB analisa o barramento para verificar a sua disponibilidade, começando a transmitir apenas quando não existe nenhuma mensagem neste. Na transmissão de uma mensagem, o dispositivo fica à escuta para detectar se outro dispositivo iniciou a transmissão ao mesmo tempo. O primeiro a detectar que a sua mensagem está corrompida, liberta o meio, parando de transmitir. A outra mensagem pode assim ser transmitida, sem problemas, para o seu destino. Assim que o dispositivo anterior acaba de enviar a sua mensagem o dispositivo que detectou a ocorrência de conflito tenta novamente aceder ao meio.

O sistema EIB.PL possibilita a transmissão das mensagens através da rede eléctrica de 220V. Desta forma não é obrigatório a passagem de um cabo extra, facilitando a instalação de uma rede EIB.

Todos os dispositivos EIB.PL possuem uma unidade de acoplamento à rede (MCU – *Mains Coupling Unit*), a qual modula os sinais de forma a serem enviados pela rede eléctrica. Apesar deste ser um meio muito instável, devido à variação no tipo e comprimento dos cabos, ao número e tipo de cargas ligadas, o EIB.PL assegura uma transmissão relativamente fiável.

O EIB.PL trabalha num modo bidireccional com um funcionamento em *half-duplex*,

querendo isto dizer que todos os dispositivos podem receber e enviar informação, mas não ao mesmo tempo. Para o envio de dados recorre-se a uma modulação na frequência do tipo SFSK – *Spread Frequency Shift Keying*. Este método injecta sinais com uma frequência de 105,6 KHz para representar um 0 binário e 115,2 KHz para representar um 1. Com o fim de assegurar uma transmissão fiável usa-se uma velocidade de transmissão de 1.200 bps, que corresponde a uma duração de 833 μs na transmissão de 1 bit [EIBHandbook, 99].

Para evitar que uma mensagem possa ser transmitida para uma instalação eléctrica próxima, pertencente a uma outra habitação, a transmissão de um pacote EIB.PL usa um campo extra denominado ID de sistema. Este campo de 8 bits permite ao projectista definir um valor entre 1 e 255, estando o valor 0 reservado para o envio de mensagens para todos os dispositivos.

2.3.3. Topologia

A topologia pode ser entendida como a configuração dos caminhos sobre os quais a informação é transportada. Em alguns meios, a transmissão de sinais não está limitada a nenhum cabo transmissor de impulsos eléctricos, como é o caso da radiofrequência e dos infra-vermelhos.

No caso do par de cobre entrançado, os segmentos eléctricos podem estar organizados nas topologias de barramento, estrela, árvore ou qualquer combinação destas, desde que as características físicas da linha (resistência e comprimento) não sejam excedidas. O uso de uma rede em anel não é permitido quando num ciclo fechado se encontra um acoplador de linha. Mesmo sendo possível o uso de topologias em anel, em alguns casos especiais, o seu uso é fortemente desaconselhado pois pode originar ciclos nas mensagens, podendo ocorrer excesso de tráfego. Para evitar estes casos, e impedir que o barramento fique com excesso de mensagens cíclicas, o EIB insere nos datagramas um contador de ciclos. Esta variável, normalmente inicializada com o valor 6 (este valor pode ser alterado), é decrementada cada vez que passa por um acoplador de linha. Caso esta variável atinja o valor 0 o pacote é eliminado, evitando assim o congestionamento do barramento.

Uma ligação em barramento necessita apenas de um cabo para o seu funcionamento. Quando são usados 2 cabos, um é dedicado à transmissão de dados e o outro pode ser usado, por exemplo, para alimentar os dispositivos. A topologia de instalação é similar à instalação eléctrica convencional, sendo

normalmente usada uma topologia em árvore.

Cada segmento de par de cobre entrançado pode suportar 64 dispositivos de barramento, permitindo, no máximo, a interligação de 64.000 dispositivos. O comprimento total de cada segmento não deve exceder os 1000m, sendo a distância máxima entre um dispositivo e a fonte de alimentação de 350m. A distância máxima entre dois dispositivos é de 700m [Sauter, 02].

Em alguns casos poderá ser necessário ligar mais do que 64 dispositivos ao mesmo segmento. Neste caso, dois segmentos de barramento físicos poderão ser ligados num segmento lógico através de um repetidor, conseguindo-se duplicar a capacidade de uma linha. Em teoria, uma linha pode conter até 4 segmentos físicos ligados através de repetidores mas, por norma, numa instalação de raiz não se deverá ligar mais do que 2 segmentos, para permitir alguma flexibilidade para futuras evoluções.

Em termos lógicos, uma linha pode conter até 256 dispositivos, podendo-se agrupar 15 linhas, através de uma linha principal, formando-se uma área. Através do uso de um *backbone*, podem ser agrupadas 16 áreas permitindo acomodar 65.535 dispositivos de barramento.

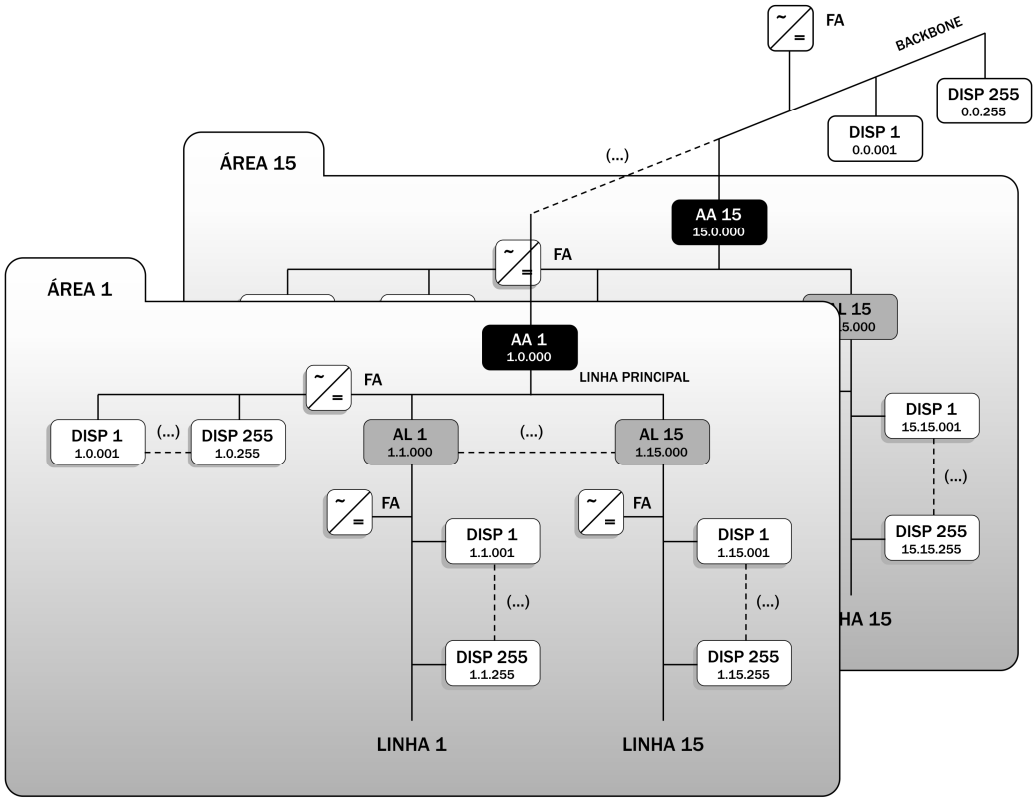

Figura 2.8 - Topologia de uma rede EIB

Devido ao contador de ciclos, referido anteriormente, um caminho de transmissão não poderá conter mais do que 6 controladores de linha, i.e., acopladores de linha, acopladores de *backbone* ou repetidores.

Os segmentos físicos ou lógicos poderão ser ligados entre si através de acopladores de linha (LC – *Line Couplers*), sendo permitido no máximo 16 segmentos. Poderão existir 15 zonas, cada uma composta por vários segmentos. Várias zonas poderão ser ligadas entre si recorrendo a outros meios de comunicação, como por exemplo a RDIS e o TCP/IP, sendo necessário nestes casos o uso de *gateways* apropriadas.

O número máximo de dispositivos que poderão ser ligados num barramento, sem o uso de repetidores, é 13105, usando apenas 12 zonas, ou de 16129, usando a totalidade das 15 zonas. Através do uso de repetidores os limites atrás referidos aumentam para 49201 e 61249, respectivamente, tornando este protocolo aplicável mesmo em edifícios de grande envergadura.

2.3.4. Modos de endereçamento

Cada dispositivo é identificado por um endereço físico único, não podendo a mesma rede EIB conter dois dispositivos com o mesmo endereço. O endereço físico consiste numa zona, numa linha e num número de dispositivo (ex. 5.2.133). Conforme referido anteriormente, o número de zona e linha pode conter um valor numérico entre 0 e 15, ao passo que um número de dispositivo pode ir de 0 a 255. O valor de 0 está sempre associado ao um acoplador de área (ex. 1.0.000) ou de linha (ex. 1.1.000) e é aconselhável deixar o valor 255 disponível para se poder inserir um dispositivo visualizador (ex. interface série para PC).

Numa mensagem, o endereço do remetente contém obrigatoriamente o seu endereço físico, enquanto que o endereço do destinatário pode ser um endereço físico ou um endereço de grupo. O uso de um endereço físico como destino aplica-se apenas para operações de inicialização, programação ou diagnóstico, casos que correspondem ao modo de acesso ao sistema.

O endereçamento de grupo corresponde ao modo normal de funcionamento. Funções de dispositivos EIB, pertencentes ao mesmo grupo, podem ser controladas apenas pelo envio de uma única mensagem. É importante salientar que um dispositivo pode ter associados vários endereços de grupo mas apenas um deles pode ser definido como endereço de grupo emissor. Esta definição é efectuada através do ETS (*EIB Toolkit Software*), na altura de programação dos dispositivos.

Um endereçamento de grupo pode ser definido usando um endereçamento de 2 níveis (grupo principal/subgrupo) ou um endereçamento de 3 níveis (grupo principal/grupo intermédio/subgrupo).

Num endereçamento de 2 níveis estão reservados 4 bits para o grupo principal e 11 bits para o subgrupo. Desta forma torna-se possível usar 16 (0 a 15) grupos principais diferentes e 2048 subgrupos. No caso de um endereçamento de 3 níveis o grupo principal ocupa os mesmos 4 bits (16 valores), estando associados ao grupo intermédio 3 bits (8 valores) e os restantes 8 bits para o subgrupo, permitindo que este possa assumir 256 valores diferentes.

A escolha do modo de endereçamento fica a cargo do projectista. O mais comum é a escolha de um endereçamento de 3 níveis usando o seguinte critério:

1. Grupo principal — área física (ex. 1º andar, R/C);
2. Grupo intermédio — função (ex. 0 funções centrais, 1 iluminação, 3 *dimmers*);
3. Subgrupo — carga ou grupo de cargas (ex. luzes da sala, persianas da suite).

A comunicação entre um sensor (ex. um interruptor) e um actuador (ex. uma lâmpada) é uma sequência de operações descritas na Figura 2.9. Neste exemplo, usando o protocolo EIB, o interruptor, estando inequivocamente definido pelo seu endereço físico, consegue comunicar com as lâmpadas usando o endereçamento de grupo.

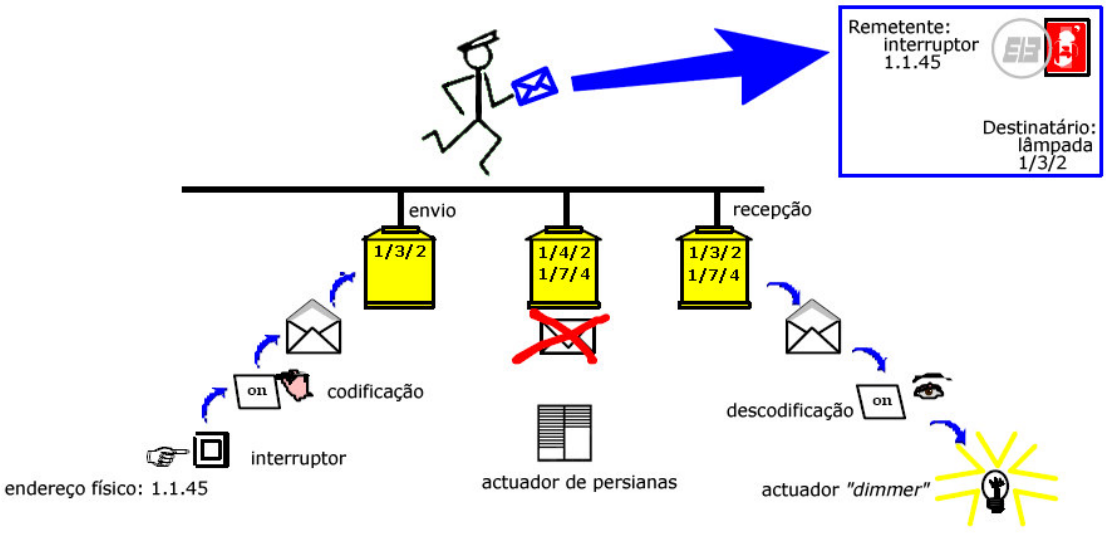

Figura 2.9 - Transmissão de uma mensagem

Se, por exemplo, no esquema anterior associarmos ao objecto de comunicação do

interruptor o endereço de grupo 1/7/4, em vez de 1/3/2, ambos os dispositivos (persianas e actuador) receberiam o telegrama (designação dada às mensagens EIB).

Normalmente um mesmo sensor contém vários objectos de comunicação associados a diferentes interacções. Por exemplo, um interruptor pode ter um objecto de comunicação associado a um toque curto e um outro associado a um toque longo. Desta forma torna-se possível através de um toque curto ligar e desligar uma lâmpada e, com um toque longo, alterar a sua luminosidade. De salientar que um actuador também pode ter associados vários objectos de comunicação. No exemplo anterior, o actuador *dimmer* contém um objecto de comunicação associado ao ligar/desligar e um outro associado ao controlo da intensidade luminosa.

Um objecto de comunicação é um endereço de memória, nos dispositivos de barramento, sendo o tamanho deste dependente da função associada. Ao usar-se direcções de grupo apenas se podem unir (atribuir o mesmo endereço de grupo) objectos com o mesmo tamanho.

2.3.5. Comunicação

Para que os diversos dispositivos EIB comuniquem entre si, não só precisam de conseguir falar uns com os outros, como necessitam de usar a mesma linguagem de forma que os dados transmitidos e recebidos tenham o mesmo significado para todos os dispositivos. O EIB soluciona este problema através da definição de EIS (*EIB Interworking Standard*).

Na tabela seguinte estão resumidos os diferentes tipos de dados definidos pela norma EIS. Ainda que o nome seja bastante indicativo do tipo de dados, não significa que estes estejam limitados a essa função. Por exemplo, o tipo de dados 2 (EIS 2), denominado *dimming* (controlo de iluminação), também poderá ser usado para controlar a temperatura. Neste caso os dados seriam interpretados como mais quente/frio em vez de mais claro/escuro.

Tabela 3 - Códigos EIS

EIS Type	Função EIB	Nº bytes	Descrição
EIS 1	*Switching*	1 bit	Ligado/desligado, verdadeiro/falso.

EIS 2	*Dimming*	4 bits	Pode ser utilizado de 3 formas: como interruptor, como valor relativo ou como valor absoluto.
EIS 3	*Time*	3 bytes	Hora, minutos e segundos.
EIS 4	*Date*	3 bytes	Dia, mês e ano (o ano encontra-se entre 1990 e 2089).
EIS 5	*Value*	2 bytes	Para enviar valores físicos.
EIS 6	*Scaling*	1 byte	Utiliza-se para a transmissão de valores relativos com uma resolução de 8 bits.
EIS 7	*Control Drive*	1 bit	Pode ser usado de duas formas: movimento contínuo (subir/descer) ou passo a passo.
EIS 8	*Priority*	1 bit	Utiliza-se em conjunto com as mensagens EIS 1 a EIS 7.
EIS 9	*Float Value*	4 bytes	Codifica um número em vírgula flutuante.
EIS 10	*16b-counter*	2 bytes	Representa os valores de um contador de 16 bit (negativos ou positivos).
EIS 11	*32b-counter*	4 bytes	Representa os valores de um contador de 32 bit (negativos ou positivos).
EIS 12	*Access*	4 bytes	Usa-se para conceder acesso a funções distintas.
EIS 13	Character	1 byte	Carácter ASCII.
EIS 14	8b-counter	1 byte	Representa os valores de um contador de 8 bit (negativos ou positivos).
EIS 15	Character String	14 bytes	Transmite uma cadeia de caracteres ASCII até 14 bytes.

É importante salientar que uma mensagem EIS é apenas uma definição abstracta de um comando, sendo a sua interpretação dependente dos dispositivos que enviam e recebem o telegrama. Aliás, na própria mensagem não se encontra nenhuma informação que indique qual o tipo de EIS enviado.

De seguida são descritas algumas das mensagens EIS mais importantes:

O EIS do tipo 1 *Switching,* é normalmente usado para alterar o estado de um actuador, mas pode também ser usado em operações lógicas ou funções de armar ou desarmar dispositivos (por exemplo, consolas de alarme).

O EIS 2 *Dimming*, pode-se decompor em três EIS distintos:

a) Um objecto de comutação (EIS 1);

b) Um objecto de 4 bits para se efectuar uma regulação relativa. Por exemplo, podemos enviar um comando a um *dimmer* para aumentar a luminosidade em 25% relativamente ao seu valor actual. Nesta mensagem o bit mais significativo indica o sentido da regulação (1 – mais claro, 0 – mais escuro), usando-se os restantes 3 bits para indicar valor da regulação (ex. 1001 – subir 50%; 0011 – descer 25%);

c) Um objecto de 1 byte (EIS 6 – *Scaling*), indicando o valor absoluto da luminosidade (1 – 0%, 255 – 100%).

O EIS 7 *Control Drive* é normalmente usado para o controlo de persianas e permite alterar o estado do motor (comando *move*), quer para colocar o motor em movimento, caso este se encontre em repouso, ou alterar o sentido do movimento, caso este já se encontre em movimento. Este EIS também pode ser usado para comandos do tipo *step*, o qual permite mover o motor durante um pequeno espaço de tempo.

O EIS 8 *Priority,* permite bloquear um actuador. Por exemplo, se na instalação de um toldo exterior associarmos um detector de vento, este deverá bloquear o actuador do toldo para que, em situações metrológicas extremas, o toldo não possa ser aberto involuntariamente por um utilizador, correndo o risco de ser danificado.

2.3.6. Estrutura interna dos Pacotes

A troca de informação entre dois dispositivos é conseguida através da troca de pacotes de dados. Todos os componentes de barramento confirmam a recepção de um telegrama através do envio de um *acknowledge*.

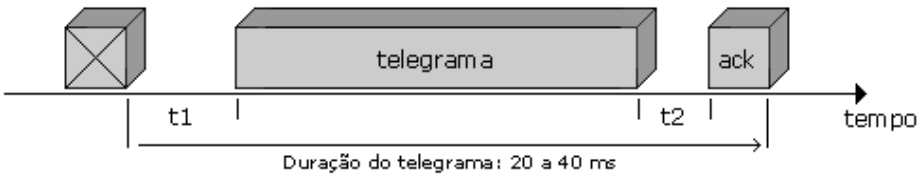

Figura 2.10 - Tempos de transmissão de um telegrama

A figura 2.10 apresenta os tempos de envio de uma mensagem. Pode-se verificar que um dispositivo do barramento apenas inicia o envio da mensagem após comprovar que o barramento está desocupado durante um período mínimo de

tempo (t1). Após a recepção do telegrama, o componente que o recebeu tem um tempo máximo (t2) para o envio de um *acknowledge*.

O pacote *acknowledge* de confirmação de recepção de um telegrama é composto por 8 bits, podendo ter 3 valores distintos como indicado na tabela seguinte:

Tabela 4 – Valores possíveis de um *acknowledge*

1 1 0 0 0 0 0 0	BUSY – Barramento ocupado
0 0 0 0 1 1 0 0	NAK – *Not acknowledge* – Recepção com erro
1 1 0 0 1 1 0 0	ACK – *Acknowledge* – Recepção correcta

No caso da não recepção de um ACK ou da recepção de um NAK, o dispositivo de barramento emissor tenta o envio mais 3 vezes. No caso de receber um BUSY, o dispositivo espera um curto intervalo de tempo e tenta novamente a emissão da mensagem.

Figura 2.11 - Composição de um telegrama

Conforme indicado na figura anterior, uma mensagem EIB é composta pelos seguintes campos [Sauter, 02]:

1. Campo de Controlo – **1 0 R T P P 0 0**

   O bit **R**, normalmente com valor **1**, indica se o telegrama está a ser enviado pela primeira vez (**1**) ou está a ser reenviado porque, por exemplo, o dispositivo que o enviou não recebeu um *Acknowledge.*

   O bit **T** indica se esta mensagem é um telegrama (valor **1**) ou não (no caso de um *Acknowledge*).

   Os bits **P P** indicam a prioridade da mensagem: **0 0** - mensagem com a prioridade máxima (mensagem de sistema); **1 0** – um alarme; **0 1** – prioridade alta; **1 1** – prioridade baixa.

2. Endereço de origem – **A A A A L L L L D D D D D D D D**

   Este campo de 16 bits indica o endereço físico (*área.linha.dispositivo*) do dispositivo de barramento emissor.

3. Endereço de destino
   Este campo indica o endereço de destino e pode ser formado de 2 formas distintas, dependendo do seu décimo sétimo bit. Este último bit, que na verdade faz parte do byte seguinte, indica se o campo de endereço de destino é um endereço físico (valor binário **1**) do tipo **AAAALLLL DDDDDDDD** ou um endereço de grupo (valor binário **0**) **xPPPPIII SSSSSSSS** do tipo grupo **P**rincipal/grupo **I**ntermédio/**S**ubgrupo. O bit **x** não tem nenhum significado.

4. Contador de ciclos + Tamanho dos dados
   O contador de ciclos é composto por 3 bits e representa a variável que é decrementada sempre que a mensagem é retransmitida por um acoplador de linha. Os 4 bits seguintes indicam o tamanho dos bytes enviados de seguida.

5. LDSU (*Link Service Data Unit*) – informação a ser transmitida
   Este campo pode ter até 16 bytes de dimensão.

6. Byte de segurança
   Este último byte enviado serve para verificar a integridade de toda a mensagem.

2.3.7. Dispositivos EIB

Os dispositivos EIB estão divididos em três grupos, dependendo do seu tipo:

- Componentes básicos do sistema, tais como fontes de alimentação (PSU – *Power Supply Unit*), bobinas, filtros, etc;
- Componentes do sistema para a criação de uma rede, tais como acopladores de barramento (BCU – *Bus Coupling Unit*), acopladores de linha (LC – *Line Coupler*), acopladores de fase, repetidores, etc;
- Dispositivos EIB orientados para as aplicações, tais como sensores, actuadores, receptores de IV, painéis de comando, etc. Estes dispositivos são ligados à rede através de um acoplador de barramento ou de uma interface similar.

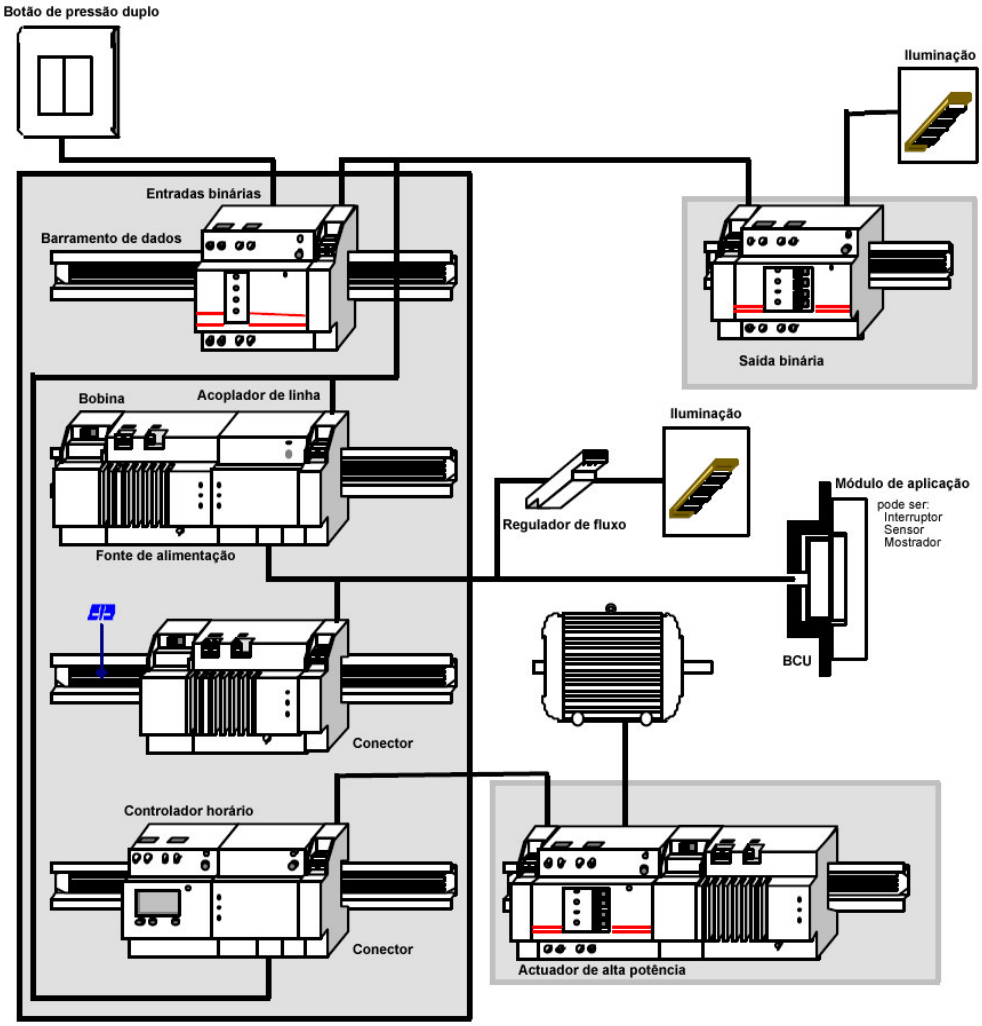

Figura 2.12 - Exemplo de uma instalação

Componentes básicos:

- Fonte de alimentação – Fornece a energia para alimentar os dispositivos EIB (SELV *Safety Extra Low Voltage*), 30V DC;

- Bobina – Serve de acoplador da fonte de alimentação para o barramento;

- Barramento – Suporte que distribui o barramento ao longo de uma calha DIN;

- Conector – Serve de ligação entre o cabo do barramento e o *Data Rail*.

Componentes do Sistema:

Cada dispositivo EIB pode ser dividido em três partes distintas:

- Unidade de acoplamento de barramento (BCU – *Bus Coupling Unit*);
- Módulo de aplicação (AM – *Application Module*);

- Programa da aplicação (AP – *Application Program*).

Ainda que existam dispositivos EIB que integram as três partes sem que exista uma distinção clara entre elas, a especificação está pensada para que cada módulo possa ser adquirido a um fabricante distinto.

Para possibilitar a comunicação entre BCU's e os módulos de aplicação de diferentes fabricantes, o EIB define a interligação destes elementos através da interface PEI – *Physical External Interface* [EIBHandbook, 99].

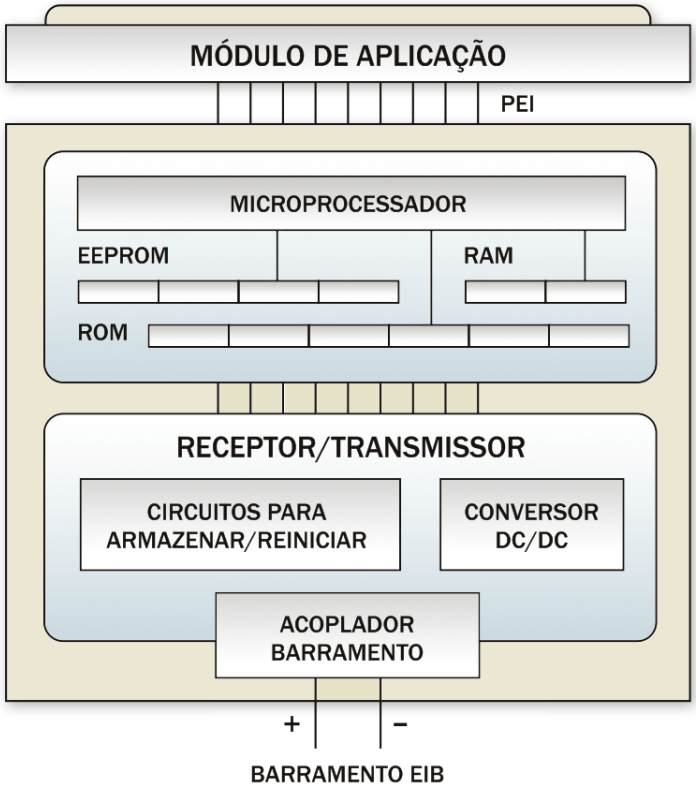

Figura 2.13 - Esquema de um BCU

O módulo BCU ocupa-se das tarefas de comunicação com o barramento e manutenção do estado interno do dispositivo. Este pode ser dividido em duas partes: receptor/transmissor e controlador. Este último consiste num microprocessador com memória constituída por uma ROM, RAM e EEPROM.

A memória ROM consiste numa série de programas de sistema, gravados no processo de fabrico, e que se encarregam de implementar as funções de comunicação e comportamento genérico do módulo BCU. A EEPROM armazena os parâmetros do sistema e o programa da aplicação, sendo este inserido aquando a

configuração do sistema.

A RAM utiliza-se como ponte de comunicação entre os programas do sistema e os programas da aplicação. É nesta memória que são armazenados os objectos de comunicação que contêm informação sobre o estado do dispositivo, como por exemplo, se uma lâmpada está ligada ou apagada, a hora de um relógio, o estado de um interruptor, etc. Cada dispositivo pode ter um ou mais objectos de comunicação, sendo o seu conteúdo definido pela norma EIS. Cada objecto tem uma direcção de grupo associada que pode ser única, caso se trate de um objecto de comunicação emissor, ou não, no caso de objectos receptores.

O módulo de aplicação adapta o dispositivo para a sua função concreta. Por exemplo, caso se trate de um sensor, o módulo de aplicação é o responsável por recolher a informação do meio (quer sejam leituras de temperatura, detecção de movimento, detecção de luminosidade, etc.) e enviar essa informação, através do PEI, para o BCU. Este por sua vez codifica a informação recebida e envia-a através do barramento, sendo da responsabilidade do BCU a verificação periódica do estado do sensor.

No caso dos actuadores ocorre o processo inverso: a BCU recebe a informação do barramento, descodifica-a e passa a informação para o módulo de aplicação, que por sua vez altera o estado do aparelho.

Actualmente apenas estão disponíveis módulos BCU para ligações através de pares de cobre entrançados e da rede eléctrica. Em breve serão disponibilizados novos BCU's que suportem outros meios de comunicação, tais como rádio frequência e infra-vermelhos.

Para interligar as diferentes linhas que compõem uma rede EIB é necessário o uso de acopladores de linha. Estes dispositivos funcionam apenas em cabos de cobre e asseguram a interligação entre as linhas secundárias e a linha principal. O acoplador de linha assegura o encaminhamento e o *buffering,* com gestão de *overflow,* de uma linha secundária para uma linha principal e vice-versa.

Um repetidor é também um componente de sistema associado apenas à transmissão através de cabos de par de cobre. Este elemento permite ligar dois segmentos eléctricos, regenerando os sinais que recebe.

2.3.8. Conclusão

O EIB é sem dúvida uma das mais promissoras tecnologias para a instalação de uma rede doméstica. Contando com o apoio dos maiores fabricantes europeus de dispositivos eléctricos e electrónicos, existem já centenas de dispositivos diferentes que permitem implementar todas as funcionalidades necessárias para uma casa inteligente.

O facto de cada dispositivo conter a sua própria inteligência, através do uso de um microprocessador e memórias internas, torna esta tecnologia algo cara, já que o dispositivo mais simples, que obrigatoriamente tem de incluir um BCU, tem um custo algo elevado. Com o aumento da procura por parte dos consumidores e o aparecimento de equipamentos eléctricos de linha branca (frigoríficos, máquinas de lavar, etc.) e castanha (televisores, aparelhagens, etc.) já compatíveis com a norma EIB, o custo do equipamento tenderá a baixar.

Esta tecnologia não foi pensada para ser instalada e configurada pelo utilizador. Neste campo a EIBA, desenvolveu uma rede de certificação que garante a qualidade nas instalações EIB. Obviamente esta qualidade acarreta um custo na instalação, obrigando a que esta seja efectuada na fase inicial da construção para não aumentar ainda mais o custo final.

2.4. CEBus

Em 1984, o *Consumer Electronics Group* da EIA – *Electronic Industries Association* iniciou um esforço tendo como objectivo a formulação de uma norma para o desenvolvimento de uma rede de comunicação para dispositivos domésticos. Esta norma veio mais tarde a ser definida como CEBus – *Consumer Electronic Bus*.

Tendo como principal motivação o desenvolvimento de um protocolo universal, de baixo custo, para a comunicação entre vários dispositivos domésticos, independentemente do fabricante, e de fácil uso por parte do consumidor, foram criados vários grupos de trabalho para a definição dos meios físicos de comunicação e do protocolo. Em 1989 foram publicados alguns "rascunhos", sendo o seu lançamento final efectuado em 1992.

Como é o caso da grande parte das especificações da EIA, o CEBus é mais uma norma de desempenho do que uma norma de construção. Ou seja, a norma define como é que um dispositivo se deve comportar em vez de definir como é que deve ser construído. Esta norma é orientada ao modelo, definindo as camadas de comunicação que permitem a implementação do protocolo de várias formas (hardware, software ou uma combinação de ambas) [Evans, 01].

A norma CEBus deve:

- Ser facilmente estendida ao longo do tempo para permitir o uso de novos meios de comunicação, novas tecnologias e novos dispositivos domésticos. A descrição das funções CAL (*Common Application Language*) deve ser projectada para permitir futuras expansões nas suas tabelas de funções;
- Funcionar num ambiente doméstico, quer este seja um pequeno apartamento ou uma vivenda com várias assoalhadas;
- Ser versátil, permitindo o uso de sistemas distribuídos e centralizados;
- Ser simples e de fácil instalação por parte do consumidor, sem que seja necessária nenhuma formação especial;
- Ser económica de forma a ser usado em simples aparelhos domésticos;
- Ser compatível e permitir que os dispositivos de diferentes fabricantes possam comunicar entre si;
- Permitir o acesso a diferentes meios e ser independente destes;
- Ter uma resposta rápida independentemente do nível de tráfego. O atraso

introduzido pelo protocolo deve ser sempre inferior a uma décima de segundo;

- Definir prioridades de acesso ao meio;
- Impedir que um dispositivo domine o acesso ao meio impedindo os restantes de comunicar.

É importante referir que o CEBus não tem como objectivo ser um protocolo de segurança, nem proteger o acesso ao meio por estranhos.

Para possibilitar a ligação a diversos tipos de equipamentos o CEBus permite o uso dos seguintes meios de comunicação: par de cobre entrançado, rede eléctrica, cabo coaxial, fibra óptica, infra-vermelhos e radiofrequência.

Todos os meios físicos de comunicação transportam o canal de controlo CEBus e transmitem a informação com o mesmo ritmo de transmissão (cerca de 8000 bits por segundo). Também podem transportar canais de dados com larguras de bandas adequadas para vídeo e áudio, dependendo do meio de comunicação físico. Os comandos e as informações do estado são transmitidos no canal de controlo na forma de mensagens, compostas por pacotes de bytes. A maior parte da especificação do CEBus é dedicada à especificação do canal de controlo [WACK97].

Para permitir aos dispositivos comunicarem entre si e executarem certas tarefas, usando regras comuns de sintaxe e de vocabulário, é especificada uma linguagem de comunicação comum – CAL [Dutt, 99].

O formato das mensagens CEBus de controlo é independente do meio de comunicação usado. Cada mensagem contém o endereço destino, sem nenhuma referência aos meios de comunicação onde o emissor e o receptor estão colocados, permitindo ao CEBus a formação de rede lógica uniforme.

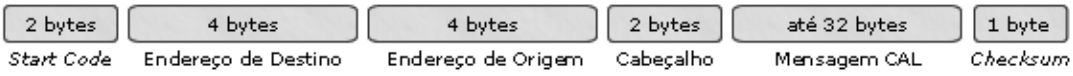

Figura 2.14 - Telegrama CEBus

As mensagens CEBus são telegramas que possuem um *start code*, um endereço de origem, um endereço destino, uma mensagem CAL e um byte de checksum. Um pacote típico CEBus inclui uma mensagem CAL de 32 bytes e 9 bytes de controlo e endereçamento [Webb, 99]. O *start code* é composto por um preâmbulo (1 byte), que testa se o meio de transmissão se encontra disponível, e um campo de controlo

(1 byte).

Além de se poderem enviar mensagens individuais, todos os dispositivos, ou grupos restritos de dispositivos, podem ser alcançados por uma única mensagem contendo um endereço de difusão único. Neste caso, todos os dispositivos respondem ao endereço de difusão. Para além do endereçamento individual, poderão ser endereçados grupos de dispositivos possibilitando, com uma única mensagem, o controlo de vários dispositivos em simultâneo.

Para a instalação e programação de uma rede CEBus, são disponibilizadas as seguintes ferramentas [Webb, 99]:

- HomeToolPro – aplicação usada para a instalação do sistema CEBus, que possibilita a criação rápida de um projecto onde se identificam facilmente os recursos e se especificam os comandos CEBus, possibilitando a criação de cenários de controlo;

- CEBench – ferramenta de desenvolvimento para Windows. Permite a criação de um modelo do projecto que representa os objectos e variáveis da linguagem de comunicação CAL;

- CETester – ferramenta usada para teste e monitorização de redes CEBus;

- CEBox – software que integra várias ferramentas e livrarias do protocolo, que facilitam o desenvolvimento de nós CEBus [CeBus, 05].

Nos últimos anos, o protocolo tem tido uma crescente aceitação no mercado Americano, estando actualmente a evoluir para o HPnP (*Home Plug'n'Play*).

2.5. LONWORKS

LonWorks, ou simplesmente LON (*Local Operating Network*), é uma topologia de rede, criada pela Echelon, com o intuito de solucionar o problema de controlo de sistemas. Esta tecnologia fornece uma solução para muitos problemas de projecto, construção, instalação e manutenção de redes de controlo, cujo tamanho pode ascender os 32000 dispositivos ligados através de par entrançado, cabo de fibra óptica, cabo coaxial ou infra-vermelhos. Trata-se de uma tecnologia e não de um produto final, uma vez que é exclusivamente vendida à indústria e não aos utilizadores finais, sendo uma solução completa para redes de controlo distribuído, podendo ser usada em qualquer lugar, desde residências a edifícios de grande envergadura [Dietmar, 01].

Todo o sistema LonWorks é controlado pelo grupo *LonMark Interoperability Association*, o qual garante a compatibilidade de todos os produtos utilizados, ainda que provenientes de diferentes fornecedores.

A plataforma LonWorks pode utilizar uma grande variedade de meios de comunicação:

- Rede eléctrica;
- Par entrelaçado;
- Rádio Frequência;
- Infravermelhos;
- Cabo coaxial;
- Fibra óptica.

A comunicação estabelece-se a taxas de transmissão que vão de 1kb/s a 1,25 Mb/s, dependendo do meio de comunicação. Os diferentes tipos de meios de comunicação usados na rede são ligados por meio de encaminhadores.

A tecnologia LonWorks é um sistema aberto, permitindo combinações de componentes de diferentes fabricantes, permitindo também, adicionar novas funções de controlo com um custo baixo.

O LonWorks possui um protocolo chamado LonTalk que implementa as sete camadas do modelo OSI (modelo de referência para a interligação de sistemas abertos) e possui mecanismos que impedem modificações acidentais ou intencionais. Uma parte integral do protocolo é a sua própria técnica de acesso ao

meio de comunicação, designadamente por "*predictive p-persistent CSMA, with optional priority and collision detection*", que supera as técnicas de CSMA/CD tradicionais. Inclui ainda outras características, tais como: autenticação do remetente, funções de reconhecimento, comunicação, prioridade na transmissão, detecção de mensagens duplicadas, retransmissão automática, detecção e correcção de erros, normalização e identificação do tipo de dados.

Apesar do protocolo LonTalk ser aberto, permitindo a qualquer empresa colocá-lo no processador que desejar, é disponibilizado o *Neuron Chip*. Especialmente concebido para a tecnologia LonWorks, permite controlar todos os dispositivos utilizados naquela tecnologia, tais como ventoinhas, motores, interruptores, etc. É o dispositivo preferencial do protocolo LonTalk porque foi criado especificamente para o efeito, sendo fabricado actualmente pela Motorola e Toshiba.

Para a configuração dos dispositivos, instalação e gestão de redes, são utilizadas as seguintes ferramentas [Wack, 97]:

- LonBuilder *Developer's Workbench* – É a ferramenta mais antiga da Echelon e é um ambiente completo para a criação, instalação e teste das redes LonWorks. Incorpora várias ferramentas de desenvolvimento para os vários nós (interfaces para PC, transdutores para os meios de comunicação, emuladores *Neuron*, compilador *Neuron* C, gestor de projecto, editor de texto, etc.), uma ferramenta de gestão de rede e um analisador do protocolo LonTalk;

- LonMaker – é um software, executado sobre o sistema operativo Windows, usado na instalação dos dispositivos LonWorks. Estes são representados graficamente por objectos que podem ser manipulados. Esta aplicação trabalha sob o LNS (*LonWorks Network Services*) permitindo a criação de ligações lógicas entre os diferentes dispositivos;

- LNS DDE Server – Permite efectuar a monitorização e o controlo da rede LonWorks, à qual se liga, tal como o anterior, através de NSIs (*Network Service Interfaces*). Permite também ligar os objectos LonWorks a quase todas as aplicações Windows (por exemplo, estabelecer uma ligação entre uma pequena rede LonWorks e o Microsoft Excel, permitindo a monitorização do estado dos nós);

- LonManager – Assim que os *Neuron Chips* e os microprocessadores associados estejam devidamente programados, a rede é configurada com

uma das ferramentas do LonManager. Este oferece várias APIs (para DOS e Windows) que permitem gerir e analisar a topologia da rede, carregar aplicações nos *Neuron Chips*, simular dados e mensagens, guardar relatórios das várias operações na rede, configurar encaminhadores, estabelecer chaves para autenticação de mensagens, etc.

A alta fiabilidade demonstrada, assim como a possibilidade da sua ligação à Internet, aliadas à facilidade de obtenção de todo o material, hardware e software, necessário ao suporte do seu desenvolvimento, instalação e gestão, levaram a que a tecnologia LonWorks tivesse sido rapidamente aceite no mercado da automação doméstica.

2.6. ASSOCIAÇÃO KONNEX

A Konnex é uma associação sem fins lucrativos, fundada em Maio de 1999, estando localizada em Bruxelas, sendo composta pelos principais fabricantes de electrónica da Europa, por prestadores de serviços e organizações interessadas. Esta associação foi desenvolvida sobre os mais de 10 anos de experiência obtida pelas normas EIB, Batibus e EHS, sendo estas a base a partir da qual foi desenvolvida a norma KNX.

Figura 2.15 - Associação Konnex

A norma KNX, desenvolvida para a gestão e controlo de edifícios e casas inteligentes, promove os seguintes pontos:

- É totalmente aberta, sem royalties para os seus membros e independente da plataforma;
- Garante um funcionamento com produtos de diferentes marcas e modelos, desde que certificados e identificados pelo logótipo KNX;
- Suporta vários meios de comunicação (cabos de cobre, rede eléctrica, radiofrequência, infra-vermelhos e ethernet) e vários métodos de configuração (software para PC's, dispositivos configuradores e *plug'n'play*).

A associação tem como missão a promoção de uma norma comum e o apoio e encorajamento aos seus membros para desenvolverem e promoverem produtos certificados.

Com o KNX pretende-se expandir o actual mercado, centrado em edifícios inteligentes, para o mercado residencial através de uma tecnologia única, fiável e económica tecnologia, tentando desta forma obter uma maior aceitação do mercado.

O KNX inclui três diferentes meios de configuração: *A-mode*, *E-mode* e *S-mode*. Estes dois últimos exigem o recurso a instaladores certificados para o teste e

configuração da rede doméstica de acordo as exigências do consumidor final. O modo automático (*A-Mode*) está concebido para que os electrodomésticos compatíveis, como por exemplo um frigorifico ou máquina de lavar roupa, possam ser ligados e configurados na rede pelo cliente final, sem recursos a conhecimentos técnicos.

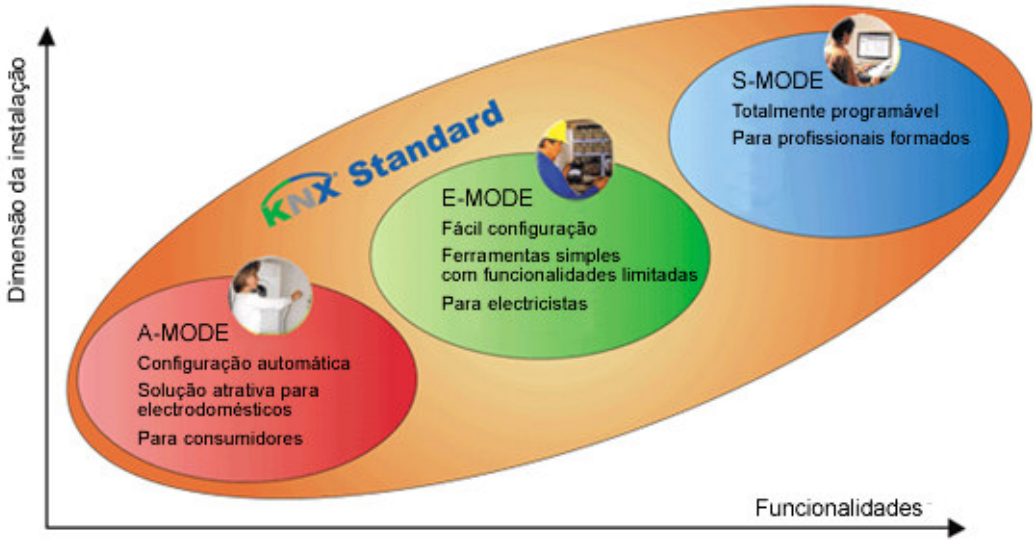

Figura 2.16 - Os meios de configuração do KNX

Actualmente a norma suporta quatro meios de comunicação distintos, facilitando a tarefa dos instaladores em adaptar a rede às condições do local e às diferentes funcionalidades requeridas. Esta flexibilidade aumenta as possibilidades de um técnico em encontrar uma solução que preencha os requisitos sem ultrapassar os limites financeiros impostos pelo cliente.

A norma KNX é baseada na camada de comunicação do EIB, mas aperfeiçoada com a camada física, modos de configuração e aplicação do BatiBUS e EHS.

Resumindo, a associação Konnex pretende com a norma KNX criar uma base tecnológica para massificar o conceito de casas e edifícios inteligentes. Edifícios de escritórios poderão ser mais facilmente adaptados, com um menor custo de manutenção, às necessidades, em constantes mutações, dos seus utilizadores. O edifício tornar-se-á mais racional em termos energéticos. No mercado residencial, para além da flexibilidade introduzida serão possíveis novos serviços para nos facilitar e tornar mais agradável a gestão do nosso lar.

3. TECNOLOGIAS DE IMPLEMENTAÇÃO E SOLUÇÕES ACTUAIS

Com este capítulo pretende-se apresentar as várias plataformas tecnológicas disponíveis para a implementação da aplicação, que se pretende desenvolver, e analisar algumas das soluções já existentes no mercado.

Na análise tecnológica será efectuado um enquadramento histórico da tecnologia actual, abordando-se as principais tecnologias existentes, com especial destaque para o Java, incluindo o RMI, e a plataforma .NET da Microsoft.

Antes de se iniciar o levantamento de requisitos será efectuada, na segunda parte deste capítulo, uma análise e comparação dos produtos de vigilância e controlo remoto disponíveis no mercado e que mais se assemelham à aplicação que se pretende desenvolver.

3.1. TECNOLOGIAS DE PROGRAMAÇÃO DISTRIBUÍDA

No inicio da era informática os computadores eram grandes e muito caros. Foi nesta altura que os *mainframes* tiveram o seu maior sucesso permitindo, de uma forma mais económica, centralizar a capacidade de processamento em apenas um computador (o qual poderia ter mais do que um processador) sendo a sua interacção efectuada através de um ou vários terminais ou de outros computadores com muito menor capacidade de processamento. Na década de 90, com o desenvolvimento tecnológico, a capacidade de processamento aumentou e o custo do equipamento informático baixou drasticamente, tornando possível o aparecimento de computadores a um custo muito acessível, com capacidade para executar várias aplicações. Com esta evolução, as aplicações passaram das *mainframes* para os computadores pessoais, transformando-se em aplicações isoladas, que são executadas sobre um único sistema operativo.

Em meados da década de 90, a Internet veio revolucionar o mundo, ao interligar vários computadores. Esta tecnologia foi de tal modo aceite pelo mercado que rapidamente começaram a ser disponibilizadas aplicações acessíveis através de páginas web. No início, limitavam-se a páginas de comércio electrónico mas, actualmente, grande parte das aplicações passaram do domínio local para a Internet, facilitando o acesso a estas por vários utilizadores. Estas aplicações, de uma forma idêntica aos *mainframes* dos primórdios da informática, são executadas em um ou vários servidores, disponibilizando a interface a uma série de computadores através de páginas web.

Hoje em dia, para além de se pretender que as aplicações possam ser acedidas por diferentes utilizadores, pretende-se também que estas possam comunicar entre si, originando o paradigma das aplicações distribuídas.

Por definição, uma aplicação distribuída é um programa que é executado em diversas máquinas mas que aparece ao utilizador como uma simples aplicação [Wu, 99].

A Internet, ao ser uma tecnologia normalizada, obriga os programadores e empresas que criam tecnologia a preocuparem-se em desenvolver aplicações e soluções que corram na Internet, sendo obrigatoriamente compatíveis com outras linguagens. É este o sentido actual e dois mundos, ainda não necessariamente concorrenciais e incompatíveis, parecem emergir como vencedores: o mais velho, a tecnologia e linguagem Java, desenvolvida pela Sun Microsystems e adoptada pela

maioria dos grandes fabricantes; e o ainda recente .Net, desenvolvido pela Microsoft. Estes prometem substituir, progressivamente e ao longo da presente década, as anteriores tecnologias de desenvolvimento e de integração.

Neste momento coexistem todas as realidades, incluindo os emergentes *Web Services*. E este é um processo que deverá decidir-se nos próximos três a cinco anos mas, até lá, a coexistência entre diferentes tecnologias de desenvolvimento será uma realidade. Afinal, o objectivo é apenas um: criar o *Real Time Enterprise Integration* das aplicações de negócio – um dos maiores desafios da indústria, uma tecnologia neutral, independente, robusta e escalável, que lamentavelmente ainda está por inventar.

O J2EE (*Java 2 Enterprise Edition*) foi desenvolvido segundo a perspectiva do servidor, enquanto o .NET reflecte o lado do cliente. Já os *Web Services*, embora sejam a retoma de um conceito, têm a vantagem de serem neutrais em relação aos componentes e tecnologias de integração, o que permite um evolução permanente, pois o investimento é mínimo e não é necessário mudar as aplicações, a infra-estrutura ou as plataformas.

3.1.1. Aplicações Web

Não poderíamos definir aplicações distribuídas sem referir a sua origem, as aplicações Web. Estas podem não ser um sistema distribuído no verdadeiro sentido da palavra, mas a sua enorme aceitação exigiu o desenvolvimento de soluções para melhorar este sistema computacional.

Uma aplicação *web* é um sistema de informação computacional, normalmente acessível através do protocolo HTTP (*HyperText Transfer Protocol*) e onde a informação trocada se baseia em documentos HTML. No inicio eram apenas um conjunto de documentos informativos e estáticos, onde a única interacção do utilizador se baseava a navegar entre estes. Com o passar do tempo, começaram a aparecer soluções que permitiram oferecer ao utilizador uma experiência mais rica e interactiva.

Uma das primeiras soluções, para o fornecimento de conteúdos dinâmicos sobre o protocolo HTTP foi disponibilizada através do protocolo CGI (*Common Gate Interface*). O funcionamento do CGI baseia-se no tratamento de pedidos HTTP de clientes para a produção de documentos HTML em tempo real, permitindo a criação de conteúdos dinâmicos cujos conteúdos poderão ser fornecidos pela leitura de base de dados externas.

Com o sucesso da *World Wide Web*, a Microsoft lançou o ASP, a sua tecnologia de servidor para a criação de páginas dinâmicas. Esta tecnologia é executada sobre o IIS (*Internet Information Service*), quando é efectuado um pedido de uma página, produzindo dinamicamente o seu conteúdo. Paralelamente foram aparecendo soluções idênticas, algumas em *open-source*, tais como o PHP, sendo outras de fabricantes conhecidos, como o ColdFusion da Macromedia.

3.1.2. CORBA

O CORBA, acrónimo para *Common Object Request Broker Architecture*, é uma especificação de tecnologia aberta e independente da implementação, que é usada para ligar computadores entre si, segundo o paradigma cliente/servidor, através de uma rede de comunicação. Esta tecnologia esquematiza os blocos de dados ou objectos e permite a partilha de capacidades para formar aplicações.

O CORBA ajuda a resolver questões de compatibilidade que se levantam quando os objectos são escritos em linguagens diferentes e correm em plataformas e máquinas diferentes. Desde o aparecimento do CORBA, em 1991, fornecedores e utilizadores têm trabalhado para fazer avançar a especificação através de um fórum aberto fornecido pela OMG (*Object Management Group*).

Esta tecnologia é poderosa, mas pode ser difícil de aprender. O que a torna tão complicada é o facto de permitir que, objectos escritos em linguagens diferentes, a correr em diferentes plataformas e localizados em diferentes lugares, partilhem funções e dados. Para ajudar estes objectos a comunicar existem os ORB's (*Object Request Brokers*), que seguem o padrão CORBA e podem estabelecer ligações entre os objectos.

Usando o protocolo IIOP (*Internet Inter-ORB Protocol*), um programa baseado em CORBA, independentemente da linguagem usada e do sistema operativo onde é executado, pode comunicar com outro programa, também baseado em CORBA.

Os ORB's são construídos especificamente para tratar da interoperacionalidade dos objectos numa rede, mas não é assim tão fácil. Tradicionalmente, o CORBA é de difícil aprendizagem. Por exemplo, os programadores precisam de aprender IDL (*Interface Definition Language*) para descreverem as propriedades e funções de um objecto ao ORB.

Os ORB's comerciais facilitam o processo ao fornecer serviços como a pesquisa automática de objectos e gestão destes. Serviços populares vão eventualmente ser

introduzidos na especificação CORBA, mas a corrida para fornecer novos serviços é o que diferencia os ORB comerciais.

Em empresas que optaram pela plataforma Windows, os programadores usam o DCOM (*Distributed Component Object Model*), da Microsoft. As empresas só com Java usam o RMI (*Remote Method Invocation*). *Bridges* deixam que os programadores combinem as diferentes tecnologias.

O CORBA compete com outras formas de *middleware*. O *Remote Procedure Calls* (RPC) liga entre si programas orientados para não objectos escritos em Cobol ou Fortran.

CORBA é a arquitectura de *middleware* mais antiga, logo a mais madura, e a que teve uma grande aceitação no passado, perdendo gradualmente a sua posição para as restantes tecnologias. Uma das principais razões para a diminuição da popularidade do CORBA baseia-se no facto de esta tecnologia não ter sido preparada para trabalhar com os protocolos de comunicação da Internet, tentando impor, com pouco sucesso, o seu protocolo IIOP como protocolo de comunicação entre aplicações *web*.

Com o passar dos tempos, a arquitectura CORBA cresceu e amadureceu. Na última versão da especificação, para além das funcionalidades básicas e comuns desde a primeira versão, foram acrescentados serviços e funcionalidades que enriqueceram a arquitectura da plataforma. Esta arquitectura é denominada de OMA (*Object Management Architecture*) [Siegel, 00].

Uma das principais vantagens desta tecnologia é o facto de esta ser uma norma da OMG. Isto significa que existe uma organização composta por várias empresas, de vários domínios da indústria, em que esta tecnologia é usada.

3.1.3. Java / Remote Method Invocation RMI

Na sua essência, o Java não é mais do que uma simples linguagem de programação. É uma linguagem orientada por objectos, ligada à rede, interpretada, robusta, segura, neutra face à arquitectura, portátil, de elevado desempenho, *multithread* e dinâmica.

Com efeito, inicialmente o Java começou por ser uma linguagem de base para criar páginas animadas na *World Wide Web* (WWW), que deixaram assim de ser estáticas e monótonas para ganharam vida, com capacidades multimédia e

animação. A partir de 1995, quando foi formalmente apresentada pela Sun, tornou-se uma norma de programação, visto potenciar a todos os utilizadores o acesso a animações, a partir de qualquer plataforma, e a execução de pequenas aplicações *Java applets* em qualquer plataforma hardware, sem passar pelo pesadelo dos *portings* a partir da plataforma que eram desenvolvidas. De repente, construiu-se uma ponte para ligar as ilhas da incompatibilidade umas às outras.

Neste sentido, o Java também pode ser entendido como uma plataforma que suporta uma nova forma de computação, assente no poder da rede e na ideia de que o mesmo software pode correr em diversos tipos de sistemas e de dispositivos (*Write once, Run anywhere*). A execução desta ideia é simples: o mesmo software Java pode correr no dispositivo mais pequeno ou num supercomputador, porque os componentes da tecnologia Java não estão vinculados ao tipo de computador, telefone, televisão ou sistema operativo onde correm; limitam-se a funcionar, desde que a plataforma onde estão suporte Java. Isto é possível graças a uma componente fundamental da tecnologia chamada *Java Virtual Machine* (JVM), uma espécie de tradutor/intérprete que transforma instruções gerais da plataforma Java em comandos específicos e à medida, que vão colocar os dispositivos a executarem as tarefas para as quais foram concebidos.

O *Remote Method Invocation* (RMI) é uma tecnologia desenvolvida e apresentada pela Sun Microsystems, em reposta à procura do mercado de objectos distribuídos, que em 1996 era na sua grande maioria composto por aplicações baseadas em CORBA. O RMI permite que um programa numa *Java Virtual Machine* (JVM) possa fazer a invocação de métodos num objecto localizado noutra JVM, que pode estar localizado numa máquina remota, permitindo aos programadores de Java, e apenas a estes, a possibilidade de utilizar computação distribuída em ambientes de rede.

O *Java Remote Method Invocation* (Java/RMI) é baseado num protocolo chamado *Java Remote Method Protocol* (JRMP). Este, semelhantemente a qualquer linguagem moderna de sistemas distribuídos, é baseado nas *Remote Procedure Calls* (RPC), de onde herdou alguns conceitos.

A *Java 2 Enterprise Edition* (J2EE) foi desenvolvida para oferecer uma plataforma normalizada que, tendo como base a plataforma Java, permite integrar e gerir aplicações empresariais multi-tecnologia orientadas ao serviço. A sua estrutura é dividida em três camadas distintas [J2EESpec, 03]:

- Camada do cliente – É a interface da aplicação. Nesta camada estão incluídos os *web browsers* e aplicações Java, incluindo as *Java applets*;

- Camada *web* – Esta camada fornece serviços a clientes através de um servidor *web*;
- Camada de negócio – Esta é a camada que contém a lógica da aplicação.

Resumindo, o Java RMI é uma tecnologia proprietária que permite implementar sistemas distribuídos, orientados aos objectos, independentemente da plataforma de *hardware* ou do sistema operativo.

Apesar da tecnologia RMI não estar normalizada, o protocolo IIOP, com o qual se integra de uma forma transparente, torna-a interoperável com outras tecnologias normalizadas, como objectos, componentes e serviços CORBA, e mesmo proprietárias, como objectos COM.

3.1.4. *Distributed Component Object Model* (DCOM/COM+)

O DCOM (*Distributed Component Object Model*) é o sistema da Microsoft para distribuir uma aplicação por mais de um computador numa rede. Este sistema foi desenvolvido a partir de outras tecnologias, com especial relevância para as tecnologias COM (*Component Object Model*) e OLE (*Object Linking and Embedding*).

OLE é principalmente uma biblioteca de interfaces predefinidas, disponibilizando interfaces para, por exemplo, permitir a interacção entre folhas de cálculo e documentos de texto. Por seu lado, o COM disponibiliza um meio para criar serviços extensíveis, denominados de componentes. À medida que se adicionam novas funcionalidades, os serviços que estes providenciam mantêm-se retro-compatíveis com componentes antigos, não precisando de ser actualizados.

DCOM é a versão distribuída do modelo COM. Embora nas primeiras versões da tecnologia COM não existisse suporte para sistemas distribuídos, o projecto para objectos distribuídos já estava incluído. O COM foi pensado desde o início para a computação distribuída, podendo usar-se um objecto COM como um servidor DCOM. A maior diferença entre COM e DCOM é o facto de este último usar *Object Remote Procedure Call* (ORPC) em vez do *Lightweight Procedure Call* (LPC) para comunicações entre componentes [MacCarty, 99].

Um componente de software numa máquina pode usar o DCOM para enviar uma mensagem, que se denomina por chamada de procedimento à distância, para um componente numa máquina diferente. O DCOM faz automaticamente a ligação, leva a mensagem e devolve a resposta do componente remoto.

O DCOM, que foi incorporado no Windows NT 4.0 e Windows 98, tem a vantagem de ser fácil de usar. Se os programadores constroem as aplicações do Windows recorrendo aos *ActiveX*, o modo da Microsoft para construir os componentes, o sistema operativo irá automaticamente criar ligações e controlar os componentes, independentemente de se localizarem ou não na mesma máquina.

Às capacidades do DCOM para interligar componentes, a Microsoft acrescentou características relacionadas com o Windows, incluindo o *Microsoft Transaction Server*, para executar transacções de bases de dados através da Internet, e o COM+ (*Common Object Model*), que simplifica ainda mais a programação distribuída com características como as bases de dados em memória.

3.1.5. A plataforma .NET

.NET é uma plataforma criada pela Microsoft que liga informação, pessoas, computadores e dispositivos por meio de *Web Services*. De um ponto de vista tecnológico, o .NET é uma arquitectura de tecnologia centrada na Internet que define como as diferentes aplicações, fontes de dados ou dispositivos podem integrar-se e partilhar informação entre si.

O .NET não é um novo tipo de interface de utilizador ou uma peça de aplicação de negócio que um utilizador necessite de aprender ou com que se tenha de preocupar. Mesmo se houver algum componente de interface de utilizador para o .NET - algo que a Microsoft define como experiências de utilizador, o .NET é uma camada tecnológica que fica sob a capa da aplicação e lhe permite trocar informação com outras aplicações ligadas ao .NET por meio de um nível de comunicações baseada na Web. O .NET está construído sobre XML (*Extensible Markup language*), uma norma da indústria de aplicações, o que representa o empenho da Microsoft no XML implementado num ambiente de Internet.

Esta plataforma é composta em duas partes fundamentais: o *common language runtime* (CLR) e a .NET *framework*, que juntas são uma implementação da norma ECMA *Common Language Infraestructure* (CLI) [CLISpec, 02]. Analisando a plataforma .NET, podemos facilmente encontrar algumas semelhanças entre o CLR e o *Java Virtual Machine*, e também entre a *.NET class library* e o *standard developement kit* (SDK) da Sun Microsystems.

O .NET inclui vários componentes de software usados para construir aplicações e serviços. Estes componentes incluem, por exemplo: ferramentas de programação *Visual Studio .NET* e a *.NET Framework*; o software de cliente .NET para os

ambientes Windows XP ou Windows CE; um conjunto de *Web Services* incluindo o *MapPoint .NET* para informação baseada na geografia; e um conjunto de aplicações e/ou serviços que a Microsoft chama "experiências", que incluem produtos ligados ao .NET como o Office, MSN, *Microsoft Great Plains Business Solutions* e outros. Mais uma vez, a organização do utilizador final não precisa de se preocupar com os componentes do .NET em si, já que o .NET deve ser visto como uma aplicação distribuída que permite *Web Services*. Neste ambiente *de Web Services* - que as tecnologias devem definir como um modelo de computação distribuída rico e acoplado de forma solta - a informação pode ficar disponível a qualquer aplicação ou a qualquer tipo de dispositivo.

O que é único no uso destes serviços é que as aplicações ligadas pelo .NET irão de forma automática descobrir, ligar-se e colaborar entre si, tornando os trabalhadores que lidam com conhecimento/informação mais produtivos e informados.

3.1.6. Web Services

No início o objectivo da Internet era permitir o acesso fácil a informação por parte de uma grande comunidade. Com o aumento da popularidade e utilização desta tecnologia, procuraram-se novas funcionalidades, tais como o acesso remoto a base de dados de empresas e o comércio electrónico. Hoje em dia estamos a chegar a uma fase na qual a Internet, que era um simples meio de informação, está a evoluir para uma Internet de serviços, onde as empresas criam e disponibilizam modelos de negócio a outras empresas, para que estas os possam consultar ou mesmo incluir nos seus próprios modelos de negócio.

Um *Web Service* é uma peça de software modular, contida e auto-descrita, que publicita as funcionalidades dos seus métodos numa forma normalizada e bem definida. A descrição dos seus métodos e propriedades pode ser disponibilizada a pedido ou distribuída numa base de dados que toma a forma de uma espécie de páginas amarelas.

Os *Web Services* são baseados em normas, como XML e HTTP, que são mantidas e geridas pelas organizações do *World Wide Web Consortium* (W3C) e a *Organization for the Advancement of Structured Information Standards* (OASIS). O protocolo de comunicação usado é o SOAP (*Simple Object Access Protocol*), sendo usado o WSDL (*Web Services Definition Language*) para a definição dos serviços.

O SOAP é um protocolo de comunicação, ou mais precisamente, de transporte de dados, em que estes são formatados através de documentos XML.

A promessa dos *Web Services* é um mundo de cooperação de serviços onde os componentes são construídos de uma forma rápida, fácil e de baixo custo, criando uma rede de serviços que permitem criar aplicações dinâmicas.

3.1.7. Conclusão

Apesar da plataforma CORBA ter sido a mais usada no início, recentemente o mercado de aplicações empresariais tem vindo a optar por outras soluções. É no entanto uma plataforma de *middleware* madura e definida a partir de normas e não será extinta tão cedo, sendo actualmente bastante usada como meio de integração em servidores de aplicações J2EE.

A Sun Microsystems, no início, desenvolveu a RMI como um protótipo de comunicação proprietário que, com o tempo e apoio incondicional de uma comunidade de código aberto, foi crescendo, vindo mais tarde a ser integrada pela J2EE.

A *Microsoft*, tendo como base o seu popular sistema operativo, ofereceu soluções proprietárias. Estas soluções têm como ponto forte a enorme facilidade de uso mas, por outro lado, forçam os utilizadores e empresas a usar o *software* da Microsoft. Esta razão levou a que parte do mercado procurasse soluções alternativas. Tendo em conta este factor, a Microsoft desenvolveu e lançou no mercado a plataforma .NET que oferece uma arquitectura que permite que esta aplicações possam ser implementadas noutras plataforma não Windows.

Recentemente tem aparecido no mercado um novo paradigma de programação distribuída, denominado *Web Services*. Esta é uma das mais promissoras tecnologias de *middleware* actuais, sendo baseada em protocolos comuns tais como o XML e HTTP. Esta tecnologia está orientada para a programação de aplicações globais, distribuídas e orientadas ao serviço.

3.2. ANÁLISE DAS SOLUÇÕES ACTUAIS

Nesta secção serão analisadas algumas das principais soluções existentes no mercado para a vigilância e controlo remoto através da Internet.

As primeiras soluções a surgir no mercado, que permitiam a visualização de imagens através da Internet, foram as WebCams. Estas tiveram grande impacto no final da década de 90, associadas ao grande sucesso da Internet. Com estas soluções massificou-se o acesso a imagens em directo de vários pontos do planeta, aparecendo vários *sites* na Internet que permitiam visualizar desde o estado do trânsito às condições climatéricas de uma remota estância de ski. No domínio particular foram vários os utilizadores que colocaram à disposição dos cibernautas imagens em directo das suas habitações.

Estas soluções são caracterizadas pela sua facilidade de instalação, baseando-se no sistema *plug'n'play*, e pelo facto de necessitarem de um computador e de um serviço de alojamento para a colocação das imagens. Em termos técnicos, estas soluções são compostas por uma ou várias câmaras com ligação USB, ligadas a um computador que, em intervalos de tempo predeterminados, envia as imagens para um servidor web. Este envio (*upload*) é normalmente efectuado com recurso ao protocolo FTP (*File Transfer Protocol*), podendo-se também configurar o software para enviar as imagens via *e-mail*. Mais recentemente foram introduzidas algumas melhorias nestas aplicações as quais permitem, através de uma análise digital das imagens capturadas, verificar a existência de movimento e, caso seja detectado, despoletar uma sequência de acções, tais como o envio de um *e-mail* com uma ou várias imagens capturadas em anexo.

Com a evolução dos actuais sistemas de vigilância analógicos, vulgarmente designados por CCTV (circuito fechado de televisão), apareceram no mercado soluções profissionais que permitiam a visualização remota das imagens capturadas por câmaras analógicas. Estas soluções são baseadas em hardware específico que, não só deve ser instalado no local a vigiar, com também é necessário o seu uso para a visualização remota. Este hardware funciona normalmente sobre RDIS, a qual permite uma melhor qualidade de serviço pois usa uma comutação de circuitos em vez da comutação de pacotes usada nas redes IP. Estes sistemas profissionais são caracterizados por permitirem a visualização de vídeo com uma boa qualidade de imagem e rápido refrescamento de imagens (entre 1 a 10 imagens por segundo), associado a um elevado preço de material e custos elevados na visualização, pois torna-se necessário manter uma ligação telefónica para a

visualização das imagens.

Com a massificação das redes Ethernet, começaram a aparecer sistemas de vigilância baseados em redes IP. Estes sistemas tiram partido das redes já existentes numa empresa para transmitir as imagens capturadas por uma câmara para qualquer computador. Esta solução obriga o uso de câmaras específicas, ou de câmaras convencionais (analógicas) ligadas a vídeo-servidores, que transformam o sinal analógico capturado num sinal digital e o transmitem através de uma saída de rede. Para a visualização destas imagens é normalmente usado um software específico que deverá ser executado num PC.

Mais recentemente têm aparecido no mercado câmaras de rede e vídeo-servidores que já incluem um servidor web interno, permitindo visualizar as imagens capturadas através de um browser. Apesar de normalmente serem usadas numa rede local também estão pensadas para permitirem o acesso remoto via web. Como desvantagem, estas soluções obrigam a que o IP do local a visualizar seja conhecido, obrigando a um custo extra na atribuição de um IP fixo em todos os locais que se pretende vigiar.

Todas as soluções apresentadas anteriormente apenas permitem a visualização de imagens ou vídeo, mas nenhuma permite a integração com sistemas de segurança e domótica. Só muito recentemente, com a evolução das tecnologias para casas inteligentes, é que têm aparecido soluções para controlo de redes de domótica. Algumas dessas soluções são o *DomoPort*, para controlo de redes EIB, o *HomeSeer* e o *InControl*, para controlo de uma rede X10, e a *Xanboo*, a qual usa um sistema sem fios proprietário.

DomoPort

O *DomoPort* é um produto profissional, vocacionado para edifícios com instalações EIB, que disponham de um módulo de comunicação via GSM, RDIS ou Ethernet compatível com a solução. Este produto além de permitir o controlo de aparelhos disponibiliza a funcionalidade de visualização de 2 câmaras USB, funcionalidade esta que ainda se encontra em fases de teste, não existindo ainda nenhum produto final com estas funcionalidades.

A solução *DomoPort* é composta por uma aplicação de servidor, acessível remotamente a partir do link www.domoport.com, e um módulo de comunicação local, o qual recebe os comandos do servidor e os reenvia para os respectivos módulos EIB. Esta solução permite o acesso a gráficos de temperatura, entre

outros, assim como a leitura do estado dos equipamentos EIB.

Como principal vantagem o *DomoPort* apresenta uma grande fiabilidade, tendo como desvantagens o custo e o facto de obrigar ao uso de redes telefónicas ou redes de banda larga com IP fixo.

Homeseer

O *Homeseer* é um software que se instala no computador do utilizador e o transforma num servidor web com possibilidade de controlar os dispositivos X10 configurados. Esta solução é a mais económica, necessitando apenas de um interface de comunicação com a rede X10. Como desvantagens podemos salientar os factos de obrigar ao uso de um PC, de uma linha com IP Fixo e do utilizador precisar de ter alguns conhecimentos técnicos. Esta solução não permite a visualização de imagens.

InControl

O *InControl* é a solução existente que mais se aproxima dos requisitos pretendidos com a presente tese. Esta solução é baseada num software, que deve ser instalado no local a controlar, e numa aplicação de servidor. O software é instalado no computador do cliente e, através do uso de um hardware próprio, permite a aquisição de imagens de câmaras do mesmo fabricante, estando limitado a 16 unidades. Para além da visualização do espaço o *InControl* permite também o controlo de aparelhos através do protocolo X10.

Este produto já tem implementado mecanismos que permitem o uso de acessos de banda larga que não dispõem de IP fixo. A principal desvantagem desta solução é o facto de obrigar a usar as câmaras do fabricante tornando a aplicação pouco flexível e evolutiva.

Xanboo

A linha de produtos *stand-alone* da *Xanboo*, a qual inclui câmaras de vigilância, sensores sem fios, termóstatos e actuadores, permite a visualização e o controlo remoto através de um interface Web.

Esta solução é baseada em produtos proprietários, desenvolvidos pela *Motorola*, que devem ser instalados no local a vigiar/controlar. No centro da aplicação pode ser usado um PC ou uma *gateway* que, através de comunicações sem fios, permite a comunicação com os diferentes dispositivos. Para poder visualizar e controlar o seu espaço, o utilizador deve aceder ao portal disponibilizado pela empresa e inserir as suas credenciais.

Recentemente, em parceria com a *Shell Oil*, a empresa lançou uma nova linha de produtos, denominada *Home Genie*.

Conclusões

No mercado já existem algumas soluções que permitem a visualização e o controlo remoto de aparelhos, mas nenhuma foi desenvolvida sobre um conceito de modularidade e compatibilidade com *hardware* de diferentes fabricantes. Após o uso de algumas das soluções verifica-se também que o nível de conhecimento técnico exigido para o seu uso e instalação é bastante elevado.

À excepção do *DomoPort* e da *Xanboo* as restantes soluções apresentam-se como aplicações que devem ser instaladas pelos utilizadores no computador local.

4. ESPECIFICAÇÃO E ARQUITECTURA

Antes de se iniciar o desenvolvimento de qualquer projecto, deverá ser efectuado um rigoroso levantamento dos requisitos pretendidos com a aplicação. Na primeira parte deste capítulo serão apresentados os requisitos funcionais e não funcionais, sendo processados usando a linguagem UML como modelo de especificação.

Na segunda parte deste capítulo será analisada a interface da aplicação e a sua arquitectura, nomeadamente a sua estratificação por camadas.

4.1. DEFINIÇÃO DOS REQUISITOS

Conforme referido no primeiro capítulo desta dissertação, pretende-se que um utilizador, sem a necessidade de instalação de nenhum software, possa visualizar através da Internet, em tempo-real, um ou vários locais onde dispõe de uma câmara, ligada ou não a um computador, controlar os dispositivos eléctricos desse mesmo local e actuar sobre o sistema de segurança. Como requisitos mínimos para o local a vigiar/controlar, pretende-se apenas que seja necessário uma vulgar ligação de banda larga à Internet e um dispositivo para a aquisição de imagens/vídeo e/ou controlo de aparelhos.

4.1.1. Requisitos Funcionais

Antes de iniciarmos a descrição dos requisitos funcionais para esta aplicação iremos primeiro definir quais os tipos de utilizadores, ou actores, usando a terminologia do UML, que irão interagir com a mesma. Podem ser definidos 4 tipos:

- **Utilizador final** – Esta categoria engloba todos os utilizadores do sistema que correspondem a clientes finais. Todas as interacções anteriormente descritas representam a sua interacção com o sistema;
- **Utilizador assistente** – Os utilizadores desta categoria são vistos pelo sistema como técnicos de manutenção. As suas funções estão relacionadas com suporte técnico e resolução de problemas dos utilizadores finais;
- **Utilizador revendedor** – Os revendedores são utilizadores especiais que têm como função gerir uma carteira de clientes (utilizadores finais). Podem criar pseudo produtos baseados no serviço, registar utilizadores no sistema e manter um conjunto de técnicos responsáveis pela manutenção. Realça-se o facto de cada revendedor apenas ter permissões para gerir os seus próprios clientes e, os assistentes afectos a um revendedor, apenas dão assistência técnica aos clientes desse mesmo revendedor. Tendo em conta a vertente comercial, todos os clientes criados por um revendedor deverão ter a sua interface adaptada visualmente à imagem do revendedor;
- **Utilizador administrador** – Só deverá existir um único utilizador nesta categoria. Este utilizador tem todas as capacidades dum revendedor acrescidas da possibilidade de gestão dos revendedores e consulta de estatísticas em tempo real relativas ao estado do serviço.

Com a definição de vários tipos de actores fica implícito que a aplicação deverá

permitir a autenticação de cada um através de um interface que permita a introdução de credenciais.

Dado o carácter internacional desta aplicação, a cada utilizador final deverá ser atribuído um idioma, no qual deverá ser disponibilizada toda a informação.

Para um utilizador final deve ser possível visualizar uma ou várias câmaras, podendo alternar entre elas ou visualizar 4 câmaras em simultâneo (*quad cam*). Para cada câmara deve ser indicado o estado da mesma e, caso esta o permita, o utilizador deverá poder capturar uma imagem e controlar a posição e o nível de zoom.

Em termos de controlo de domótica, e caso o utilizador disponha de algum dispositivo de controlo X10, devem ser listados os aparelhos configurados no sistema e, para cada um desses aparelhos, deve ser possível controlar o seu estado (ligado, desligado). Na Figura 4.1 está representado um *Use Case* de interacção do utilizador com a página principal da aplicação.

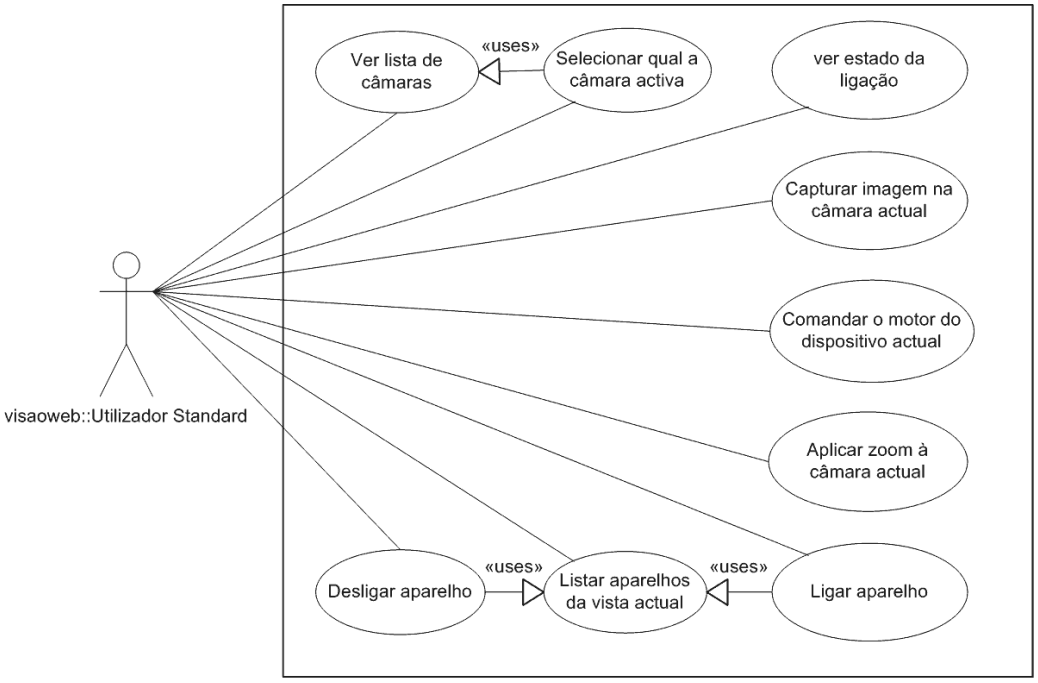

Figura 4.1 - *Use Case* de interacção com a página principal

Para a consulta das imagens capturadas manual ou automaticamente, deve ser disponibilizada uma área que, de uma forma fácil, permita listar as imagens associadas ao utilizador e filtrá-las de acordo com as variáveis de pesquisa: data e hora, câmara e tipo de evento. Para cada imagem deverão ser indicados a data/hora de captura, a câmara que a capturou, o tipo de evento e o nome. Este último é um

parâmetro que o utilizador poderá alterar.

Para a gestão das imagens guardadas deverão existir ferramentas que permitam apagar todas ou algumas das imagens do utilizador. Também deverão ser implementadas funções que permitam a gravação para o disco, impressão, envio por e-mail e protecção de cada imagem.

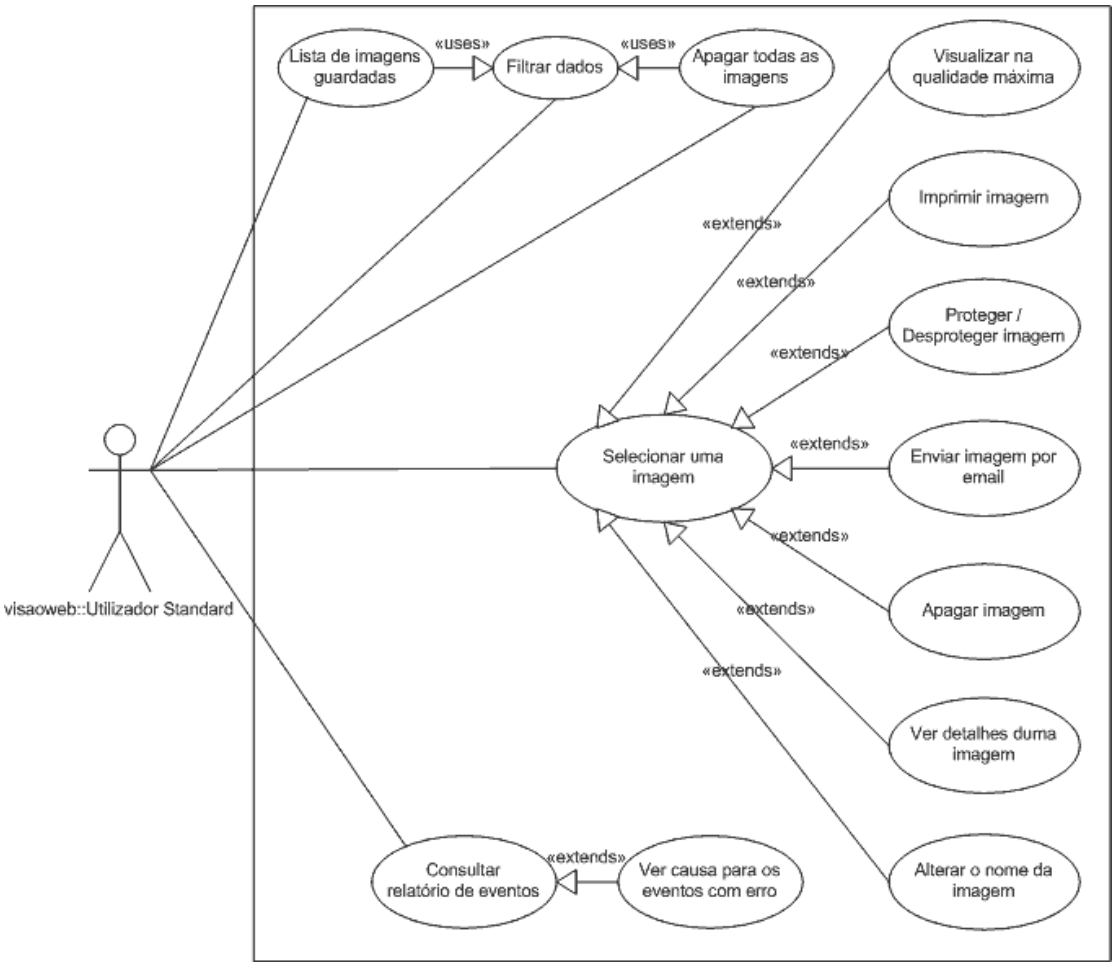

Figura 4.2 - *Use Case* da área de histórico

Dado que, neste tipo de aplicações, as mesmas credenciais podem ser usadas por diferentes indivíduos, deverão ser registados e fornecidos ao utilizador os principais eventos, como por exemplo, a data e hora da entrada e saída na aplicação, a captura ou remoção de uma imagem, etc. Esta área, denominada relatório de eventos, deve permitir ao utilizador efectuar uma pesquisa sobre todos os registos e imprimir os resultados.

Para permitir a um utilizador final a gestão dos dispositivos devem ser-lhe disponibilizadas funções que lhe permitam adicionar, alterar e remover dispositivos,

quer estes sejam câmaras, controladores X10 ou qualquer outro tipo de dispositivo com interface IP. Como forma de impedir que um utilizador possa alternar facilmente entre câmaras e visualizar mais câmaras do que o seu produto permite, o campo *Mac-Address* não poderá ser alterado. Na área das configurações também deve ser permitido ao utilizador gerir os módulos X10 de uma forma semelhante à gestão dos dispositivos.

Cada utilizador final deve poder alterar os seus dados pessoais, alterar a palavra-chave e escolher o idioma no qual deve ser disponibilizada a interface da aplicação.

Para não tornar esta dissertação muito extensa apenas está representado na Figura 4.4 o *Use Case* para gestão de câmaras, sendo o diagrama de gestão de módulos de domótica muito semelhante.

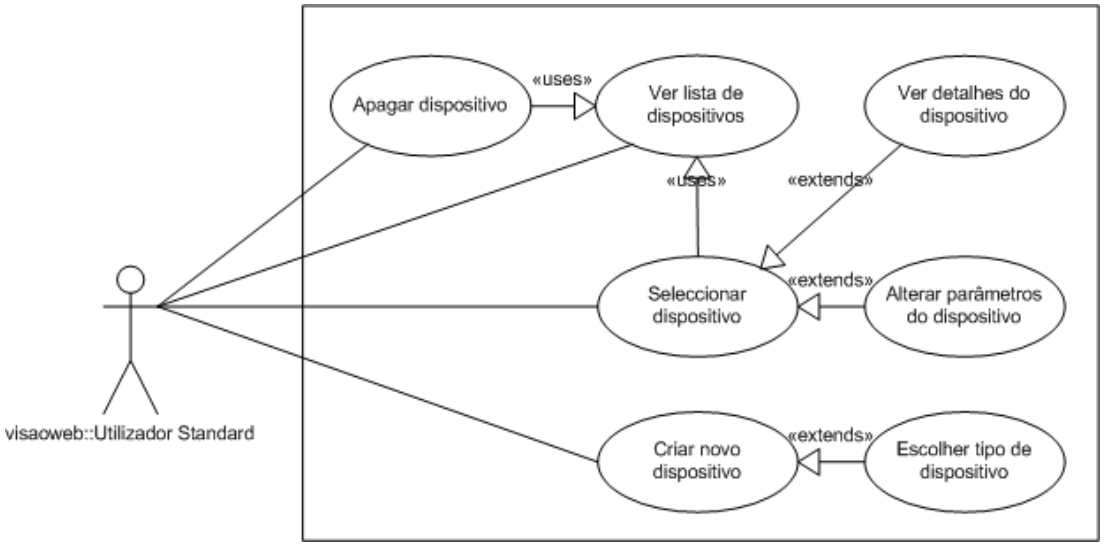

Figura 4.3 - *Use Case* de gestão de câmaras

Um dos principais requisitos da aplicação é a possibilidade de criação de eventos. Um evento é caracterizado como uma sequência de acções que é executada de acordo um horário pré-estabelecido ou após a ocorrência de um alarme de um dos dispositivos. A acção de um evento pode ser composta por uma ou mais acções específicas, tais como captura de imagem, controlo de aparelho ou pausa, sendo a sua ordem definida pelo utilizador. No final de cada evento o utilizador terá a escolha de ser ou não notificado via SMS, MMS ou E-mail. Nos dois últimos casos deve ser enviada em anexo a imagem capturada pela câmara, caso esta tenha sido uma das acções escolhidas pelo utilizador.

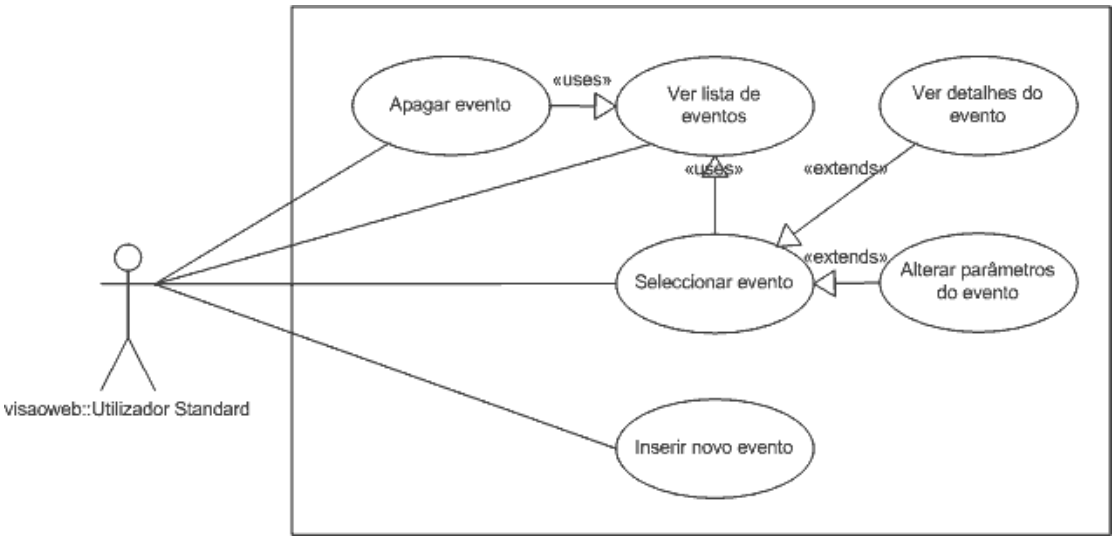

Figura 4.4 - *Use Case* de gestão de eventos

De forma a permitir a criação de diferentes planos de comercialização da aplicação, esta deverá estar preparada para associar a cada utilizador um produto diferente, sendo este produto caracterizado pelo número de câmaras, número de imagens guardadas e/ou número de eventos.

Conforme referido inicialmente, a aplicação deverá ser compatível com dispositivos móveis. Nestes dispositivos e tendo em conta as suas limitações, apenas interessa disponibilizar ao utilizador as funcionalidades de ver, alternar e controlar câmaras, controlar dispositivos X10 e aceder ao histórico de imagens. A figura 4.5 representa o *Use Case* da interacção de um utilizador através de um dispositivo móvel.

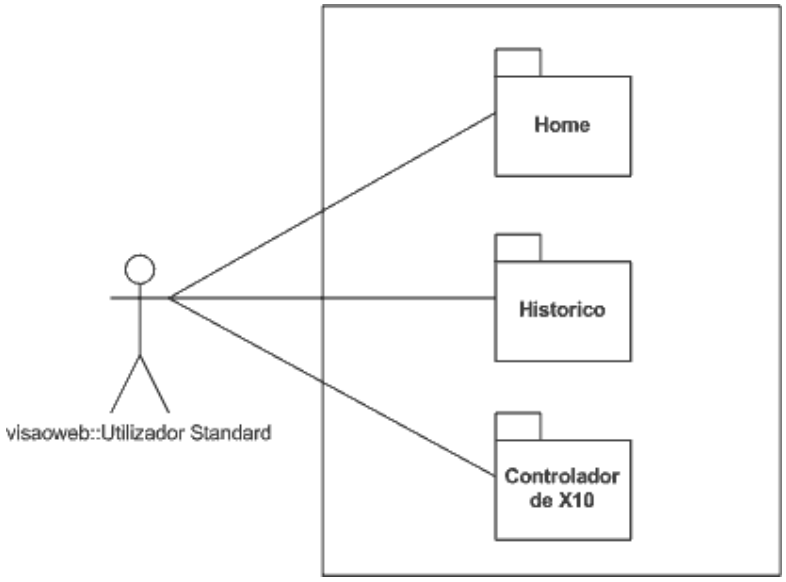

Figura 4.5 - *Use Case* da interface móvel

Um dos requisitos comerciais da aplicação é que esta possa ser comercializada por diferentes revendedores ou parceiros. A cada revendedor deve então ser permitida a gestão de utilizadores finais. Nestes casos a interface destes deve estar visualmente associada ao revendedor através da disponibilização do logótipo do mesmo.

Para a gestão de toda a aplicação deve ser disponibilizada uma área de *back-office*, que permita a um administrador gerir revendedores e que estes possam gerir produtos, utilizadores e assistentes técnicos. Finalmente, esta área deve fornecer a um assistente técnico as ferramentas necessárias para o diagnóstico e a resolução de problemas associados aos utilizadores.

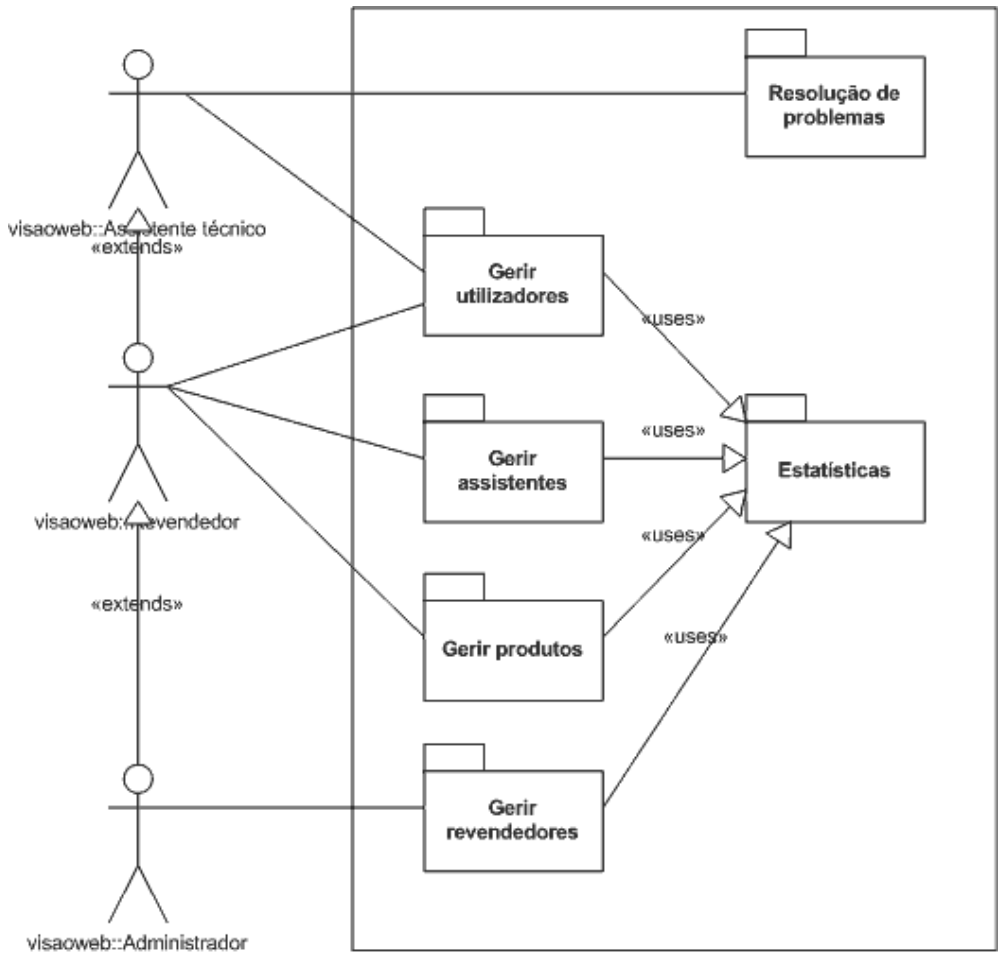

Figura 4.6 - *Use Case* da área de administração

4.1.2. Requisitos Não Funcionais

Usabilidade

Sendo o serviço destinado a utilizadores "normais", sem necessidade de quaisquer conhecimentos técnicos específicos, a usabiblidade é uma questão fundamental. Todas

as interfaces devem ser intuitivas e desenhadas segundo os mesmos padrões de acesso para facilitar a navegação e criar a habituação à aplicação. As interfaces com o utilizador devem ser simples e nunca devem conduzir a ambiguidades na escolha de opções e a ajuda de contexto deve estar sempre presente. No caso particular das interfaces web, devem ser acompanhadas sempre por um sistema de navegação que permita ao utilizador chegar rapidamente a todas as funcionalidades com um número mínimo de interacções com o rato.

Compatibilidade

De forma a distinguir esta aplicação das existentes, um dos principais requisitos não funcionais da mesma é o da compatibilidade. Desta forma pretende-se que a aplicação possa ser acedida a partir de qualquer computador, a correr qualquer sistema operativo. Da mesma forma também deve poder ser executada em qualquer dispositivo móvel, tendo como requisitos mínimos um ecrã policromático e um acesso Wap.

Para responder às necessidades de compatibilidade, pretende-se que a interface da aplicação seja composta por páginas HTML ou WML, devendo estas poder ser executadas em qualquer browser.

Uma das limitações técnicas na instalação de várias câmaras no mesmo local, associadas à mesma ligação à Internet, é que obriga ao uso de portos não normalizados. Neste caso, se o utilizador remoto estiver ligado através de uma *firewall*, esta poderá bloquear o acesso aos portos das câmaras, impedindo o acesso às mesmas. Pretende-se que através da aplicação seja removida esta limitação, podendo o utilizador comunicar com os seus dispositivos através do porto 80 (porto predefinido para comunicações HTTP), mesmo quando estes estão configurados localmente em portos diferentes.

Modularidade

Conforme foi referido inicialmente, esta tese não inclui o projecto e a criação de nenhum hardware para aquisição de imagens e controlo de redes de domótica. Tendo em conta este facto facilmente se compreende a necessidade de ter uma aplicação modular que rapidamente possa ser adaptada a qualquer dispositivo, seja este uma câmara de rede, uma consola de segurança ou um controlador de domótica. O requisito fundamental para estes dispositivos é a disponibilização de uma interface de rede que permita uma comunicação sobre o protocolo TCP/IP.

Inicialmente apenas foi pensada a interacção com câmaras, alarmes e dispositivos

domóticos compatíveis com a norma X10, mas pretende-se, num futuro próximo, alargar a compatibilidade para redes EIB.

Fiabilidade

Um dos objectivos desta aplicação é o da vigilância profissional de instalações. Neste tipo de aplicações não pode haver informação menos correcta, sob pena do próprio sistema ser posto em causa. Nestes termos será sempre preferível que o sistema não produza nenhuma informação a apresentar informação errónea. Por outro lado se for dado um comando ao sistema espera-se que este seja executado de acordo com o previsto, não acontecendo este comportamento normal o utilizador deve ser notificado.

Desempenho

O desempenho num sistema de vigilância e controlo remoto é um factor crítico. É fundamental que imagens emitidas quase em tempo real não sofram um atraso demasiado grande devido a tratamento no servidor. De igual modo, cada vez que um comando é enviado pelo utilizador, este deve ser executado em tempo útil, incluindo o tempo de feedback ao utilizador.

Escalabilidade

Prevê-se que, uma vez iniciada a comercialização do serviço, este esteja em constante crescimento. Como uma das suas funcionalidades principais assenta na transmissão de vídeo em tempo real para múltiplos clientes, o crescimento implica um aumento significativo no tráfego nos servidores. Sabendo que o sistema deverá crescer significativamente, este deverá ser desenvolvido de forma a minimizar as alterações à sua estrutura.

Manutenção

Dada a grande dimensão e complexidade do serviço, cuidados especiais relacionados com a manutenção deverão ser tidos em conta. Deverão existir procedimentos organizados para detecção e correcção de problemas, registos de erros ocorridos e registos referentes ao estado actual do serviço. Todas as configurações inerentes ao serviço deverão estar separadas da implementação e ser facilmente alteradas.

Para facilitar a inserção de novos idiomas, deve ser implementado um mecanismo que permita de uma forma rápida e fácil a tradução de todos os conteúdos.

4.2. REQUISITOS DE HARDWARE

O desenvolvimento do hardware necessário, no lado do cliente, para a visualização e controlo está fora do âmbito desta dissertação, devendo a aplicação ser compatível e facilmente adaptável para diferentes hardwares que possam servir de *gateways* para o ambiente domótico ou fornecer vídeo ou imagens. Neste caso, o principal elemento de hardware necessário para o desenvolvimento da aplicação é, sem dúvida, a câmara, que poderá ser de três tipos:

WebCam – É caracterizada pela sua facilidade de instalação, pois baseia-se no sistema *plug'n'play*, e pela possibilidade de permitir um acesso fácil de qualquer programa informático às suas imagens. A estas câmaras deverá estar sempre associado um PC;

Câmara de rede – Este dispositivo já inclui um controlador TCP/IP com uma ligação Ethernet (RJ45) o que torna esta câmara num dispositivo *stand-alone* – não necessita de um PC para o seu funcionamento;

Câmara analógica – Este tipo de câmaras é o mais vulgar e caracteriza-se por dispor de uma ou várias saídas analógicas do tipo BNC. A este tipo de hardware terá de ser obrigatoriamente associado um vídeo-servidor, que transforme os sinais analógicos capturados pela câmara e os codifique segundo uma norma de compressão de vídeo ou de imagens.

Qualquer um dos tipos de câmara atrás referidos poderão dispor de características particulares, como por exemplo, o suporte de IP dinâmico, a filmagem a cores ou preto e branco, a sensibilidade à luz – incluindo visão nocturna, zoom digital ou analógico, motorização PTZ, entradas para ligar sensores de movimento, saídas para controlar aparelhos e uma porta de série para ligar o dispositivo a uma rede de domótica.

Após uma pesquisa na Internet e a participação em algumas feiras da especialidade, nomeadamente na CeBit 2004, em Hannover, foram seleccionados os seguintes produtos, com os quais a aplicação deverá ser logo de início compatível. A Tabela 5 resume as suas características e funcionalidades.

Tabela 5 – Comparativo das funcionalidades dos dispositivos usados

Compatibilidade Dispositivo	Imagem	Vídeo	IP Dinâmico	PTZ	X10	Sensores	Alarme	Gravação local
Axis 210	X	X(2)	X	O	O	X(3)	O	O

Axis 2130	X	X(1)	X	X	O	X	O	O
Pixord 4000	X	X(1)	X	O	O	X(3)	O	O
Axis 2460	X	X(1)	X	O	X(4)	X	O	X
Alarme Jablotron	O	O	X	O	X	X	X	O
Software Local	X	X	X	O	X	X(3)	O	X(4)

(1) Necessário um ActiveX – Apenas MotionJpeg

(2) Permite streaming em MPEG4

(3) Inclui detecção de movimento digital

(4) Funcionalidade ainda não implementada

As câmaras da Axis são das melhores câmaras do mercado para o desenvolvimento de software de vídeo-vigilância. Os produtos desta marca têm incorporado um *webserver* a executar uma versão reduzida do sistema operativo *Linux*, no qual poderemos inserir pequenos programas desenvolvidos em *PHP* ou *Shell Scripts*. Esta câmara disponibiliza uma API HTTP, a qual permite que o interface do cliente capture directamente a imagem da câmara sem que esta passe pelo servidor e sem a necessidade do uso de componentes *ActiveX* ou *Java Applets*. A existência de uma porta de série permite que no futuro seja acoplada uma interface de comunicação com a rede X10.

O modelo 2130 diferencia-se dos restantes pelo motor que possui, permitindo movimentar a câmara segundo os eixos horizontal e vertical, assim como aproximar ou afastar a imagem. Esta câmara é bastante completa e inclui também funções avançadas que permitem ao dispositivo memorizar um determinado número de posições, permitindo alternar facilmente entre posições.

O Pixord, modelo 4000, é um vídeo-servidor com quatro entradas nas quais se pode ligar qualquer tipo de câmaras analógicas. Esta solução é bastante económica quando se pretende no mesmo local colocar três ou mais câmaras. Este produto contem 4 entradas universais, nas quais se podem ligar sensores tradicionais, permitindo alertar a aplicação quando uma das suas entradas é alterada. Caso se pretenda é possível configurar o vídeo-servidor para efectuar uma detecção de movimento digital, individual para cada câmara, que permite despoletar um evento na aplicação.

O modelo 2460 da Axis é um vídeo-servidor, também de quatro entradas analógicas, que se distingue do anterior por dispor de discos internos, permitindo a gravação local permanente dos vídeos obtidos a partir das câmaras analógicas. Também disponibiliza entradas universais mas não implementa a funcionalidade de detecção de movimento digital. Este equipamento permite ao utilizador pesquisar e consultar remotamente

todos os vídeos armazenados nos seus discos internos

Para podermos fornecer o serviço de segurança ao cliente temos que incluir uma consola de alarme que disponha de um comunicador TCP/IP. Inicialmente não foi encontrada no mercado nenhuma consola de alarme com estas características, lançando-se o desafio a uma empresa portuguesa, sedeada em Braga, representante das consolas Jablotron, para desenvolver o comunicador TCP/IP. Essa empresa, em parceria com a Universidade do Minho, desenvolveu uma placa, a qual pode ser adicionada à consola de alarme, que permite através de pedidos GET o controlo do seu estado (armar, desarmar, etc.) e a leitura de todas as variáveis (estado da consola, dos sensores, do estado das baterias, etc.). Tendo sido esta placa desenvolvida totalmente de raiz foram também incluídas as funcionalidades para controlo de redes X10 e suporte de redes com IP dinâmico.

O software local não é propriamente um dispositivo, mas antes uma solução baseada em câmaras USB, interfaces para PC e um software que deverá ser executado no computador do cliente. Este software será o responsável por capturar imagens através de uma câmara USB e controlar aparelhos através da interface X10 para PC, sempre que solicitado pelo servidor. A descrição do desenvolvimento desta aplicação não será incluída na presente tese.

Conforme já referido anteriormente, para além do hardware mencionado é necessário que o cliente disponha de um acesso à Internet permanente, preferencialmente com uma largura de banda de *upload* mínima de 128 Kbps. Esta linha não necessita de ter um IP fixo, ficando da responsabilidade da aplicação a manutenção do IP actual do dispositivo.

4.3. INTERFACE COM O UTILIZADOR

A interface é o rosto da aplicação, a componente visual que permite ao utilizador introduzir e receber informação, quer esta seja visual ou textual.

Para o desenvolvimento deste componente, e tendo em conta um dos principais requisitos – a não necessidade de nenhum software no lado do cliente, facilmente poderemos concluir que este deverá ser baseado nas normas W3C de forma a ser compatível com a grande maioria dos browsers de Internet.

4.3.1. Interface web

Dado que a interface pode ser considerada como uma sequência de páginas web, o estudo e implementação destas deve seguir os princípios de usabilidade. O desenvolvimento de um website deve seguir um conjunto de simples regras funcionais [Nielsen, 00]:

a) O menu deverá ser facilmente interpretado como tal e a sua posição deve ser constante e permanente ao longo do site, sendo preferencial a sua colocação no lado esquerdo ou no topo da página;

b) O utilizador deverá saber sempre em que área do site se encontra;

c) Deve-se reduzir a distância em termos de cliques do rato entre a entrada da aplicação e as principais funções disponibilizadas no site;

d) O excesso de informação complica a descoberta da informação essencial.

Entrada da aplicação

A entrada na aplicação deve ser efectuada através de uma página que permita ao utilizador a inserção das suas credenciais. A informação inserida pelo utilizador e posteriormente enviada será encriptada através do uso do protocolo HTTPS e de SSL – *Secure Socket Layer*.

O design aplicado nesta página de entrada, dada esta aplicação ser de carácter comercial, deve usar um esquema de cores idêntico à imagem corporativa da empresa e identificar claramente esta e o nome do seu produto. É importante salientar que a entrada na aplicação poderá, para além desta página, ser efectuada a outros níveis.

Ao entrar na aplicação VisãoWeb, através da inserção do endereço www.visaoweb.com num browser, é apresentado o campo de login (1) e palavra-chave (2). Ambos os campos deverão ser preenchidos com as credenciais do utilizador, atribuídas na

aquisição do serviço, e deverá ser pressionado o botão "Entrar »"

Figura 4.7 - Entrada na aplicação

Para memorizar a palavra-chave de forma a que, em acessos futuros, não seja necessária a sua introdução, deve ser seleccionada a opção "memorizar password" (3). Por razões de segurança esta opção só é recomendada para computadores com apenas um utilizador.

Caso o utilizador não se recorde da sua palavra-chave poderá pressionar o botão "Recuperar password" (4) para reaver a sua palavra-chave de acesso. Para que tal aconteça é obrigatório o preenchimento do campo login (1) antes de pressionar o botão. Esta opção envia uma nova palavra-chave de acesso para a caixa de correio configurada no servidor.

Página principal

Esta será a página principal da aplicação. Aqui o utilizador pode visualizar uma ou várias câmaras, controlar a sua posição, capturar imagens, alternar entre dispositivos e controlar aparelhos e alarmes.

É nesta área que a interface desempenha um papel fundamental pois deve disponibilizar todas as principais funcionalidades da aplicação de uma forma visualmente simples, funcional e intuitiva.

Figura 4.8 - Interface da aplicação

A interface da aplicação está organizada em três partes distintas:

- (1) Menu Principal – Esta secção é constante ao longo de toda a aplicação permitindo o acesso a todas as áreas da aplicação;
- (2) Controlos de aparelhos – Aqui encontram-se listados todos os aparelhos configurados que podem ser simples aparelhos eléctricos, consolas de alarme, equipamentos de controlo de temperatura, entre outros;
- (3) Área central – Nesta área podem-se visualizar e controlar as várias câmaras configuradas.

Esta organização mantém-se constante para as restantes áreas da aplicação. No caso das páginas de histórico e configuração, na área 2 são indicadas pequenas dicas textuais para ajudar o utilizador a usar as diferentes funcionalidades.

Figura 4.9 - Visualização de câmaras

Para facilitar o acesso ao vídeo das câmaras, é apresentado imediatamente o primeiro dispositivo. Para alternar entre câmaras ou dispositivos, é apenas necessário pressionar os botões no topo (1). Caso o utilizador disponha de mais do que 4 dispositivos poderá alternar entre eles através das setas localizadas no canto superior direito (2).

Na visualização de um dispositivo pode-se sempre saber qual é o seu estado, através do campo Estado (3). A um dispositivo poderão estar associados os seguintes estados:

- A ligar… – Este estado indica que a ligação ao dispositivo ainda está a ser efectuada;
- Dispositivo *on-line* – Indica que a ligação ao dispositivo foi correctamente estabelecida;
- Dispositivo *off-line* – Significa que a ligação ao dispositivo não foi efectuada devido a um erro desconhecido. Os estados seguintes representam também um estado *off-line*, mas cuja causa é conhecida;
- Tempo de espera esgotado – Este estado indica que foi atingido o tempo máximo de espera por uma ligação ao dispositivo. Este é o estado de um dispositivo que foi desligado da rede;
- Dispositivo errado – Indica que um dispositivo respondeu ao pedido, mas a resposta não foi a esperada;

- Ligação recusada – Acontece quando o uso de uma *firewall* não permite o acesso ao dispositivo.

Juntamente com o estado da câmara é indicada a data e hora das imagens visualizadas. Em funcionamento normal, a data e hora representarão o tempo actual mas, no caso de a câmara não estar *on-line*, a imagem apresentada é a última imagem capturada pela câmara. Neste caso a data e hora indicarão a data da referida imagem.

Pode-se optar por dois modos de visualização, sequência de imagens (4) ou por filme (5), seleccionando os botões acima da imagem do lado direito. Para a visualização do tipo filme é necessário instalar um componente *ActiveX*.

Através do botão "rec" (6) pode-se capturar imagens que se esteja a visualizar e mais tarde consultá-las na zona de histórico.

Algumas câmaras detêm a capacidade de serem apontadas segundo dois eixos (função *Pan & Tilt*), bem como permitem a definição da ampliação de imagem (*Zoom*). Para as câmaras que possuam estas características, as funções de controlo aparecerão disponíveis no ecrã de visualização (7).

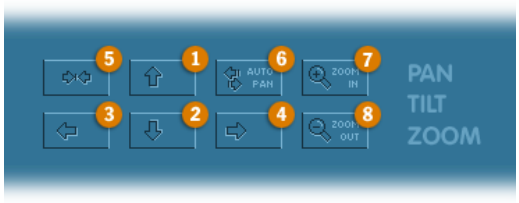

Figura 4.10 - Controlo do PTZ

O controlo *PTZ* (*Pan, Tilt & Zoom*) contém diversos botões de controlo, entre os quais setas verticais (1, 2) e horizontais (3, 4) que possibilitam o direccionamento segundo estes dois eixos. O botão centrar (5) impõe o regresso da câmara à posição predefinida. Os botões "*zoom in*" (7) e "*zoom out*" (8) permitem a regulação da ampliação da imagem. A função "*auto pan*" (6) solicita à câmara o varrimento automático, segundo o eixo horizontal, do seu espaço de visualização.

Ao seleccionar pela a 1ª vez o modo de visualização tipo filme, é solicitada a instalação do componente *ActiveX* respectivo (Figura 5), através do aparecimento de uma janela com a informação necessária.

Após clicar no link para efectuar o *download* do programa (1) o utilizador deverá descarregar o software de instalação, escolhendo a opção "Guardar" (2), e executá-lo

de seguida. O assistente de instalação irá acompanhá-lo nos diversos passos necessários.

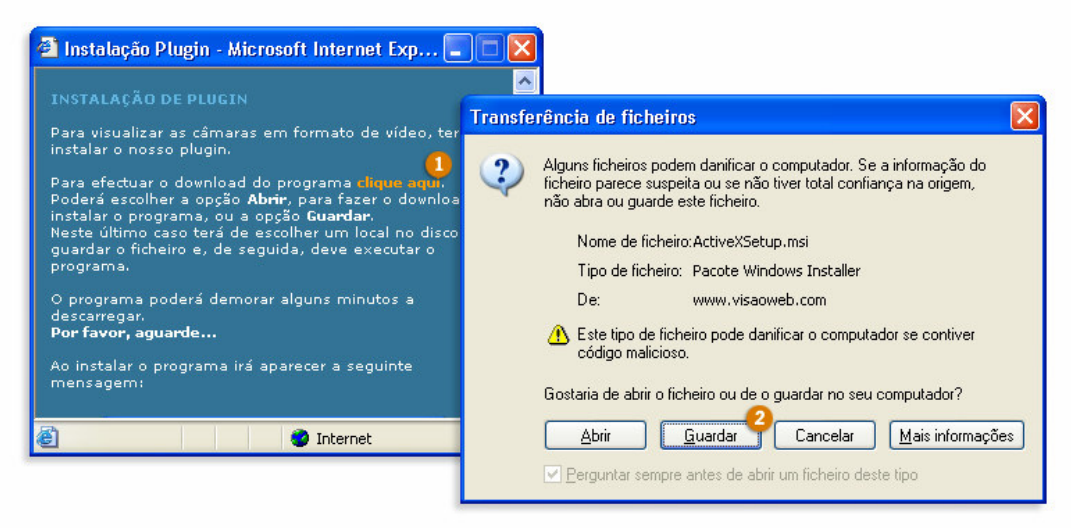

Figura 4.11 - Instalação do *ActiveX*

Figura 4.12 - Controlo de domótica

A interacção com os dispositivos eléctricos, controlados por elementos X10, previamente configurados, poderá ser efectuada através da zona "módulos X10".

À frente do nome de cada dispositivo estão os botões "ON" e "OFF" (1) através dos quais pode-se ligar ou desligar o dispositivo respectivo.

Existe ainda a possibilidade de armar ou desarmar a Consola de Alarme de forma rápida, através do botão (2) disponível na zona "módulos X10" para esse efeito. Este botão também permite saber o estado da consola através da sua coloração. Quando este apresenta um fundo azul, significa que a consola não está armada, um fundo laranja indica o estado de armada, sendo a cor vermelha usada no caso da consola de alarme estar no estado temporário que antecede o estado de armada.

Histórico de imagens

Para facilitar a consulta das imagens registadas na área de histórico, são disponibilizados vários filtros que optimizam a pesquisa:

- Pesquisa por intervalo de tempo em dias, definindo as datas de pesquisa (1);
- Pesquisa de imagens de uma única câmara (2);

- Pesquisa de imagens por tipo de aquisição/evento (3): modo automático (imagens registadas pela acção de um evento), modo manual (imagens registadas manualmente através da função "rec"), modo de alarme (imagens registadas pela disparo de um alarme).

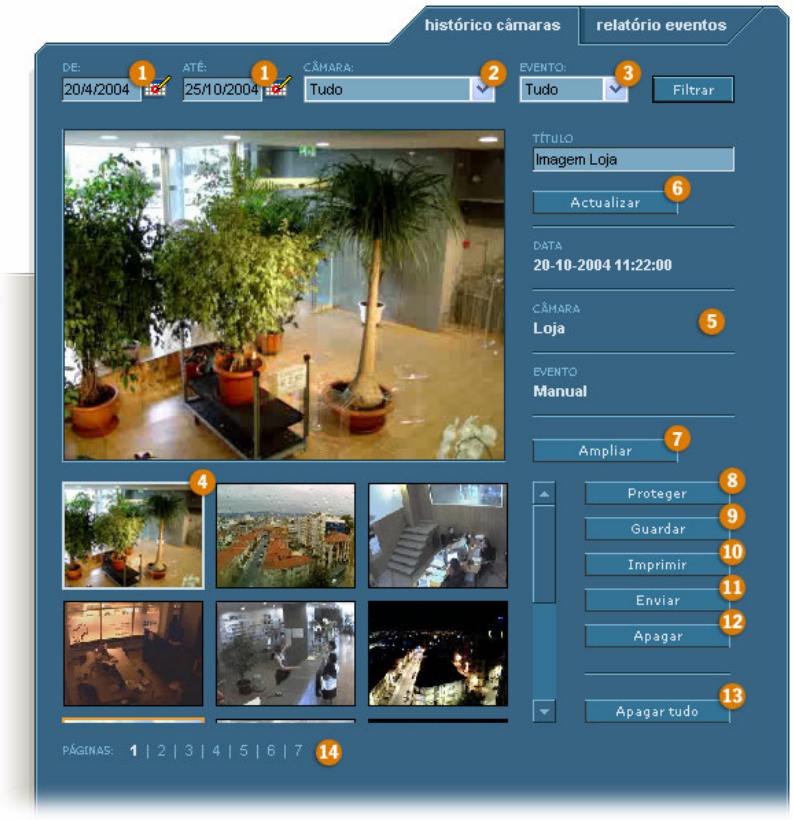

Figura 4.13 - Histórico de imagens/filmes

Para facilitar a inserção das datas de pesquisa, o utilizador pode pressionar o ícone, ao lado de cada caixa de texto, para abrir um pequeno calendário.

As imagens resultantes de uma pesquisa são apresentadas cronologicamente na área de *thumbnails*. Após seleccionar uma imagem (4) pode-se consultar (5) a data e hora de registo, bem como saber qual a câmara utilizada e qual o tipo de aquisição. É permitida a edição do título (6), assim como a visualização da imagem na maior resolução disponível, através do uso do botão "ampliar" (7).

A protecção de uma imagem impede que esta seja inadvertidamente eliminada manualmente, ao pressionar o botão para eliminar tudo, ou automaticamente (quando um evento de gravação ocorre e o utilizador tem o seu espaço de imagens completo).

As imagens podem ser protegidas ou desprotegidas através da função proteger/desproteger (8). O estado de protecção é indicado através da cor da moldura do *thumbnail* da imagem seleccionada: se a moldura for branca, a imagem encontra-se desprotegida, se for encarnada, esta encontra-se protegida. Em imagens não seleccionadas as molduras negra ou amarela indicam o estado desprotegido e protegido respectivamente.

Ao lado direito das imagens são disponibilizadas as funções para guardar no disco rígido (9), imprimir (10) ou enviar por e-mail (11). A função "apagar" (12) permite eliminar individualmente imagens desprotegidas. A função "apagar tudo" (13) elimina, de uma só vez, todas as imagens que não estejam protegidas.

Caso se use a opção de "apagar tudo" depois de se ter aplicado um filtro de pesquisa, apenas serão eliminadas as imagens apresentadas.

À medida que o número de imagens guardadas vai crescendo, estas vão sendo paginadas. Para alternar entre páginas de imagens deve-se pressionar o número da página pretendida (14).

Histórico de vídeos

Caso o utilizador disponha de um dispositivo com capacidade para gravar e armazenar vídeos, ser-lhe-á permito o acesso ao histórico de vídeos. De uma forma idêntica ao histórico de imagens, os vídeos armazenados poderão ser pesquisados por data (1) ou pela câmara que o capturou (2).

Dado que toda a informação dos vídeos capturados está alojada nos dispositivos locais, a listagem inicial ou uma pesquisa sobre os vídeos poderá demorar algum tempo a carregar.

A listagem dos vídeos é apresentada ordenada cronologicamente, sendo indicada a data/hora do inicio da gravação, o seu tamanho, duração e qual a câmara que o capturou (3). Para alternar entre páginas de vídeos, deve-se pressionar o número da página pretendida (4).

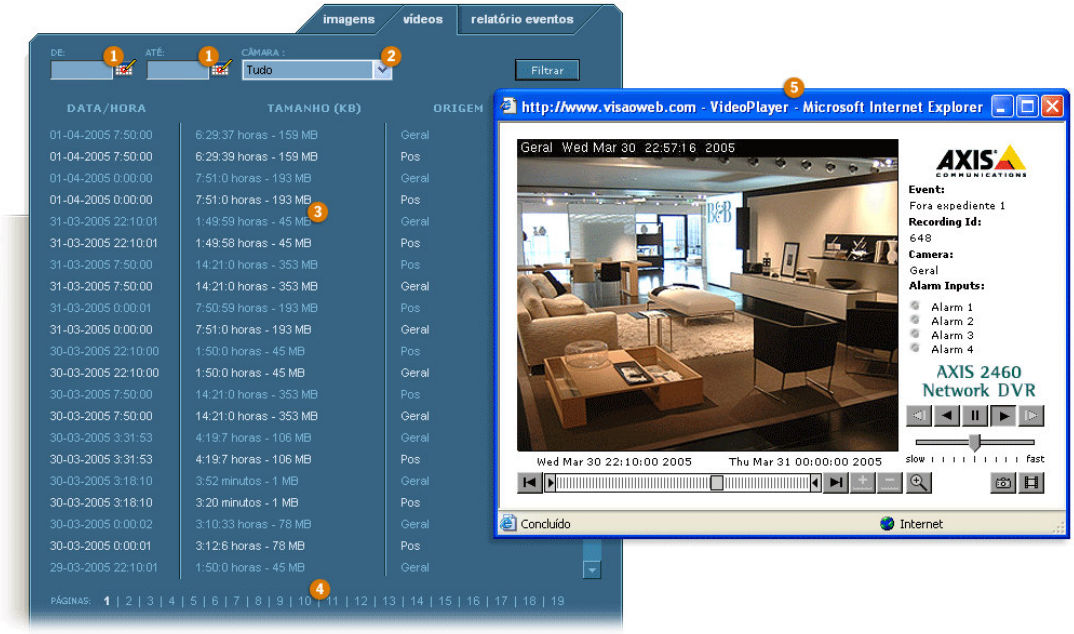

Figura 4.14 - Histórico de vídeos

Para visualizar um dos vídeos pretendidos deverá pressionar sobre o mesmo. Esta operação abre uma nova janela onde é apresentado o vídeo (5), juntamente com alguns controlos que permitem parar, reproduzir e controlar a velocidade e direcção da reprodução.

Relatório de eventos

Todas as acções efectuadas na aplicação, quer sejam manuais ou automáticas, são registadas no relatório de eventos da área de histórico, estando disponíveis para consulta os 1.000 últimos eventos registados.

Para facilitar a consulta dos eventos registados, podem-se utilizar uma série de filtros que optimizam a pesquisa:

- Pesquisa por intervalo de tempo em dias, definindo as datas de pesquisa (1);
- Pesquisa por tipo de evento (2);
- Pesquisa por origem do evento (3): câmara, consola, utilizador, dispositivo X10, etc.

Em alguns dos registos, por exemplo no login de um utilizador, poder-se-á aceder a mais informação passando o rato sobre o campo (4).

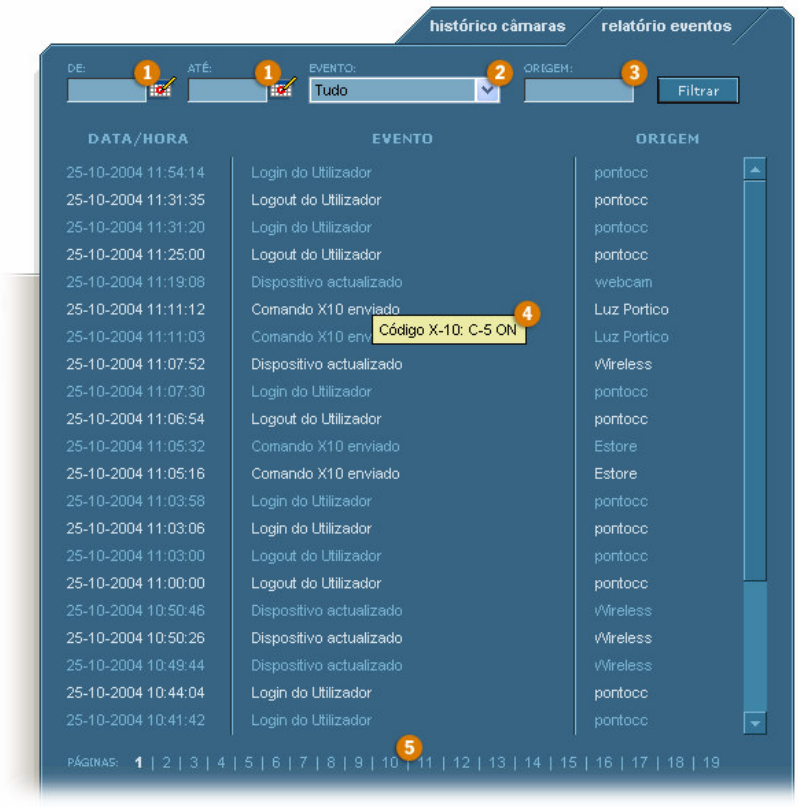

Figura 4.15 - Relatório de eventos

Quando o número de registos aumenta, estes são divididos em diversas páginas. Para alternar entre páginas de registos deve-se clicar no número da página pretendida (5).

Área de configurações

Esta é a área onde o utilizador poderá gerir os seus módulos X10, as suas câmaras, criar eventos e alterar os seus dados pessoais. De forma a organizar a informação de uma forma idêntica à página principal, as diferentes opções estão acessíveis através de etiquetas localizadas na parte superior.

Configurações de módulos de domótica

Para adicionar um módulo X10 à consola de controlo é necessário aceder à área das "configurações" e seleccionar a etiqueta "domótica" (Figura 9). Nesta área o utilizador deve introduzir o endereço X10 (3) do aparelho a ser controlado, atribuir um nome ao módulo (2), definir qual o dispositivo que irá enviar o sinal (1), escolher quais os locais (4) em que o comando estará disponível e, por fim, pressionar sobre o botão "Adicionar" (5).

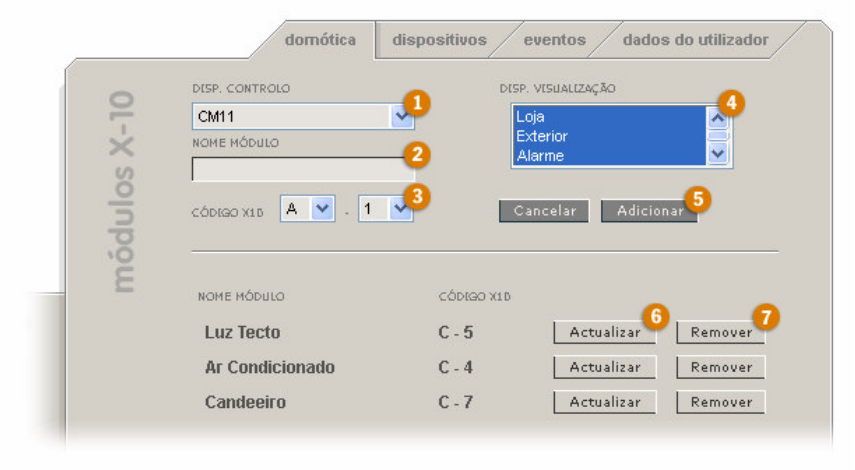

Figura 4.16 - Configurações de módulos de domótica

Na mesma área poderá em qualquer altura editar ou remover módulos. Ao pressionar o botão Actualizar (6), os campos superiores serão preenchidos com a informação associada ao módulo em edição, estando o fundo deste numa cor mais escura, e o botão (5) altera-se para "Actualizar". A opção para eliminar aparelhos de domótica obriga a uma confirmação por parte do utilizador.

Configurações de dispositivos

Figura 4.17 - Configuração de dispositivos

O VisãoWeb utiliza dispositivos que deverão ser instalados nos locais a vigiar/controlar e que permitem o envio e a aquisição de informação. Esses dispositivos podem ser câmaras, consolas de segurança ou vídeo-servidores, entre outros.

Para inserir um novo dispositivo deve-se escolher o tipo de dispositivo (1) entre aqueles que estão disponíveis. Pelo facto de se tratar de software executado em servidores, sempre que existirem novos dispositivos compatíveis com a aplicação, a lista será automaticamente actualizada. Após a escolha do dispositivo pretendido surgirá informação adicional que deverá ser preenchida. Os campos "nome" (2) e "descrição" (3) são elementos de identificação pessoais, pelo que podem conter qualquer tipo de informação. No caso do dispositivo o exigir, os campos login (4) e password (5) deverão ser preenchidos. Para facilitar a inserção dos dados, o campo "Endereço IP" (6) é preenchido por omissão com o IP da ligação do utilizador, pelo que este não deverá ter necessidade de alterar esta informação, e a "Porta TCP" (7), que identifica a porta aberta ao exterior configurada no *router* local, é configurada com o valor por omissão do equipamento. Os campos seguintes, "Endereço IP LAN" (8) e "Porta TCP LAN" (9), deverão ser preenchidos com o IP interno do dispositivo e a porta em que está disponível dentro da rede local. O preenchimento do campo "Endereço físico" (10) deverá ser efectuado mediante o número de série que consta no próprio dispositivo. O campo "Compressão" (11) define a qualidade da imagem a visualizar, em que 0 representa qualidade máxima (menor compressão) e 100 a máxima compressão (menor qualidade). O campo "Prioridade" (12) define o ordenamento dos dispositivos em termos de visualização, que por sua vez se repercute na sequência das respectivas etiquetas. Por fim, clicando no botão "Adicionar" (13) o novo módulo é adicionado à aplicação.

Em qualquer altura o utilizador poderá editar (14) ou eliminar (15) um dispositivo. Para facilitar a aprendizagem da aplicação, estes botões têm um funcionamento idêntico à página de configurações de módulos de domótica.

Configuração de eventos

O serviço permite a criação de eventos, sendo estes caracterizados por uma sequência de acções, despoletadas de acordo com uma série de parâmetros definidos.

Para criar um evento deve-se definir um nome (1) que o identifica e, caso seja necessário, um intervalo de tempo (2) em que o evento está activo. De seguida deverá escolher-se uma ou várias acções a serem executadas (passo 1), tendo em atenção a ordem pela qual são inseridas.

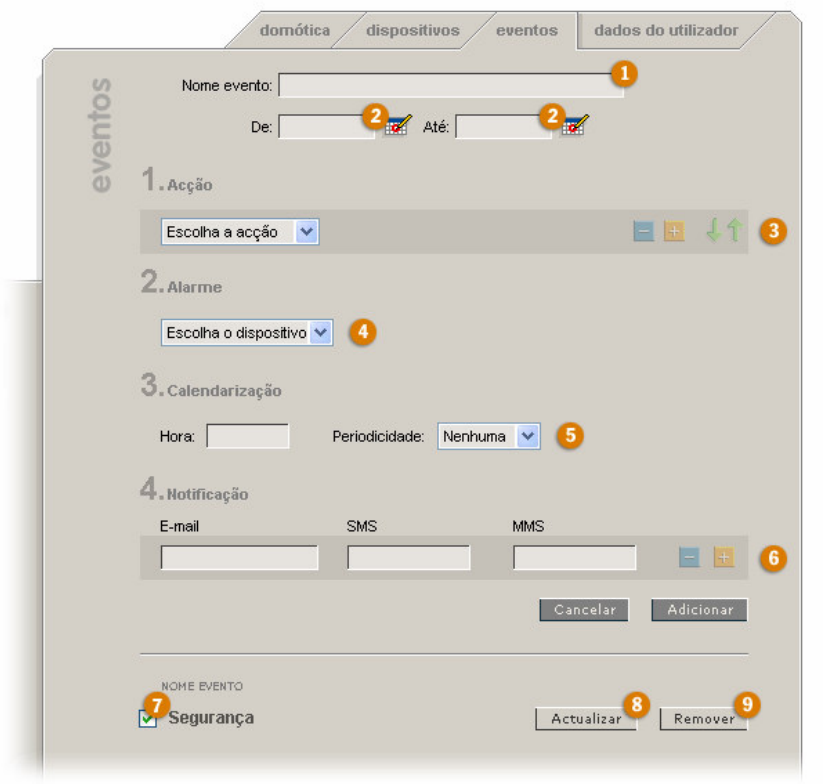

Figura 4.18 - Configuração de eventos

Uma sequência de acções pode ser despoletada após a ocorrência de um alarme (passo 2) ou a partir de uma calendarização pré-definida (passo 3), podendo ser escolhidas ambas as opções. Após essa ocorrência o utilizador poderá, se desejar, ser notificado (passo 4) através dos vários meios disponíveis.

Um evento pode ser activado/desactivado (7), editado (8) ou eliminado (9) em qualquer altura.

Passo 1 – Acção

Existem 4 tipos de acções disponíveis:

- Capturar imagem – solicita ao servidor a captura de uma imagem da câmara escolhida;
- Controlar aparelho – solicita o envio de um comando X10 para o aparelho definido;
- Controlar alarme – permite ligar, desligar ou activar a função de pânico de uma consola de alarme;
- Pausa – impõe um tempo de espera entre acções. É particularmente útil em casos onde, por exemplo, se pretende ligar uma luz antes de capturar uma imagem.

De salientar que algumas destas opções apenas estão acessíveis caso o utilizador tenha configurado os dispositivos necessários para a sua execução, i.e., o controlo de alarme só é possível se existir uma consola de alarme configurada.

Figura 4.19 - Configuração de acções

Os botões (6) e (7) permitem, respectivamente, remover e adicionar uma acção. Para alterar a ordem de execução das acções poderão ser usados os botões (8) e (9).

Passo 2 – Alarme

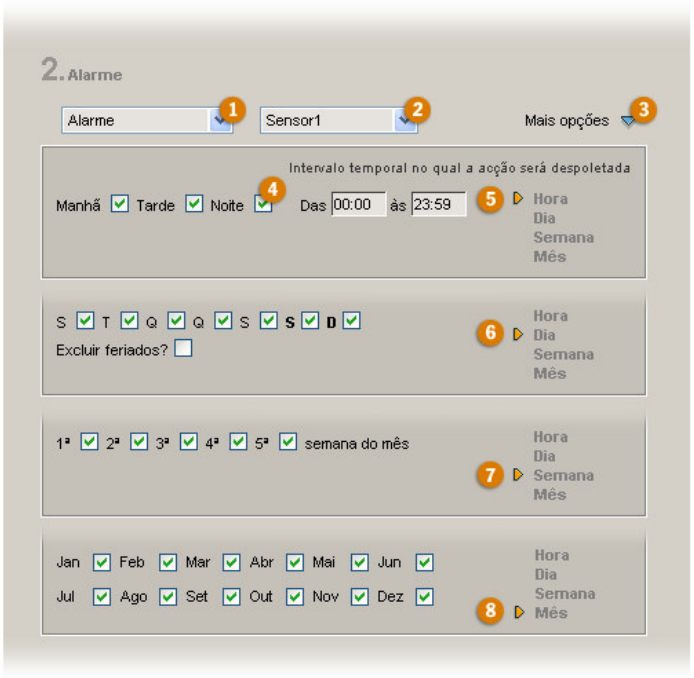

Figura 4.20 - Configuração do alarme

Caso se pretenda definir a execução das acções após a ocorrência de um alarme, deverá escolher-se o dispositivo (1) que accione o evento. Este pode ser uma consola de alarme ou uma câmara com a capacidade de detecção de movimento. No caso de um alarme pode-se escolher qual o sensor (2) a analisar.

Através do botão "mais opções" (3) pode-se definir os intervalos de tempo durante os quais o evento será executado após a recepção do alarme. Por exemplo, caso se pretenda capturar uma imagem e enviá-la para o telemóvel apenas no horário nocturno dos dias da semana e durante os fins-de-semana deve-se, na área de "mais opções", escolher a secção "Hora" (5) e seleccionar apenas a opção de "Noite" (4). Caso se pretenda escolher um horário diferente apenas é necessário alterar os campos de "das" e "às". Na área "Dia" (6), deve-se escolher apenas as opções de Sábado e Domingo. Na área "Semana" (7) podem-se definir as semanas do mês nas quais as acções deverão ser efectuadas e na área "Mês" (8) pode-se especificar quais os meses nos quais se pretende que as acções ocorram.

Passo 3 – Calendarização

Para configurar as acções a serem executadas num horário predefinido, deve-se preencher o campo "Hora" (1) e, caso se pretenda, a opção "Periodicidade" (2).

Caso escolha uma periodicidade poderá, através da opção "Mais opções" (3), restringir a execução das acções em termos de horas (4), dias (5), semanas (6) ou meses (7).

Figura 4.21 - Calendarização do Evento

Passo 4 – Notificação

O campo "Notificação" permite a notificação, após a execução das acções, através do envio de um e-mail (1), SMS (2) ou MMS (3). Nesta área pode-se, por exemplo, configurar um evento para se receber um MMS, com as imagens capturadas pelas câmaras, após a detecção de movimento no período nocturno.

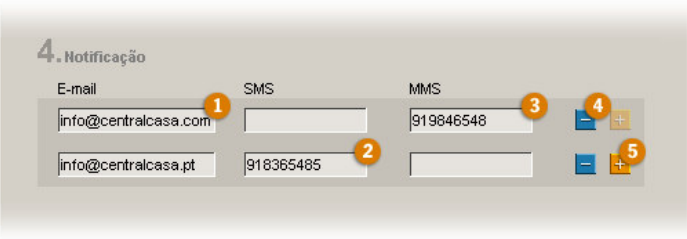

Figura 4.22 - Configuração da notificação

Para cada evento podem ser acrescentados (5) vários contactos para notificação, sendo a remoção destes efectuada através do botão (4).

Dados do utilizador

Na etiqueta "dados do utilizador" do menu "configurações" pode-se efectuar alterações nos dados pessoais, mudar o idioma da interface (2) ou alterar a palavra-chave (4) de acesso.

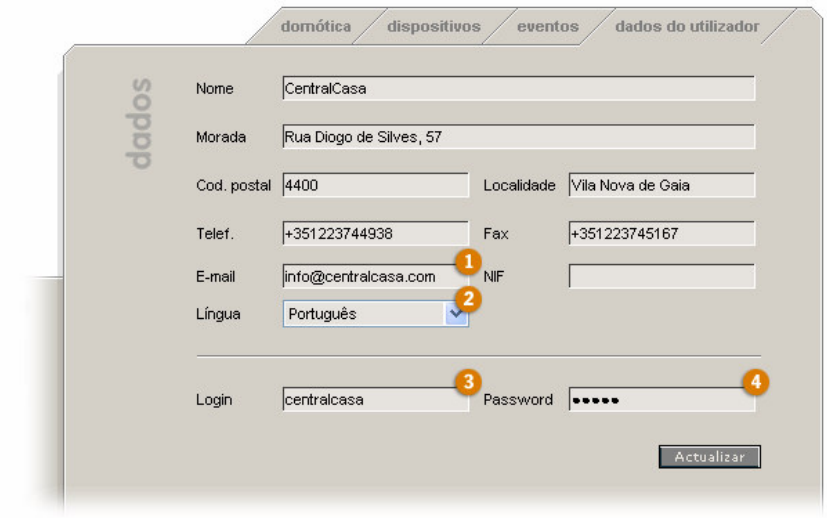

Figura 4.23 - Configuração dos dados de utilizador

O campo "login" (3) contém o login atribuído ao utilizador e não é editável.

Pode-se também alterar o e-mail (1) para o qual é enviada a palavra-chave no caso de se necessitar de recuperar a mesma. O endereço electrónico aqui indicado é também o que aparece por omissão quando se configura uma notificação de um evento e o que aparece no campo remetente quando se pretende enviar uma imagem do histórico por e-mail.

Dicas

Ao longo de todas as páginas, com excepção da página principal, devido à falta de espaço útil no ecrã, estão presentes pequenas dicas sobre como usar os comandos básicos. Desta forma o utilizador mais inexperiente terá um acesso fácil e directo a um pequeno manual que ajuda a perceber os comandos existentes e qual a sua função.

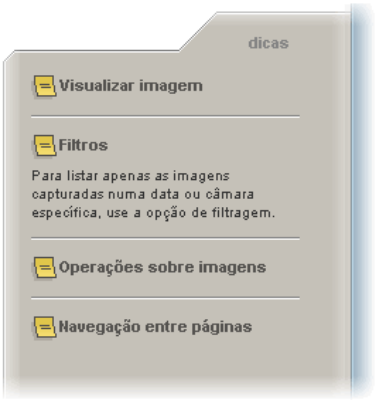

Figura 4.24 - Dicas de ajuda

4.3.2. Interface móvel

O desenvolvimento de uma página para um dispositivo móvel não poderá ser visto como uma mera adaptação de uma página HTML, isto é, todo o conteúdo e navegação devem ser pensados e optimizados para estes dispositivos, com características únicas, tais como a baixa capacidade de processamento, o ecrã reduzido e a forma de navegação/inserção de informação [Weiss, 02].

Para aumentar a compatibilidade e reduzir o tamanho dos conteúdos, não deverão ser usadas imagens para botões nem nenhum outro elemento decorativo, usando-se apenas texto [Milroy, 02]. De forma a distinguir um *link* do restante texto, optou-se por usar parêntesis recto em todos os textos que são *links*. O uso de tabelas também é evitado já que a sua compatibilidade não é garantida nos dispositivos móveis.

A página inicial, para além de identificar o produto, disponibiliza dois campos editáveis para a inserção de *login e password*. Após uma correcta validação, o *login* e *password* são armazenados numa *cookie* do cliente, caso o equipamento suporte *cookies*. Desta forma, numa visita posterior, o utilizador não necessita de inserir novamente a informação sendo redireccionado para a página principal. Na Figura 4.25 está representada a página inicial, tal como é visualizada no dispositivo móvel da SonyEricsson, modelo P900.

Figura 4.25 - Página de entrada (SonyEricsson P900)

A página principal mostra no topo o nome da câmara actual, seguido da data da imagem. De forma a optimizar as dimensões do ecrã, as imagens capturadas pelas câmaras são redimensionadas de forma a ocupar na totalidade a largura do ecrã do dispositivo móvel.

Figura 4.26 - Controlo de aparelhos (Nokia 6610)

Para implementar a funcionalidade de controlo de aparelhos devem ser visíveis, logo na página principal, todos os aparelhos e os respectivos *links* para ligar e desligar, tornando desta forma o acesso mais imediato. Por outro lado, caso o utilizador tenha bastantes dispositivos, tornará a primeira página demasiado extensa, indo contra um dos princípios básicos de usabilidade nos telemóveis. Para minimizar as desvantagens

optou-se por um meio-termo onde, caso o número de aparelhos seja superior a quatro, mostram-se os primeiros 4 seguidos de um botão que, quando premido, redirecciona o utilizador para uma página exclusiva de controlo de domótica.

Figura 4.27 - Visualização de câmaras (Siemens CX65)

No caso da escolha das câmaras aplica-se o mesmo raciocínio usado para o controlo de aparelhos: caso o utilizador disponha de mais de 4 câmaras, apenas serão listadas as primeiras quatro seguindo-se um botão que, quando pressionado, leve a uma página onde sejam apresentadas todas as câmaras. De salientar que a câmara actual não aparece listada, poupando-se algum espaço precioso do ecrã.

Caso a câmara seja motorizada são disponibilizados os *links* para controlo da sua posição e nível de zoom.

No fim da página encontra-se o menu de navegação que permita ao utilizador navegar para à área de histórico ou sair da aplicação.

4.4. ARQUITECTURA

A aplicação será constituída por vários subsistemas organizados e pensados de forma modular. A separação clara de funcionalidades deverá permitir o desenvolvimento independente de cada módulo, assim como possibilitar a reutilização de código [MSFT.NET, 03].

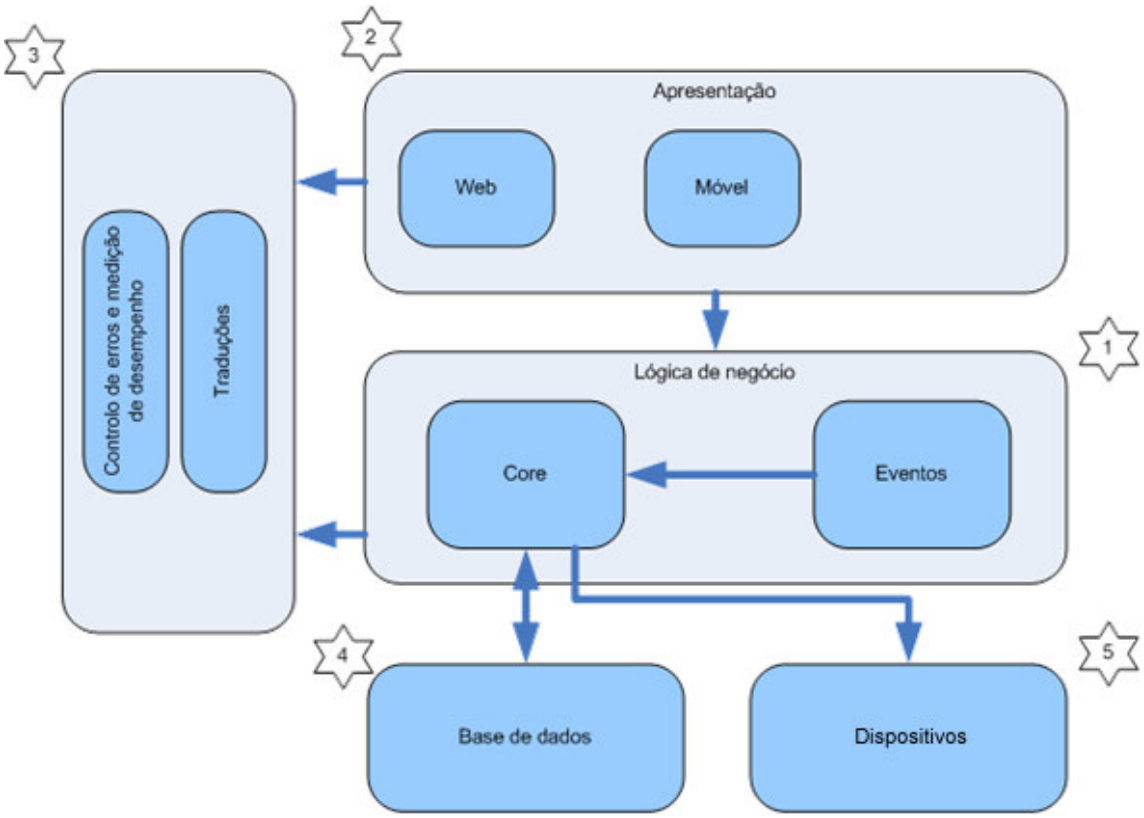

Figura 4.28 - Arquitectura conceptual da aplicação

Lógica de negócio

O módulo central será o da lógica de negócio (Figura 4.28 bloco 1). As suas responsabilidades são garantir o fornecimento de todas as funcionalidades necessárias ao módulo de apresentação. Nestes termos é neste módulo que serão satisfeitos grande parte dos requisitos da aplicação, incluindo a execução dos eventos da aplicação.

Apresentação

O módulo de apresentação (bloco 2) engloba todos os subsistemas de interacção com o utilizador. Estão pensadas duas interfaces distintas: aplicação web, para ser usada com um *browser* HTTP genérico e uma interface para dispositivos móveis. Todos estes

subsistemas interagem com o módulo de Lógica de Negócio.

Suporte

O módulo de suporte (bloco 3) fornece serviços de apoio à aplicação. Esses serviços são o serviço de tradução e de controlo de erros. O primeiro garante a internacionalização da aplicação, adaptando as interfaces com o utilizador a vários idiomas. O serviço de controlo de erros trata, numa primeira fase, de apoiar o próprio desenvolvimento e, numa fase posterior, após a entrada em produção, encarregar-se-á de medições de desempenho e notificação de erros ocorridos.

Base de dados

O módulo de base de dados, além de incluir a própria base de dados, contem um serviço de acesso aos dados. O componente deverá ser suficientemente modular para permitir o seu uso independentemente da base de dados usada.

Dispositivos

O módulo dos dispositivos (bloco 5) garante a comunicação entre o módulo da lógica de negócio e os dispositivos físicos. A concepção deste módulo em particular terá de ser orientada para permitir expansões regulares aos dispositivos suportados e também permitir que as interfaces de interacção sejam uniformes. A capacidade de adaptação às propriedades específicas de cada dispositivo é imprescindível.

4.5. A ESTRUTURA DE DADOS

Dada a dimensão dos diagramas *Entity-Relationship*, estes foram divididos em dois para uma representação mais clara. O primeiro diagrama (Figura 4.29), representa as tabelas que interligam o utilizador com os seus dispositivos e a sua área de histórico. No segundo (Figura 4.30), estão representadas as tabelas desenvolvidas para armazenar toda a informação associada aos eventos. Para uma melhor compreensão os diagramas contêm algumas tabelas comuns.

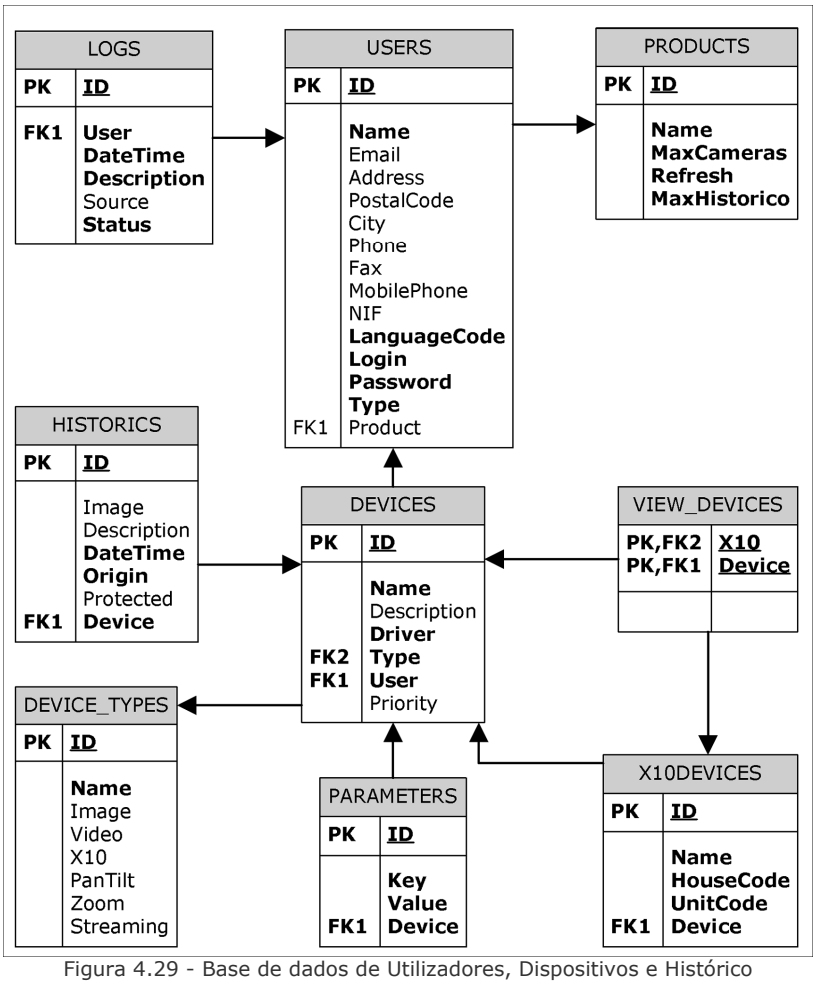

Figura 4.29 - Base de dados de Utilizadores, Dispositivos e Histórico

Users

A tabela `Users` armazena todos os dados pessoais dos utilizadores, incluindo o login, password, idioma, nome, morada, etc. Também está representada na tabela `User` uma chave externa (`product`) que associa o utilizador a um produto. O mesmo produto poderá ser associado a vários utilizadores, sendo que a um utilizador poderá apenas ser atribuído um produto único. Desta forma poderemos usar alguns produtos

base e, caso necessário, criar produtos específicos para cada cliente.

Products

Inicialmente estima-se que a aplicação seja comercializada através de três planos: o plano residencial, profissional e empresarial. No primeiro o utilizador apenas poderá usar o software local para a aquisição de imagens e controlo de aparelhos, não sendo possível o uso de outros dispositivos. No plano profissional e empresarial, não existem restrições no tipo de dispositivos, sendo a principal diferença entre ambos o número máximo de dispositivos permitidos, 1 e 4 respectivamente. Em cada um dos casos o utilizador poderá aumentar o número máximo de dispositivos, adquirindo licenças em separado.

Segundo a base de dados idealizada, será possível criar três produtos diferentes e, caso um utilizador queira associar mais uma câmara ao produto base, criar um outro produto, denominado, por exemplo, de Profissional+2, onde o campo `MaxCameras` passa a ter o valor 3.

Devices e Devices_Types

A tabela `Devices` representa todos os dispositivos associados a um utilizador. Esta tabela contém os campos `Name` e `Description` para identificação do dispositivo, o campo `Driver` para instanciar o objecto de comunicação correcto da lógica de negócio e o campo `Priority`, para permitir ao utilizador escolher qual a ordem de visualização dos dispositivos. A tabela também contém duas chaves estrangeiras: a chave `user`, que associa o dispositivo unicamente a um utilizador e uma chave `Type` que associa o dispositivo a uma outra tabela, `Device_Types`, que contem toda a informação sobre as características do dispositivo.

Parameters

Tendo em conta que um dispositivo pode ser uma câmara ou qualquer outro equipamento que suporte uma rede IP, como por exemplo, um alarme, torna-se impossível definir quais os campos necessários para cada dispositivo, sendo necessária a criação de uma nova tabela: `Parameters`. Esta tabela armazena de uma forma dinâmica todos os parâmetros associados ao dispositivo.

Historics

Para armazenar toda a informação associada às imagens capturadas quer manualmente, através da interacção do utilizador, ou automaticamente, através da execução de eventos, é usada a tabela `Historics`. É importante referir que, por

questões de facilidade de consulta, as imagens propriamente ditas são guardadas no disco, sendo o nome do ficheiro o `id` da imagem antecedido por "img_". Para associar uma imagem ao dispositivo que a capturou, é usada a chave externa `Device`.

Logs

Para registar todas as interacções do utilizador com a aplicação foi criada a tabela `Logs`. Esta regista a data e hora da ocorrência do evento, uma pequena descrição e o estado do mesmo (se foi bem ou mal sucedido). A chave externa `User` é usada para associar a cada `Log` um utilizador.

X10 Devices e View_Devices

Para a associação de um módulo X10 com o respectivo dispositivo que o controla deparámo-nos com um problema singular. Após alguma reflexão sobre o funcionamento do X10, facilmente nos apercebemos de que uma instalação pode ter um ou vários dispositivos com capacidade de controlo X10. Se tivermos, por exemplo, uma vivenda com duas câmaras instaladas, uma na entrada e outra no jardim, e uma consola de alarme responsável pelo controlo dos equipamentos X10 (luzes do jardim e da entrada), terá lógica que, enquanto estamos a visualizar a câmara do jardim possamos actuar sobre as luzes dessa área, sendo o mesmo aplicável à câmara da entrada.

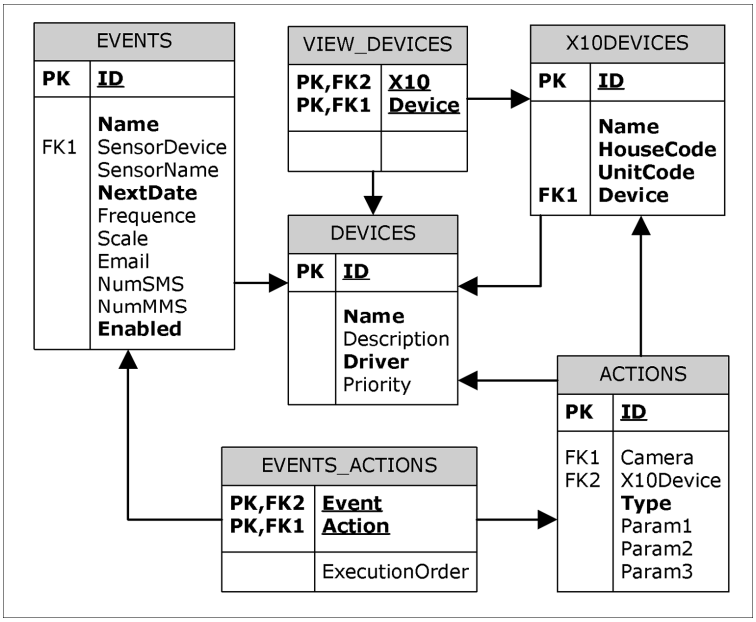

Figura 4.30 - Base de dados de Eventos

Seguindo este raciocínio torna-se claro que os módulos X10 não devem estar associados apenas ao dispositivo de controlo mas também a um ou mais dispositivos, aos quais chamaremos dispositivos de visualização. A criação da tabela `View_Devices` vem resolver este problema. Desta forma um `X10Device` está associado a um dispositivo de controlo, através da chave estrangeira `Device`, ao mesmo tempo que, através da tabela `View_Devices`, pode estar associado a um ou mais dispositivos de visualização.

Para a representação dos eventos foi necessário adicionar 3 novas tabelas: `Events`, `Events_Actions` e `Actions`.

Events

Esta tabela é responsável por armazenar o nome, a data de execução (`NextDate`), a respectiva repetição, através dos campos `Frequence` e `Scale`, indicando o primeiro a frequência de execução e o segundo a unidade da frequência (meses, semanas, dias, horas ou minutos), e a informação necessária para a notificação. De notar que os campos `Email`, `NumSMS` e `NumMMS` não são de preenchimento obrigatório. A sua existência indica se deverá existir ou não notificação. Para além destes campos existe uma chave estrangeira que associa esta tabela à tabela de dispositivos. Esta chave permite-nos definir se o evento deve ou não ser despoletado após a detecção de movimento do dispositivo associado.

Events_Actions

A tabela `Events_Actions` é a responsável por armazenar as acções a serem executadas. O campo `ExecutionOrder` garante que estas são executadas segundo uma certa ordem. Por exemplo, no caso do utilizador escolher uma acção do tipo capturar imagem e uma acção do tipo rodar câmara, será mais lógico que a última seja executada primeiro para que, quando a imagem for capturada, a câmara esteja na sua nova posição.

Actions

As acções propriamente ditas ficam armazenadas na tabela `Actions`. Através do campo `Type`, define-se qual o tipo de acção a ser executado. Dependendo do tipo de acção poderá ser necessário o preenchimento dos campos `Param` e das chaves estrangeiras `Camera` e `X10Device`.

4.6. ARQUITECTURA FÍSICA

Nesta secção será discutida a distribuição dos componentes da arquitectura pelas diversas máquinas que suportam a aplicação. Esta arquitectura foi definida tendo em conta os requisitos iniciais de segurança, fiabilidade e desempenho, mas contrapondo as limitações financeiras da empresa e tentando aproveitar ao máximo os recursos informáticos existentes.

A primeira linha de defesa contra ataques informáticos, incluindo vírus e *worms*, fica a cargo de uma *firewall*, embebida num *router*. Este dispositivo analisa o tráfego de informação proveniente da Internet e, de acordo as suas definições, rejeita ou redirecciona os pedidos para os respectivos servidores.

A aplicação é disponibilizada com recurso a três servidores distintos, cada um com a sua tarefa, sendo a ligação assegurada por uma linha dedicada de 1024/1024 Kbps. Conforme será referido posteriormente, um dos servidores poderá ser alimentado por uma outra linha separada.

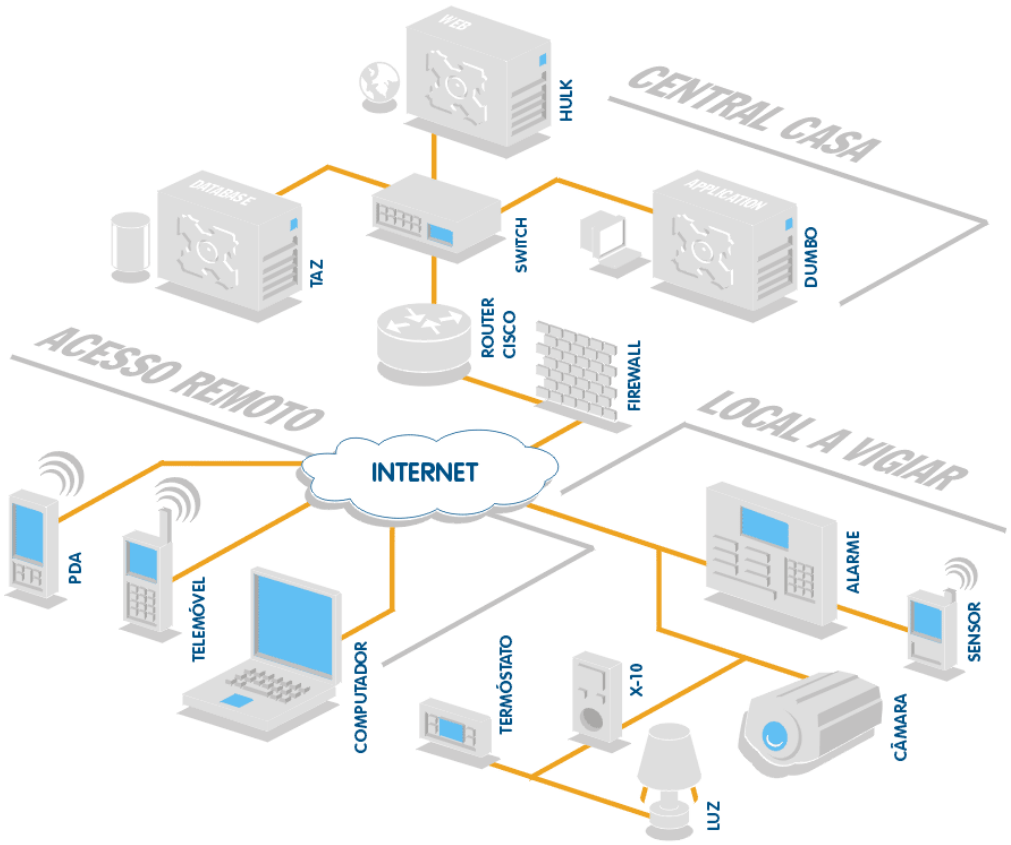

Figura 4.31 - Arquitectura física da aplicação

Os três servidores estão designados pelos seus nomes internos: Hulk, Taz e Dumbo.

HULK

O núcleo da aplicação encontra-se fisicamente no servidor HULK. Esta máquina é suportada pelo sistema operativo *Microsoft Server Web Edition 2003*, dedicada a servir páginas web através do componente IIS (*Internet Information Services*), sendo todos os pedidos feitos pelos clientes, recebidos e processados por este servidor. Este servidor contém todos componentes da lógica de negócio, assim como os componentes de acesso a dados, interacção com dispositivos e a interface com o utilizador.

Tendo em vista a escalabilidade da aplicação, este servidor está preparado para suportar a técnica de *web garden* e *web farm*. O primeiro distribui os pedidos dos clientes por processos separados, conseguindo-se desta forma maximizar a utilização dos recursos de máquinas com mais do que um processador. O segundo permite escalar o número de pedidos atendidos através do uso de novas máquinas. Esta técnica permite um balanceamento de carga automático, sendo o conjunto das máquinas visto pelos utilizadores como uma única máquina. Com esta implementação, se um dos servidores falhar, os restantes continuarão a fornecer o serviço sem existirem interrupções.

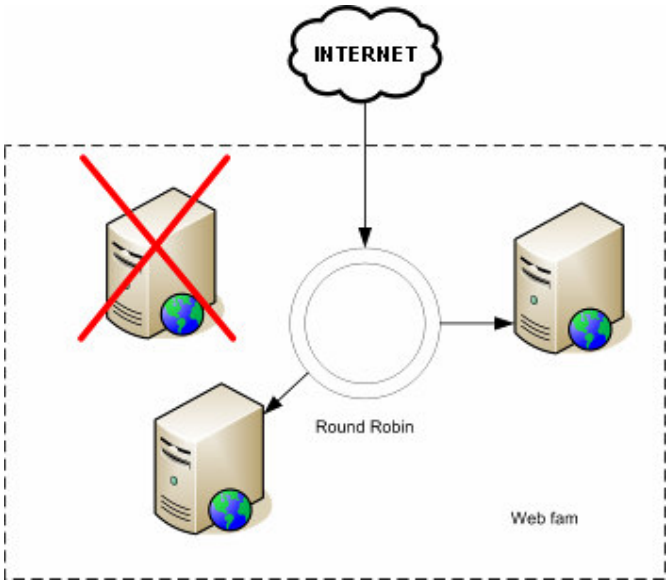

Figura 4.32 - Funcionamento de uma web farm

Foi escolhida a centralização do núcleo do serviço apenas nesta máquina, a que dispõe de maior capacidade de processamento (duplo processador Xeon a 2,4GHz) e memória RAM (2 GB), com o objectivo de facilitar a manutenção e evolução do

serviço. A manutenção está facilitada pois todos os ficheiros de configuração estão reunidos numa só máquina, sendo também fácil a evolução pois basta actualizar qualquer componente no servidor HULK, que todos os serviços que o usam passam automaticamente a usar a versão mais recente, não necessitando eles próprios de ser actualizados separadamente.

TAZ

Este é o servidor que mantém a base de dados de toda a aplicação. Tendo em conta que o único servidor que comunica com a base de dados é o HULK, foi dimensionada uma ligação entre ambos com uma largura de banda de 1 *gigabit/s*. O servidor de bases de dados usado é o *SQL Server 2000*, estando já nativamente preparado para ser escalado por várias máquinas, sendo neste caso o balanceamento de carga automático.

Tendo em conta as grandes exigências ao nível de armazenamento, quer em termos de espaço, fiabilidade e desempenho, este servidor está equipado com um sistema de RAID de nível 5 (redundância + performance) contendo actualmente 4 discos SCSI de 36 GB, com possibilidade de expansão até 10 num sistema de *Hot-Swap*.

DUMBO

Inicialmente esta máquina não estava prevista, pois pensava-se incluir todas as aplicações no servidor HULK mas, com o desenvolvimento das mais recentes funcionalidades, torna-se muito útil o uso de um servidor separado para efectuar tarefas repetitivas, como o processamento de eventos ou outras funcionalidades que obriguem ao uso de um serviço permanente.

No futuro espera-se desenvolver um componente de servidor que permita a recepção, em tempo-real, de filmes transmitidos pelas câmaras locais. Esta funcionalidade obriga à existência de um serviço que esteja permanentemente à escuta e, sempre que receba uma transmissão de vídeo, o comprima e grave no local certo. Uma outra vantagem do uso de um servidor separado é a possibilidade de ligar este através de uma linha de acesso à Internet mais económica, como por exemplo, uma linha ADSL.

5. IMPLEMENTAÇÃO

Após o levantamento dos requisitos e a definição da arquitectura, efectuados no capítulo anterior, foi iniciada a programação da aplicação. Neste capítulo serão analisadas em detalhe as diferentes camadas que compõem a aplicação, incluindo as suas principais classes. Será dado um grande destaque à comunicação com os dispositivos e às soluções empregues para resolver alguns dos problemas levantados.

A tecnologia base para o desenvolvimento deste trabalho foi a plataforma *.NET* da Microsoft, usada quer ao nível dos sistemas operativos, *Windows Server 2003 Web* e *Standard Edition*, quer ao nível de ferramentas de desenvolvimento, *Visual Studio .NET*, tendo sido usado o C# como linguagem predominante. Esta escolha, para além de motivos pessoais e profissionais, prende-se ao facto de ser uma das mais recentes e promissoras tecnologias que oferece muitas vantagens do desenvolvimento de soluções distribuídas.

Na programação, de forma a uniformizar todo o código desenvolvido, optou-se por usar a norma da Microsoft no que se refere à nomenclatura, sendo o nome de todas as funções, propriedades, tabelas, campos, etc. representado na língua Inglesa. Os comentários estão sempre presentes na língua Portuguesa.

5.1. DEFINIÇÃO DAS CAMADAS (N-TIER)

Como qualquer arquitectura de software moderna, a arquitectura desta aplicação baseia-se na filosofia de desenvolvimento de software por camadas, chamadas aplicações *n-tier*. Este tipo de arquitectura é mais flexível, uma vez que cada camada apenas necessita de se preocupar com as camadas adjacentes, ficando independentes das restantes.

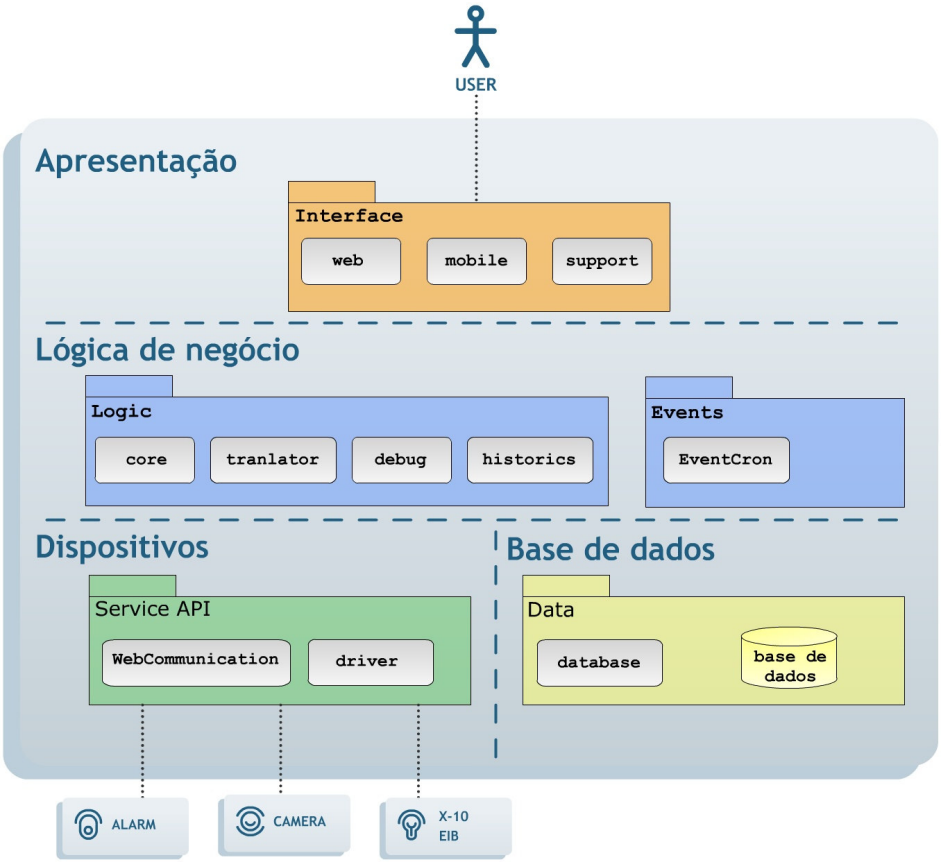

Figura 5.1 – Arquitectura por camadas

Tendo em conta a arquitectura da aplicação (consultar capítulo 4.4, Figura 4.28), serão definidas as seguintes camadas:

- Apresentação – Esta é a camada que controla a interacção com o utilizador e dispositivos. Esta camada é composta por três interfaces distintas: A interface para a `web`, uma interface `mobile`, para dispositivos com menos capacidades e uma interface `support`, para receber notificações por parte dos dispositivos. Esta camada comunica apenas com a camada inferior – lógica;

- Lógica de negócio – Esta camada é normalmente definida como *middle-tier* ou *business tier*, sendo composta pelo pacote Logic, onde estão centralizadas as funcionalidades e regras centrais da aplicação, e o pacote `Events` responsável pela execução dos eventos;

- Dispositivos – É a camada responsável pela interacção com os diferentes equipamentos. O componente `WebComunication` inclui as classes necessárias para a abertura das ligações TCP/IP com os dispositivos, sendo da responsabilidade do componente `Driver` o comando dos diferentes aparelhos;

- Base de dados – Como o próprio nome indica esta é a camada responsável pela interacção com a base de dados. Todas as interacções com a base de dados são efectuadas através do pacote `DataBase`, garantindo-se assim uma independência da aplicação ao tipo de base de dados usada.

5.2. CAMADA DA LÓGICA DE NEGÓCIO

Todos os objectos com relevância para a semântica do negócio, assim como todas as suas relações e operações, estão representados neste componente, o qual é constituído por cinco componentes: `Core`, `Translator`, `Debug`, `Historics` e `Events`.

Logic.Core

O núcleo da aplicação assenta neste componente. Devido à sua grande complexidade e necessidade de evolução está pensado de forma modular e independente. A sua estrutura segue a recomendação da Microsoft [MSFT.NET, 03] sobre a arquitectura de componentes, segundo a qual a lógica está dividida em componentes autónomos. Devido à capacidade de isolamento de funcionalidades proporcionadas por esta arquitectura, é garantida a evolução do sistema sem modificações significativas na sua estrutura.

Neste componente estão incluídas todas as enumerações, as classes responsáveis pelas constantes globais e as classes de suporte ao serviço de tradução.

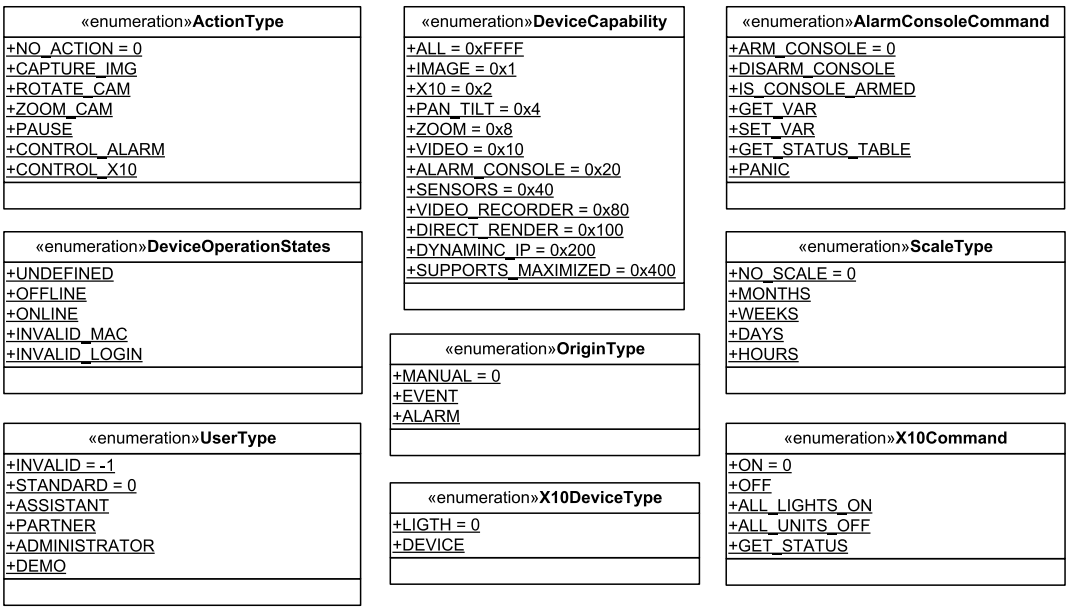

Figura 5.2 - Diagrama de enumerações do componente `Logic.Core`

Na figura anterior podem ser observadas as principais enumerações usadas na aplicação:

- Enumeração `ActionType` – Tipo de acção passível de ser executada pelo sistema de eventos;
- Enumeração `AlarmConsoleCommad` – Diferentes comandos suportados por um dispositivo do tipo consola de alarme;
- Enumeração `DeviceCapability` – Características reconhecidas nos dispositivos, i.e., cada dispositivo terá uma ou mais destas características;
- Enumeração `DeviceOperationStates` – Define os vários estados de um dispositivo;
- Enumeração `OriginType` – Representa as origens possíveis para uma entrada no histórico;
- Enumeração `ScaleType` – Intervalos de tempo disponíveis para o escalonamento da execução de eventos;
- Enumeração `UserType` – Categorias possíveis para utilizadores da aplicação;
- Enumeração `X10Command` – Comandos X10 que o sistema suporta;
- Enumeração `X10DeviceType` – Categorias em que se classificam os módulos X10.

Classe `StaticConsts`

Reúne todas as constantes globais, sendo inicializada através dum construtor estático que lê as constantes a partir do ficheiro de configurações. O ficheiro de configurações é determinado pela plataforma *.NET* conforme o contexto onde a classe seja carregada. Se o ambiente for uma página web o ficheiro de configurações é o *web.config*, se o ambiente for um executável o ficheiro de configurações é o *machine.config* (ou outro equivalente). As principais constantes presentes na classe indicam o caminho para a pasta onde estão as directorias pessoais dos utilizadores (`UserHomesDirUrl` e `UserHomesDirPath`), onde se encontra o ficheiro com as traduções (`text.xml`) e a *string* de ligação à base de dados (`DataBaseConnectionString`).

Logic.Translator

Dado um dos requisitos ser a disponibilidade da aplicação em várias línguas, existe a necessidade em toda a aplicação de apresentar conteúdos adaptados a diversos idiomas. O serviço de traduções consiste num mecanismo de transformação duma *string* de referência num texto no idioma pretendido. O serviço assenta na manutenção de vários dicionários indexados pela *string* de referência.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<texts>
   <text Reference="errSessionTimeout">
      <language Code="pt">A sua sessão expirou.</language>
      <language Code="es">La sesión terminó.</language>
      <language Code="en">Session timeout.</language>
   </text>
   <text Reference="lblMemorizePass">
      <language Code="pt">Memorizar password</language>
      <language Code="es">Recuerde la contraseña</language>
      <language Code="en">Remember password</language>
   </text>
   ...
```

Extracto do ficheiro de traduções `text.xml`

Classe `TranslateService`

É a classe motor do serviço de traduções. Estão implementadas nesta classe funcionalidades para carregar os textos de tradução e para fazer a tradução em si. A classe autoinicializa-se através dum construtor estático onde são carregados os textos para tradução a partir dum ficheiro XML (`text.xml`).

- Método `LoadTexts()` – Encarrega-se de fazer o *parse* do ficheiro XML com os textos traduzidos. Durante este processo são identificadas as *strings* de referência que são adicionadas, juntamente com as respectivas traduções, ao dicionário de cada língua. Se o método falhar é criada uma entrada no ficheiro de logs indicando a razão da falha. Caso o processo seja concluído com sucesso é activada a *flag* `textsLoaded` que indica disponibilidade dos conteúdos traduzidos;

- Método `AddReferenceText()` – Método privado, auxiliar do método `LoadTexts()`, que tem por função aumentar o dicionário de traduções. Por razões de desempenho na pesquisa, o dicionário é uma tabela de *hash*, indexada pela *string* de referência;

- Método `Translate()` – Recebe como parâmetros a *string* de referência, o código do idioma pretendido para a tradução e, opcionalmente, parâmetros de formatação do texto de saída. Retorna o texto traduzido ou, em caso de problemas na tradução, devolve a *string* de referência. O método antes de tentar obter qualquer tradução certifica-se de que já se procedeu ao carregamento dos dados de tradução;

- Propriedade `SourceFile` – Em qualquer altura é possível alterar o ficheiro XML fonte dos dados. Cada alteração desta propriedade provoca nova leitura dos textos de tradução através da chamada ao método `LoadTexts()`.

Com o objectivo de clarificar a sequência de actividades implicadas no processo de tradução de uma referência mostra-se, na Figura 5.3, o diagrama de actividades, em UML, do processo de tradução. Note-se que este processo está desenhado para resistir a falhas diversas e, para optimizar o desempenho, os textos apenas são carregados em memória na primeira vez que é feito um pedido de tradução, evitando-se a necessidade de carregar idiomas que não são lidos. Caso aconteça qualquer tipo de erro que impeça o correcto carregamento dos textos ou seja pedida uma tradução impossível de satisfazer com os dicionários actuais, é retornada a própria *string* de referência.

Figura 5.3 - Diagrama de actividades do processo de tradução

Logic.Debug

Seguindo o princípio básico de que errar é humano, deveremos contar com falhas na implementação ou, mais grave, falhas na concepção. Dado ser impossível a detecção de todas estes *bugs* antes e depois de colocar o serviço em produção,

optou-se pelo desenvolvimento de uma classe que se responsabilizasse pela captura e análise das excepções ocorridas e não tratadas pelas camadas inferiores, encaminhando os erros para os assistentes técnicos, responsáveis pela gestão técnica do serviço.

O serviço de `debug` assenta nas funcionalidades fornecidas pela classe `EventLogs`, conjugando-as com uma metodologia de tratamento de erros específica. Esta metodologia está esquematizada na Figura 5.4. A sua filosofia de funcionamento é que todos os erros devem ser capturados, tratados e, se não for possível recuperar deles, devem ser de novo lançados para serem capturados em camadas superiores.

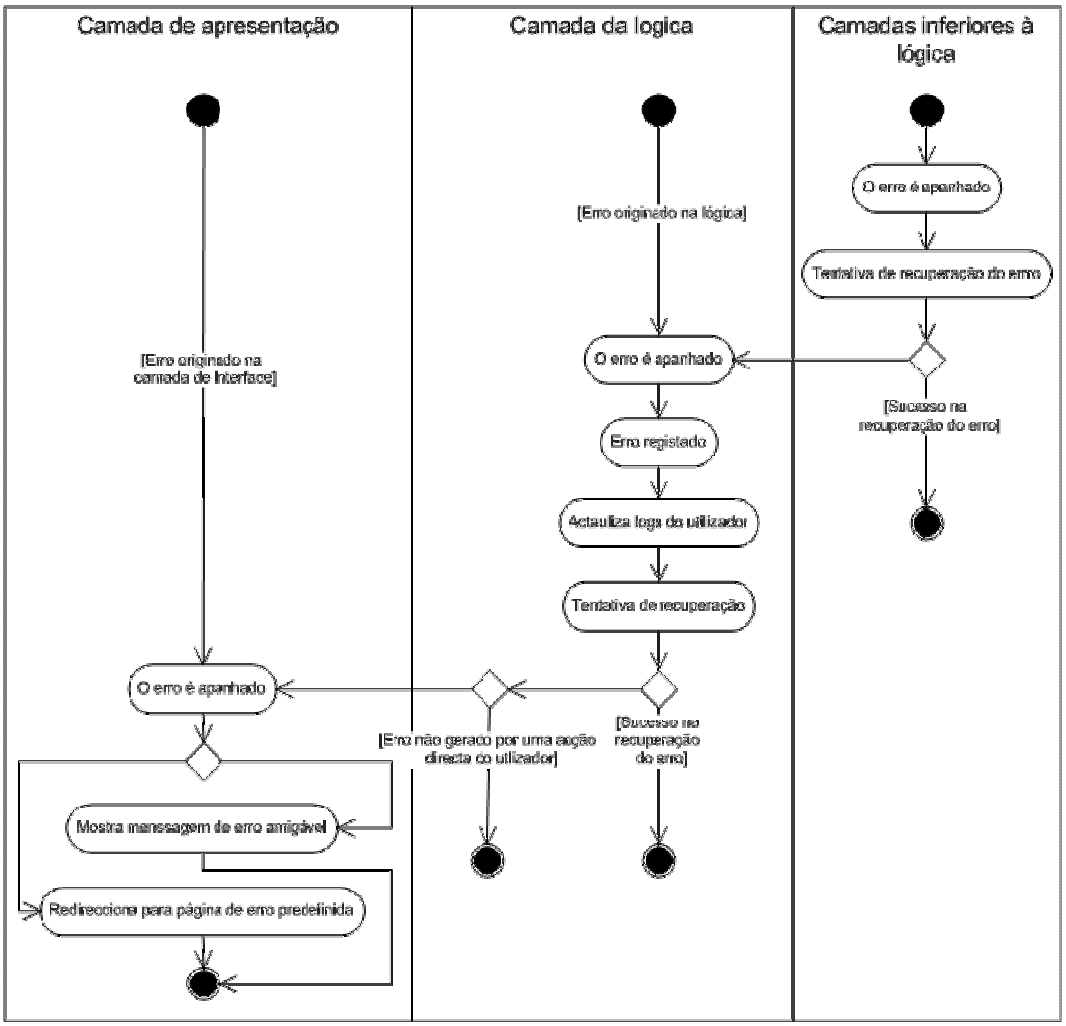

Figura 5.4 - Processo de análise de erros

Quando acontece um erro num serviço abaixo da lógica, seja o erro no serviço de acesso a dados ou nos `drivers`, a aplicação tenta recuperar da situação anómala. Caso não consiga, limita-se apenas a propagar o erro para a classe da lógica que

fez a chamada que originou a excepção.

Na camada da lógica os erros, quer sejam eles originários de camadas inferiores ou da própria lógica, são sempre registados num ficheiro, em formato XML, reservado para esse efeito. Os erros originados ou propagados até à camada de apresentação provocam dois comportamentos distintos: ou a situação de erro está prevista e é dada informação ao utilizador através de uma mensagem de erro amigável ou, em última instância, se o erro não tiver sido apanhado em parte alguma da camada de apresentação, antes deste provocar a finalização precoce da aplicação ou o aparecimento duma mensagem de erro não tratada, o utilizador é redireccionado para uma página de falha genérica (`genericError.aspx`). Esta última situação não faz parte do comportamento normal da aplicação e por isso cada vez que a página de erro é chamada, o administrador do sistema deve ser notificado.

Nos parágrafos seguintes descreve-se o mecanismo de registo de erros em disco. O mecanismo é baseado em funcionalidades da classe `EventLogs`, estando o seu diagrama de classes representado na Figura 5.5.

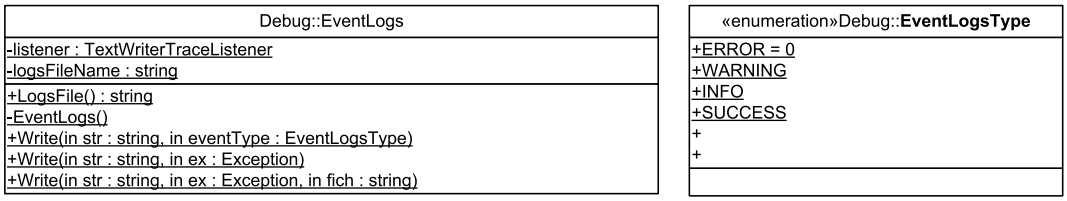

Figura 5.5 - Diagrama de classes do componente Logic.Debug

Classe `EventLogs`

É a classe responsável por manter um rasto de todos os erros ou informações importantes geradas em *runtime*. Essa informação é armazenada em formato XML de forma a possibilitar o seu tratamento e filtragem por ferramentas automatizadas.

- Método `Write()` – Nas suas várias implementações este método é responsável por adicionar entradas no referido ficheiro XML. De entre os vários parâmetros possíveis de serem passados ao método, é sempre necessário passar um texto descritivo e, no caso de se tratar da notificação dum erro, também deve ser enviada uma referência ao erro. Para fácil identificação no código fonte do local exacto onde ocorreu a falha, passa-se uma referência ao ficheiro fonte;

- Propriedade `LogsFile` – Inicialmente contém o caminho para um ficheiro de registo de erros indicado pelo ficheiro de configurações. A qualquer momento é possível alterar a localização do ficheiro de `Logs`.

Para além das funcionalidades descritas, a classe `EventLogs` tem uma funcionalidade extra de associar um *listener* na variável da plataforma *.NET Trace*. Serve esta funcionalidade para, durante o processo de desenvolvimento e teste da aplicação, guardar, juntamente com os erros ocorridos, informações de *debug* diversas que são produzidas e enviadas para o ficheiro virtual `Trace`.

Logic.Historics

Este componente é o responsável pela gestão de todas as imagens e vídeos dos utilizadores. Este componente é composto pela classe `Historic`, a qual representa uma imagem ou vídeo.

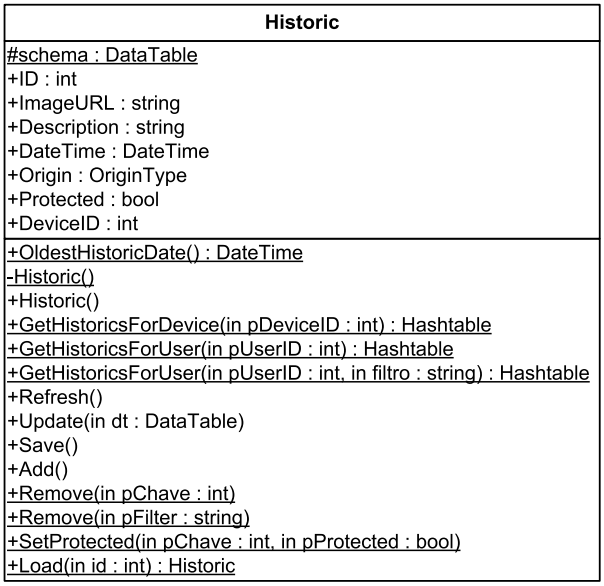

Figura 5.6 - Diagrama de classes do componente Logic.Historic

- Propriedade Origin – Indica qual a câmara que capturou a imagem/vídeo;

- Método `OldestHistoricDate()` – Retorna a data da última imagem capturada;

- Métodos `GetHistoricsForDevice()` e `GetHistoricsForUser()` – Ambos os métodos são estáticos e retornam uma *Hashtable* de vários objectos do tipo `Historic`. O primeiro retorna as imagens/vídeos associados a um

dispositivo, ao passo que, no segundo caso, são retornadas todas as imagens capturadas pelos vários dispositivos associados a um utilizador.

Pacote Events

Uma das principais funcionalidades da aplicação é a execução de macros (sequências de acções) após a ocorrência de certas condições. Estas condições podem estar relacionadas com um evento temporal ou com alterações no estado de sensores ou dispositivos ligados à aplicação e são conhecidas como `Triggers`.

O componente `EventCron` existe exactamente para suportar os `DateTriggers` (`Triggers` que disparam eventos de acordo variáveis temporais). Assim o `EventCron` procura periodicamente (em intervalos de um minuto) eventos que estejam prontos a ser executados e executa-os (método `fire`). Após a sua execução, a data de execução (`NextDate`) é actualizada de acordo os campos `frequence` e `scale`.

5.3. CAMADA DOS DISPOSITIVOS

Esta camada funciona como ponte de comunicação entre classes da lógica e controladores de dispositivos ou os próprios dispositivos. Do ponto de vista de uma arquitectura em camadas, esta camada fornece serviços que são usados pela lógica.

Esta camada é composta pelos componentes `WebCommunication` e `Driver`.

WebCommunication

Na sua grande maioria os dispositivos analisados disponibilizam uma interface HTTP. Nestes casos será usada a classe `System.Net.WebClient` da plataforma .NET. No caso do dispositivo não suportar o protocolo HTTP, mas um outro, independentemente este seja proprietário ou não, é usado o componente `WebCommunication` para a abertura e manutenção de *sockets* de comunicação.

Este componente disponibiliza classes para o estabelecimento de ligações como cliente ou servidor. A classe `WebCommunicationClient` é composta pelos seguintes elementos:

- Propriedade `TimeOut` – Especifica o tempo máximo de espera pela resposta da ligação;
- Propriedade `IPAddress` e `Port` – Especifica o endereço IP e a porta com a qual deve ser efectuada a ligação;
- Método `Connect()` – Estabelece a ligação com o servidor especificado. Este método lança um novo *thread* no qual é efectuado o pedido de ligação ao servidor, garantindo desta forma que a execução do código não fica bloqueada até que o servidor responda ou seja retornado o erro por ultrapassar o tempo de espera;
- Método `SendMessage()` – Envia uma mensagem para o servidor. Este método apenas pode ser executado caso haja uma ligação aberta com o servidor;
- Método `ReceiveMessage()` – Bloqueia o *thread* actual até receber uma mensagem por parte do servidor;
- Método `Disconnect()` – Fecha a ligação com o servidor.

A classe `WebCommunicationServer` é semelhante à versão de cliente pelo que não será aqui apresentada.

Driver

O objectivo principal deste componente é definir uma interface de acesso uniforme a qualquer dispositivo integrável na aplicação. A única operação a executar para tornar um dispositivo compatível é implementar uma ou várias das interfaces padrão. Cada tipo de dispositivo identifica-se como disponibilizando uma série de funcionalidades e, desta forma, o software consegue facilmente questionar e determinar quais as capacidades de qualquer dispositivo. Esta abordagem permite uma expansão fácil do número de dispositivos suportados e permite criar um `driver` normalizado que permita posteriormente, o desenvolvimento de dispositivos com suporte nativo à aplicação. Esta última possibilidade não será implementada nesta tese sendo antes um dos requisitos futuros da aplicação.

A arquitectura das interfaces assenta numa estrutura hierárquica que facilita a reutilização de código e agregação de funcionalidades.

`IBaseDrive`

Esta é a interface base que cada `driver` específico tem que obrigatoriamente implementar. Sobre esta interface, comum a todos os dispositivos, devem ser implementadas uma ou mais das interfaces descendentes.

Figura 5.7 - Interface `IBaseDevice`

- Propriedade `Online` – Indica o estado da ligação física ao dispositivo. Este valor está sempre actualizado;
- Propriedade `HasGui` – Indica se o dispositivo representado por esta interface tem alguma interface gráfica com o utilizador. Esta funcionalidade é utilizada na camada de apresentação para decidir quais os dispositivos que são mostrados na área principal;
- Métodos `GetConfigParams`() e `GetAutoConfigParams`() – Têm como função indicar quais os parâmetros que é necessário passar ao dispositivo na sua

inicialização e quais os seus valores por omissão, respectivamente. Na prática essa informação é usada para geração automática de formulários de inserção e configuração de dispositivos;

- Método `OpenDevice`() – Como o nome sugere, o método abre uma ligação física com o dispositivo. Esta ligação é mantida aberta até à chamada do método `CloseDevice`(). Perante um erro na comunicação com o dispositivo a ligação é automaticamente terminada e o estado do objecto passa para *offline*;
- Método `GetSerialNumber`() – Devolve um identificador único do dispositivo. Aquando do estabelecimento duma ligação ao dispositivo é normalmente verificado se o dispositivo é realmente o correcto. A verificação é feita comparando o número de série que está na base de dados com o número de série retornado pelo dispositivo;
- Método `CloseDevice`() – Fecha a ligação física ao dispositivo, caso exista uma, libertando todos os recursos alocados aquando da abertura do canal de comunicação. Este método nunca gera excepções.

`ICamera`

Esta interface é aplicável a dispositivos com capacidade de visualização de imagens ou vídeo, sejam eles, câmaras de rede, servidores de vídeo ou simples webcams, disponibilizadas através do software local.

A interface apresenta uma série de propriedades que permitem indagar quais as capacidades específicas das câmaras, como por exemplo, a existência de motor ou qual a resolução máxima suportada.

- Propriedade `Resolutions` – Lista das resoluções possíveis para a obtenção de imagens;
- Propriedade `HasStreaming` – Indica se o dispositivo suporta a transmissão de vídeo em *streaming*. Esta funcionalidade exige o uso de um *ActiveX* ou *Java Applet* no lado do cliente;
- Propriedade `HasImageSequence` – Indica se o dispositivo suporta o envio de imagens sequenciais;
- Propriedade `HasDirectAccess` – Possibilidade do dispositivo enviar imagens directamente aos clientes, sem passar pelo servidor central. Para tal o dispositivo deve permitir o pedido de imagens através de pedidos HTTP;

- Propriedades `MaxResolutionX` e `MaxResolutionY` – Largura e Altura máximas de uma imagem retornada pelo dispositivo (as medidas estão em pixeis);
- Propriedade `HasPanTilt` – Indica da existência da funcionalidade de *Pan* (movimentação horizontal) e *Tilt* (movimentação vertical);
- Propriedade `HasZoom` – Indica se o dispositivo tem capacidade de zoom, óptico ou digital;
- Método `GetImage()` – Captura uma imagem do dispositivo. Se não for indicada uma resolução para a captura é utilizada a resolução por omissão;
- Método `GetDirectImageURL()` – Devolve o URL da página do dispositivo que retorna uma imagem, em formato JPEG, com as características pretendidas;
- Método `GetHTMLStreaming()` - Devolve código HTML para embeber numa página WEB. O código devolvido é responsável por renderizar vídeo vindo directamente do dispositivo através do uso do controlo *ActiveX* desenvolvido.

Interface `IPtz`

A interface `IPtz` representa a capacidade existente em algumas câmaras de rodar vertical e/ou horizontalmente e alterar o nível de zoom, quer este seja óptico ou digital.

Esta interface não pode ser implementada isoladamente, devendo ser instanciada a partir da interface `ICamera`.

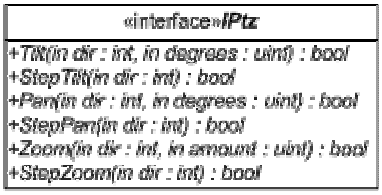

Figura 5.8 - Interface `IPtz`

- Métodos `Tilt()` e `Pan()` – Enviam o comando de rotação vertical (`Tilt`) ou horizontal (`Pan`) ao dispositivo. A rotação é considerada em termos absolutos;
- Métodos `StepTilt()` e `StepPan()` – Estes métodos são semlhantes aos anteriores, com a diferença de a rotação ser efectuada em termos relativos à posição actual. Os valores passados ao método indicam o nível e sentido

da rotação (valores positivos indicam uma rotação para a direita ou para cima, valores negativos, o inverso);

- Método `Zoom`() – Envia um comando de zoom;
- Método `StepZoom`() - Aumenta ou diminui uma unidade ao nível actual de zoom do dispositivo.

Interface `IX10`

Representa a categoria de dispositivos com capacidade para controlar aparelhos X10.

- Propriedades `HouseCode` e `UnitCode` – Retornam a tabela de tradução dos códigos de casa e códigos de aparelho nos endereços X10. A finalidade desta tabela é permitir que, na lógica de negócio, seja utilizada uma representação universal dos endereços X10. Apenas quando se comunica com um dispositivo em particular é que os endereços são traduzidos para o formato do fabricante;
- Método `SetOn()` – Envia um comando X10, ao módulo cujo endereço X10 foi passado como parâmetro, indicando uma ordem do tipo "Ligar aparelho". Todos os módulos configurados nesse endereço deverão executar a ordem;
- Método `SetOff()` – Método equivalente ao anterior mas envia comandos do tipo "Desligar aparelho";
- Método `ExecuteCmdX10()` – Executa um comando X10 genérico, passado como parâmetro, juntamente com o endereço X10 a controlar. A finalidade deste método é possibilitar a introdução de comandos suportados apenas por alguns dispositivos.

Interface `IAlarmConsole`

Esta interface representa todos os dispositivos da categoria de consolas de alarme.

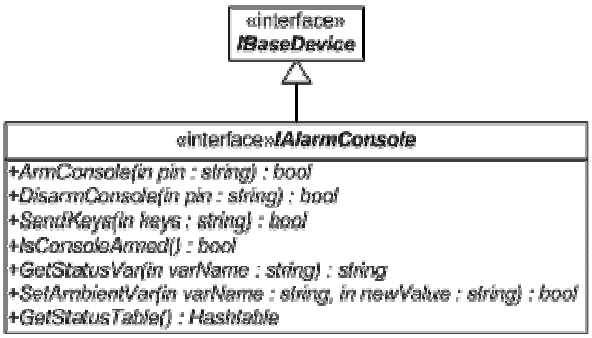

Figura 5.9 - Interface `IAlarmConsole`

- Métodos `ArmConsole()`, `DisarmConsole()` – Ordenam que a consola seja armada ou desarmada, respectivamente;
- Método `SendKeys()` – Envia sequência de teclas para a consola. Emula operação no teclado da consola permitindo o envio de comandos avançados;
- Método `IsConsoleArmed()` – Indica se a consola de alarme está armada;
- Método `GetAmbientVar()` e `SetAmbientVar()` – Permite ler e escrever o valor de uma variável de ambiente;
- Método `GetAmbientTable()` – Obtém o valor das variáveis de ambiente na consola.

Interface `ISensors`

Esta interface aplica-se a dispositivos com capacidade de integrar sensores, sendo estes usados para lançar os eventos configurados pelo utilizador. Esta interface está normalmente associada à interface `IAlarmConsole` ou à interface `ICamera`, não existindo nenhum dispositivo que apenas tenha estas funções. Esta interface é composta apenas por uma propriedade:

- Propriedade `SensorsList` – Retorna uma lista com os sensores suportados pelo dispositivo.

Interface `IVideoServer`

Esta interface foi criada tendo em vista os vídeo-servidores com capacidade para gravarem filmes localmente, em discos internos. Esta interface não dispõe de nenhuma propriedade, apenas dos seguintes métodos:

- `GetRecordingsList()` – Devolve um vector com os dados das gravações disponíveis no servidor. Opcionalmente poderá ser definido um intervalo de tempo;
- `ReleaseRecording()` - Marca uma gravação como estando pronta para ser reescrita ou mesmo apagada. Nem todos os dispositivos dispõem desta funcionalidade;
- `ProtectRecording()` - Garante que a gravação fica no servidor até ao caso extremo de falta de espaço em disco. Igualmente ao método anterior, existem dispositivos sem esta funcionalidade;

- `BackupRecording()` - Guarda no servidor uma cópia do vídeo. Esta operação pode ser bastante demorada;
- `GetFirstFrame()` - Retorna a primeira imagem do vídeo;
- `HtmlPlayerUrl()` - Retorna código HTML necessário para reproduzir uma gravação remota;

Descritas as interfaces que constituem a arquitectura do componente `drivers` descreve-se, de seguida, a classe onde foram implementadas as interfaces base. Todas as classes que implementam uma *gateway* de comunicação com um dispositivo estendem directa ou indirectamente a classe abstracta `Driver`.

Classe `Driver`

Representa logicamente um `driver` dum dispositivo, o qual deve ser entendido como uma entidade que, através da implementação de interfaces predefinidas, expõe funcionalidades próprias a cada dispositivo. Esta é uma classe abstracta que implementa parte da interface `IBaseDevice`. Para cada tipo de dispositivo, é criada uma classe concreta que herda esta classe e implementa todas as suas funções.

A Figura 5.10 representa um diagrama de sequência de um pedido, por parte de um utilizador final, de uma imagem.

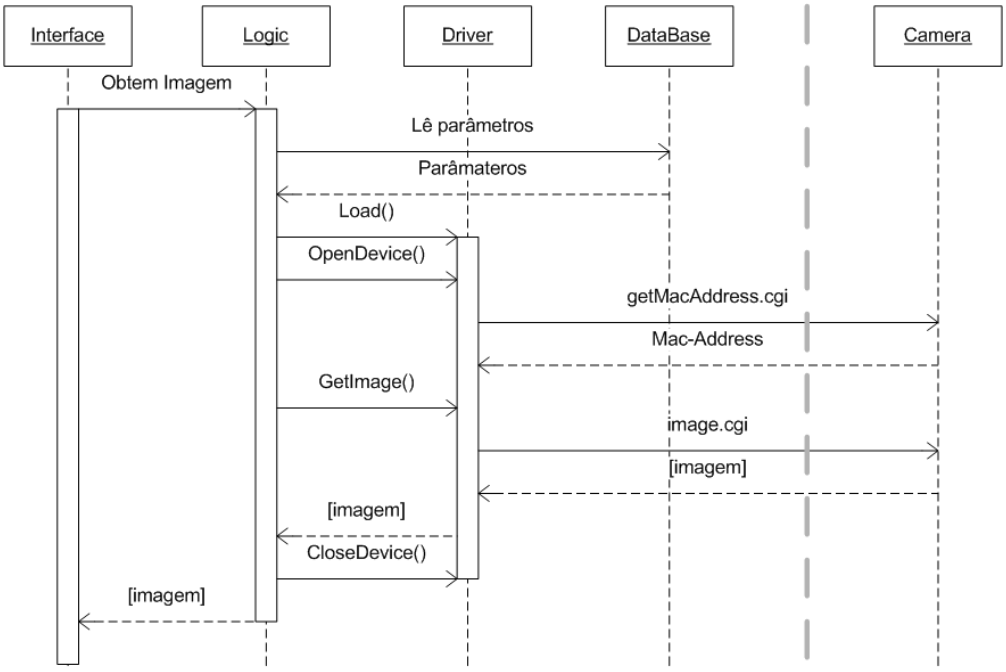

Figura 5.10 - Diagrama de sequência da captura de uma imagem

O processo descrito na figura inicia-se com um pedido do utilizador, efectuado através da interface disponibilizada pela aplicação, para a captura de uma imagem. O pedido é colocado à lógica a qual obtém da base de dados os parâmetros do dispositivo, tais como o tipo de dispositivo (uma *string* que identifica o `driver` a ser instanciado), o *mac-address*, o endereço de ip e a porta a partir da qual está à escuta. Com esta informação é carregado o `driver` do dispositivo (comando `Load`) e é aberta a ligação a este.

Ao abrir a ligação ao dispositivo o `driver` deve verificar a autenticidade e o estado do dispositivo (on-line ou off-line). Para tal pede o número de série (normalmente o *mac-address*) ao dispositivo e confirma-o com o valor lido da base de dados. Se o número de série não for idêntico ao armazenado na base de dados ou o dispositivo não responder ao pedido do `driver`, o estado deste será actualizado para: dispositivo errado ou off-line, respectivamente.

Após confirmar que o dispositivo está ligado é pedida uma imagem. No caso de um dispositivo com uma interface HTTP (será o caso dos dispositivos analisados), este pedido representa o acesso a uma determinada função, onde os parâmetros são enviados no próprio URL do pedido. Após a obtenção da imagem a ligação ao dispositivo é fechada (comando `CloseDevice`), sendo a imagem retornada para a interface.

Na secção 5.6 serão analisados em maior detalhe os comandos necessários para a comunicação com os diferentes equipamentos.

5.4. CAMADA DA BASE DE DADOS

Esta camada é composta pelo componente `DataBase` e pela própria base de dados.

Pacote `DataBase`

Este pacote é o responsável por garantir a interacção com a base de dados, tendo como funcionalidade garantir a persistência em disco dos objectos da lógica de negócio. Este objectivo é conseguido fornecendo métodos para carregar e salvar em base de dados o estado de objectos da lógica. Para maior organização do código desenvolvido e para permitir uma evolução serena da aplicação é fornecida uma interface única para persistência de objectos. Nestes termos todos os objectos da lógica que requeiram persistência descendem da classe abstracta `DbLine`.

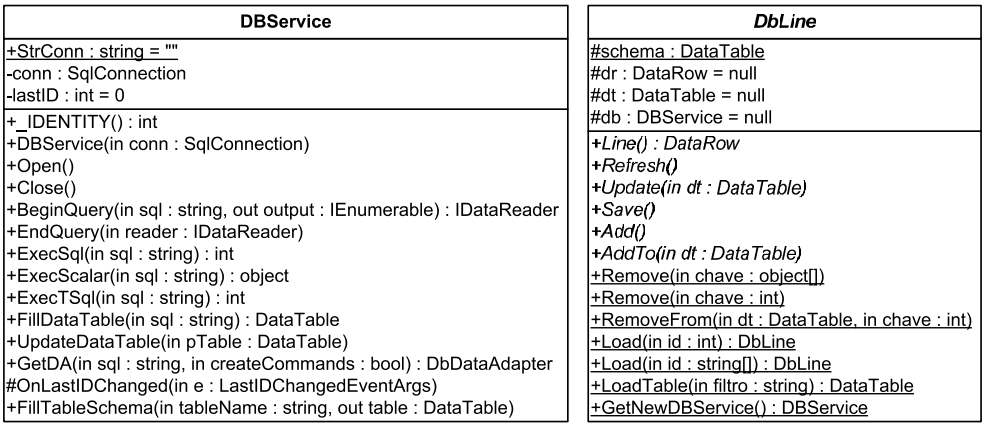

Figura 5.11 - Diagrama de classes do Componente `Database`

Classe `DbLine`

Esta classe define o esqueleto que deverá ser implementado por cada objecto da lógica que necessite de persistência, disponibilizando métodos que permitem inserir, apagar ou alterar registos numa tabela da base de dados.

- Variável `schema` – Após inicialização da classe deverá ficar com o esquema da tabela à qual o objecto pertence;
- Variável `dr` – É do tipo `DataRow` e armazena o estado do objecto;
- Variável `dt` – É do tipo `DataTable` e tem uma referência para uma tabela com o esquema definido em `schema`;
- Variável `db` – por omissão está instanciada num objecto do tipo `DBService`, o qual pode ser usado para aceder à base de dados;

- Propriedade `Line` – Retorna uma representação do objecto através de um outro objecto do tipo `DataRow`;
- Método `Add()` – Adiciona o objecto à tabela correspondente da base de dados;
- Método `Update()` – Actualiza a `DataTable`, passada como parâmetro, adicionando, removendo ou alterando a linha correspondente a este objecto;
- Método `Save()` – Torna persistente o estado do objecto na base de dados. O objecto deverá já ter sido criado como uma linha na base de dados através do método `Add()`;
- Método `Remove()` – Apaga o objecto da base de dados, sendo este identificado pela sua chave primária;
- Método `Load()` – Sendo passado um identificador único do objecto, este método carrega-o em memória. O resultado é um objecto descendente da classe `DbLine`;
- Método `LoadTable()` – Recebe uma pergunta em SQL e devolve os resultados encapsulados num objecto do tipo `DataTable`.

Classe `DBService`

Esta classe concentra todas as funcionalidades requeridas na implementação da classe `DbLine` relacionadas com manipulação de dados em bases de dados *SqlServer*. A implementação da classe baseia-se no API ADO.NET, sendo responsável pela manutenção das ligações à base de dados.

- Construtor da classe – Recebe uma *string* de ligação ou o próprio objecto de ligação à base de dados e inicializa o estado do objecto;
- Propriedade `IDENTITY` – Contém o valor do último *auto number* gerado na base de dados;
- Método `BeginQuery()` – Recebe uma consulta em SQL e devolve um objecto que permite a leitura sequencial dos dados resultantes;
- Método `ExecSql()` – Executa um instrução SQL diferente de uma consulta de dados (SELECT);
- Método `ExecScalar()` – A partir de uma *string* de consulta em SQL devolve o valor da primeira coluna da primeira linha;

- Método `ExecTSql()` – Executa um conjunto de instruções SQL, passadas como parâmetro, segundo a filosofia de transacção. Este método assegura que ambas as operações são executadas com sucesso ou não são executadas de todo. Por exemplo, se fizermos uma pergunta do tipo *Update*, seguido de um *Insert*, caso este último não seja executado, por exemplo devido à ocorrência de um erro, o *Update* também não o é, mantendo a integridade da base de dados. Este método actualiza a variável `_IDENTITY` que armazena a última chave atribuída (usada em situações que é necessário ler o `ID`, atribuído automaticamente a um campo inserido);
- Método `FillDataTable()` – Nas suas várias alternativas o método preenche um objecto `DataTable` a partir da base de dados;
- Método `UpdateDataTable()` – Sincroniza a base de dados com o conteúdo da tabela passada como parâmetro;
- Método `GetDA()` – Devolve um *DataAdapter*. O objecto retornado tem como funcionalidade comunicar com o *driver* da base de dados;
- Método `FillTableSchema()` – Tem funcionalidade semelhante ao `FillDataTable()` com a particularidade de apenas mapear a estrutura da tabela;
- Evento `LastIDChanged` – É disparado após a execução de uma instrução SQL *INSERT*. Tem como objectivo expor o `ID` dos objectos logo após a sua criação (aplicável quando o `ID` é um *autonumber*).

5.5. A CAMADA DA INTERFACE

Tendo em conta o tipo de utilizadores, ou mesmo de dispositivos, que necessitam de remotamente comunicar com a aplicação, foram desenvolvidas três interfaces: interface web, para utilizadores com recurso a PC's, interface móvel, para utilizadores que acedem através de dispositivos móveis e interface suporte, para dispositivos que necessitam de comunicar com a aplicação.

5.5.1. Interface Web

Para o desenvolvimento deste componente, e tendo em conta um dos principais requisitos – a não necessidade de nenhum software no lado do cliente, facilmente se conclui que este deverá ser baseado na norma W3C de forma a ser compatível com a grande maioria dos browsers de Internet.

A principal vantagem da programação de páginas de servidor com recurso à tecnologia ASP.NET prende-se com o facto de, como toda a programação é efectuada ao nível do servidor, a compatibilidade destas páginas é grande, pois não existe a necessidade de incluir código, normalmente em *JavaScript*, no lado do cliente. Esta vantagem acarreta consigo uma grande desvantagem: a necessidade de efectuar um novo pedido ao servidor (*post back*) sempre que um botão, com uma acção associada, for pressionado.

Para cumprir com o requisito não funcional da usabilidade e melhorar os tempos de resposta, tendo como resultado uma melhoria na experiência de utilização do serviço, foram usadas as seguintes técnicas:

- Desactivar o *buffer* de saída em todas as páginas geradas no servidor que contenham listagens longas. Desta forma o servidor retorna o conteúdo à medida que ele é produzido, não forçando o utilizador a esperar que a página termine completamente de ser processada;
- Utilizar sempre que possível mecanismos do lado do cliente que minimizem o número de ligações com o servidor e o refrescamento da interface gráfica. Esta técnica baseia-se no uso de programação no lado do cliente (*Javascript*) para, por exemplo, validar a informação do utilizador antes da submissão dos dados, ou então para permitir a execução de comandos em páginas escondidas (normalmente em *frames* ou *iframes* de dimensão nula);
- *Download* assíncrono de conteúdos. Uma vez que os conteúdos são principalmente dinâmicos não é viável manter uma *cache* de páginas *web*.

Para suavizar a transição para novos conteúdos (por exemplo, na visualização sequencial de imagens) usam-se técnicas de *download* assíncrono, mantendo-se um *buffer* no lado do cliente e apenas mostrando os conteúdos, ou imagens, quando estiverem carregados. Outra possibilidade para o denominado "*download* assíncrono", é a previsão e obtenção antecipada de conteúdos. Esta técnica foi usada, por exemplo, na página de autenticação, onde as maiores imagens usadas na página seguinte (interface principal) são colocadas em *cache*.

Antes de responder a qualquer pedido, com excepção da página de entrada, a aplicação identifica e autentica o utilizador que está a fazer o pedido, estando esse mecanismo representado na Figura 5.12. No caso do utilizador não estar autenticado é redireccionado para a página inicial.

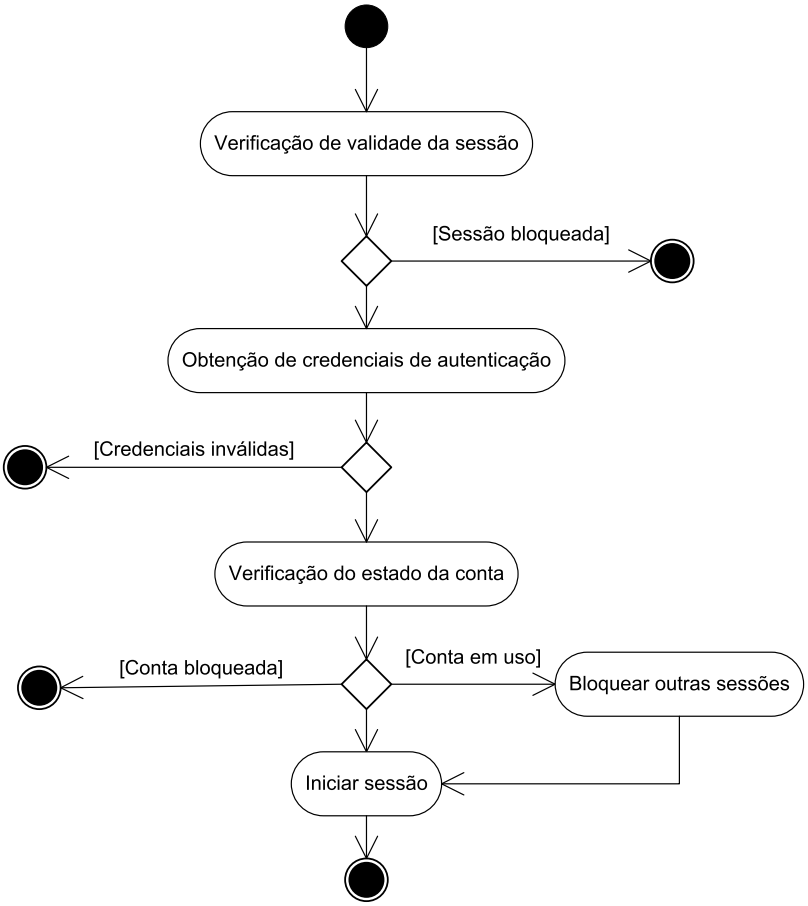

Figura 5.12 - Diagrama de actividades do mecanismo de autenticação

Existem várias razões as quais poderão levar um utilizador a perder a sua sessão na aplicação e, consequentemente, a ser redireccionado para a página inicial de forma a poder autenticar-se novamente. De seguida apresentaremos algumas

dessas razões:

- Caso o utilizador não efectue nenhuma operação durante um período de 20 minutos (tempo por omissão para terminar uma sessão em páginas ASP.NET), a sessão expira;

- Por uma questão comercial e de fiabilidade do serviço, alguns dos produtos disponibilizados não permitem o acesso simultâneo a múltiplos utilizadores. Nestes casos, se outro utilizador entrar na aplicação com as mesmas credenciais, irá terminar a sessão do utilizador existente;

- Caso o pagamento do serviço não seja efectuado, a conta é bloqueada pelo administrador. Esta opção não só termina a sessão de um utilizador como também impede o seu acesso.

Conforme referido anteriormente, de forma a evitar os incómodos *post-backs* do uso de programação no servidor, optou-se pelo uso de *frames* e *iframes* escondidas, conforme indicado na Figura 5.13. Esta opção traz como vantagem acrescida a flexibilidade em termos de tamanho de ecrã, pois quando o conteúdo do interior das *iframes* é superior ao seu tamanho são criadas barras de *scroll* automaticamente.

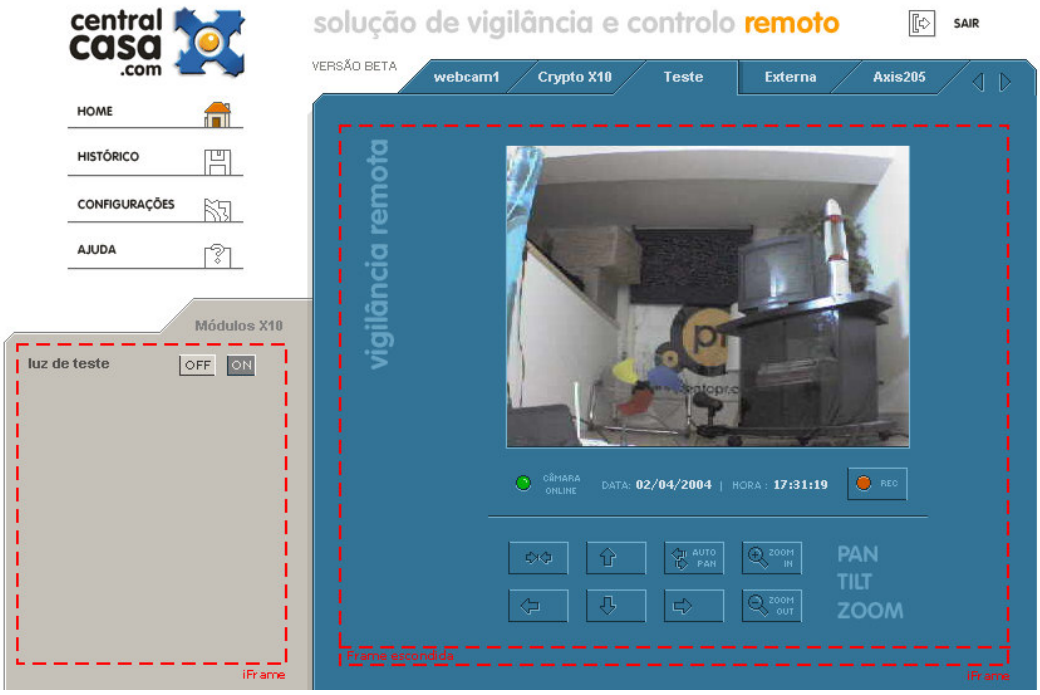

Figura 5.13 - Interface da Página Principal

Seguindo o mesmo principio inseriu-se uma *frame* escondida, na página do conteúdo central, para a execução de comandos. Aos botões existentes na página principal, tais como os botões o controlo PTZ da câmara ou captura de imagem, foram associadas funções desenvolvidas em *JavaScript* que chamam uma página - `command.aspx`, e enviam por *GET* (inseridas no *URL*) as variáveis de comando para o servidor. Este, por sua vez, ao interpretar a página chamada executa o comando através da lógica de negócio. Esta solução permite que as acções sejam executadas no servidor sem a necessidade da área visível sofrer um *reload*, sendo que os únicos botões que obrigam a um *reload* completo da página são os botões do menu.

Após o desenvolvimento da solução anteriormente descrita detectou-se que, na execução de comandos demorados, tais como um comando de captura de imagem, caso o utilizador volta-se a premir o mesmo ou outro botão, a execução da acção seria parada, pois a página `command.aspx` seria novamente chamada. Para evitar esta situação, criou-se uma variável – `Active`, booleana, que quando contem o valor "falso" não permite a execução de comandos. Desta forma, cada função associada a um botão responsabiliza-se por colocar esta variável com o valor falso. Assim que o comando for executado e for retornada a página na *frame* escondida, esta trata de alterar o valor da variável, através da função `activate()`,novamente para "verdadeiro". Para evitar que, caso ocorra um erro, nenhum comando possa ser executado, é usada a função `window.setTimeout("activate()", 5000)` para que, após o decorrer de 5 segundos, a função `activate()` seja obrigatoriamente executada.

Para o desenvolvimento da página do histórico optou-se novamente pelo uso de *iframes* e *frames*, mas desta vez visíveis. Neste caso a listagem de imagens foi desenvolvida como uma página separada e inserida dentro de uma *iframe*, tornando-se possível navegar nas imagens. Esta *iframe* também é usada para executar alguns dos comandos, tais como, filtrar, alterar o nome, proteger, apagar e apagar tudo.

5.5.2. Interface móvel

O subsistema `web` permite que um utilizador, a partir de um *browser* possa aceder a todas as funcionalidades da aplicação. Na realidade não é possível estender o leque de *browsers* suportados a *browsers* presentes em dispositivos móveis, como PDAs, Smartphones e telemóveis, pois não só a capacidade de processamento e memória é escassa, como o ecrã não permite mostrar tanta informação. É por essa

razão que existe a versão móvel da aplicação, adaptada para este tipo de dispositivos e com um menor número de funcionalidades.

O subsistema `Mobile` implementa justamente uma interface da aplicação para dispositivos móveis. No âmbito dos dispositivos móveis surge o problema da diversidade de formatos e capacidades dos equipamentos encontrados no mercado. Para resolver o problema, a implementação do serviço móvel será baseada na tecnologia *ASP.NET Mobile Controls* [ASP.NET Mobile, 04]. Esta tecnologia permite escrever conteúdo para dispositivos móveis sem conhecer à partida as características específicas suportadas pelo aparelho. Em *runtime* os conteúdos são formatados de acordo com as capacidades do equipamento.

Tendo em conta o tipo de uso de um dispositivo móvel, e as suas limitações, principalmente no que se refere à inserção de dados (poucos são os dispositivos que disponibilizam um teclado QWERTY) e tamanho do ecrã, algumas funcionalidades serão limitadas.

A tecnologia *ASP.NET Mobile Controls* obriga a manutenção de um ficheiro de configurações onde estejam armazenados todos os dispositivos móveis e as suas características. Apesar da Microsoft divulgar com alguma frequência (menor que a desejada) actualizações para este ficheiro [MobileControls, 04], estas não são suficientes, sendo necessário criar os nossos próprios ficheiros que incluam os últimos modelos de telemóveis comercializados em Portugal. De seguida é mostrado um exemplo de um ficheiro de configurações `machine.config`, para um dispositivo da marca *SonyEricsson,* modelo *P800*:

```
<browserCaps>
   <case
      match="P800"
      with="${deviceID}">
      breaksOnInlineElements = "false"
      browser = "SonyEricsson"
      cookies = "true"
      inputType = "virtualKeyboard"
      isColor = "true"
      mobileDeviceManufacturer = "SonyEricsson"
      preferredImageMime = "image/jpeg"
      preferredRenderingMime = "application/xhtml+xml"
      preferredRenderingType = "xhtml-basic"
      screenBitDepth = "24"
      screenCharactersHeight = "16"
```

```
      screenCharactersWidth = "28"
      screenPixelsHeight = "320"
      screenPixelsWidth = "208"
      supportsCss = "true"
      supportsFontSize = "true"
      supportsItalic = "true"
      tables = "true"
    </case>
```

As funcionalidades disponíveis na versão móvel não incluem quaisquer configurações nem administração, estando centradas na vigilância e controlo remoto. Atendendo à portabilidade destes dispositivos, estes são aproveitados para notificar o utilizador da ocorrência de eventos importantes, tais como detecção de intrusão, através do envio de mensagens em formato SMS ou MMS.

Devido às limitações dos dispositivos e também para reduzir o tráfego consumido, as imagens transmitidas são estáticas, não existindo nenhum mecanismo que efectue automaticamente o refrescamento das imagens. Para o utilizador refrescar a imagem, deve pressionar sobre esta, forçando um refrescamento da página e, consequentemente, um refrescamento da imagem.

De forma a redimensionar a imagem de acordo o dispositivo que a está a apresentar, foi desenvolvido um componente – `Mobile Dynamic Image`. Este componente analisa as características do dispositivo, através da classe `MobileCapabilities`, definindo a largura da imagem de acordo a largura do ecrã do dispositivo, sendo a altura estipulada de forma a manter a mesma proporção que a imagem original.

```
MobileCapabilities caps = (MobileCapabilities)Request.Browser;
String deviceModel = caps.MobileDeviceModel;

…

System.Drawing.Image img = dev.LastPicture;

int ecranW = caps.ScreenPixelsWidth;
float sx = (ecranW / (float)img.Width);
float aspectRatio = ((float) img.Width) / ((float) img.Height);

System.Drawing.Bitmap bmp = new Bitmap(img,new Size((int)(sx *
img.Width),(int)( (sx*img.Width) / aspectRatio )));
ImageCodecInfo[] icf = ImageCodecInfo.GetImageEncoders();
```

```
EncoderParameters encps = new EncoderParameters( 1 );
EncoderParameter encp = new EncoderParameter(
System.Drawing.Imaging.Encoder.Quality, (long)75);

//Set quality
encps.Param[0] = encp;

try
{
   foreach(System.Drawing.Imaging.ImageCodecInfo codec in icf)
   {
   if (codec.MimeType == caps.PreferredImageMime)
   {
       bmp.Save( Response.OutputStream,codec,encps );
       break;
   }
}
Response.ContentType = caps.PreferredImageMime;

}
catch(Exception ex)
{
   Logic.Debug.EventLogs.Write("Erro no envio da imagem para mobile ", ex,
"Visaowebmobile_LastImage.aspx.cs:Page_Load()");
}
   finally
{
   bmp.Dispose();
}
```

5.5.3. Interface Suporte

Dentro da camada da interface existe um componente - `web.support`, que não se destina à interacção com o utilizador, mas sim à interacção com dispositivos.

A comunicação com dispositivos não existe sempre por iniciativa da aplicação. Por exemplo, quando uma consola de alarme detecta movimento, terá que notificar a aplicação para que esta possa verificar se existe algum evento a ser executado. Nestas situações é necessário disponibilizar um mecanismo que esteja à escuta destas mensagens.

Conforme referido no capítulo 3.1, as possibilidades são diversas. Actualmente o

estado de arte aponta para a utilização de *WebServices*, existindo também a possibilidade de *RPC's*, *.NET Remoting* ou HTTP. O uso de *WebServices* é o mecanismo mais adequado, mas não pôde ser aplicado devido à falta de capacidade de processamento e de recursos nos dispositivos. As soluções baseadas em *RPC's* ou *.NET Remoting* não são aplicáveis por falta de compatibilidade nos dispositivos. O mecanismo escolhido para escutar a notificação dos dispositivos foi o HTTP por ser amplamente suportado pela maioria dos dispositivos compatíveis com a aplicação, pouco exigente e simples de implementar.

Actualmente existem três casos onde é necessária a comunicação dos dispositivos com a aplicação. Para cada um deles foi desenvolvida uma página, conforme indicado na tabela seguinte, que recebe os pedidos e processa-os através da Lógica de Negócio. No futuro, poderão vir a ser implementadas mais interfaces que possam receber novas notificações de dispositivos mais evoluídos que disponibilizem outro tipo de funcionalidades.

Tabela 6 – Interface de suporte

Interface	Função
`alarmNot.aspx`	Interface responsável por receber as notificações de alarme de todos os dispositivos que implementam a interface `Isensors`, como por exemplo, consolas de alarme ou câmaras com detecção de movimento digital;
`X10Not.aspx`	Página que recebe todas as notificações sobre alterações de estado de dispositivos X10;
`updateIp.aspx`	Esta é a interface disponibilizada para que os dispositivos possam actualizar o seu ip na base de dados.

Para suportar cenários em que a ligação dos dispositivos locais, instalados no cliente sobre vulgares linhas de banda larga, é baseada em IP dinâmico (significa que o endereço IP do cliente pode variar ao longo do tempo de forma não previsível), há a necessidade do dispositivo enviar regularmente o seu endereço actual para a aplicação.

O intervalo de tempo entre estas notificações deverá ser o menor possível para maximizar a disponibilidade do dispositivo, tendo em atenção que intervalos muitos curtos criarão sobrecargas nos servidores, gerando tráfego redundante. A solução implementada é uma solução de meio-termo. Optou-se por manter as notificações em intervalos curtos (5 minutos), mas implementou-se um sistema de forma a

minimizar as pesquisas e actualizações desnecessárias na base de dados.

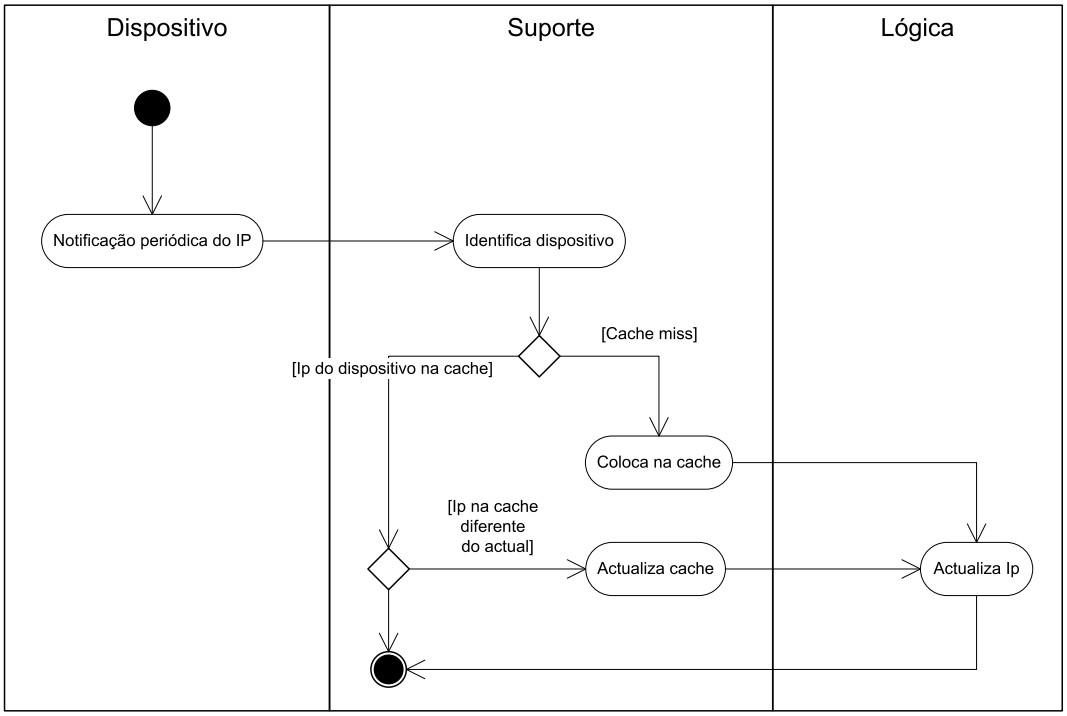

Figura 5.14 - Diagrama de actividades da notificação de alteração de IP

Uma alternativa à solução apresentada baseia-se na tecnologia UPnP (*Universal Plug'n'Play*) [ref]. Com esta tecnologia seria possível que a própria *gateway* existente entre o dispositivo e a Internet, notificasse o servidor quando detectasse alteração do seu IP externo. Esta alternativa não foi adoptada pois a maioria dos dispositivos usados ainda não suporta a norma UPnP [Jeronimo, 03].

5.6. COMUNICAÇÃO COM OS DISPOSITIVOS

Neste capítulo serão analisados os problemas e soluções adoptadas para a comunicação, através dos protocolos TCP/IP, com os diferentes dispositivos.

Assim que se começou o desenvolvimento de cada `driver` individual, facilmente se compreendeu que os dispositivos de mais fácil integração são os que disponibilizam uma API HTTP. Desta forma a comunicação com os dispositivos baseia-se na troca de mensagens HTTP, onde os parâmetros são normalmente enviados através do método GET, ou seja, são inseridos no URL com o qual se faz o pedido.

Antes de se iniciar a comunicação com um dispositivo deve-se primeiro abrir a ligação a este através do método `OpenDevice()` (consultar secção 5.3). Este método é o responsável por verificar se o dispositivo está disponível (*on-line*) ou indisponível (*off-line*). O comando usado para esta verificação é o pedido do número de série (usualmente o *Mac-address* do dispositivo), pois não só garante que existe um dispositivo a responder ao pedido como também indica se o dispositivo é o correcto.

Cada `driver` implementa uma ou várias interfaces dependendo do tipo de dispositivo. De seguida serão analisadas as interacções com alguns tipos de dispositivos.

Axis 210

Conforme indicado na secção 4.2, este dispositivo implementa as interfaces `ICamera` e `ISensors`, sendo fácil de compreender que a funcionalidade mais importante neste tipo de dispositivos é a captura de imagens em tempo real.

A Axis facilita a integração do produto pois disponibiliza uma API muito completa para interacção com o dispositivo. Para capturar uma imagem neste dispositivo basta efectuar um pedido GET, conforme exemplificado de seguida:

http://<*servername*>/axis-cgi/jpg/ image.cgi[?<*parameter*>=<*value*>[&<*parameter*>=<*value*>...]]		
<parameter>=<value>	**Valores**	**Descrição**
resolution=<int>x<int>	<*width*>,<*height*>	Retorna uma imagem com <*width*> pixeis de largura e <*height*> pixeis de altura.
compression=<int>	0 - 100	Define a compressão JPEG da imagem. 100 corresponde a compressão máxima e pior qualidade.

date=<int>	0, 1	Mostra ou esconde o *timestamp* com a data/hora: 0 = esconde, 1 = mostra
quad=<int>	0, 1	Devolve uma imagem em *Quad*: 0 = normal, 1 = quad
text=<int>	0, 1	Mostra ou esconde o *timestamp* com o texto: 0 = esconde, 1 = mostra

Por exemplo, se pretendermos uma imagem com uma resolução de 320x240 e uma compressão de 25, deveremos efectuar o seguinte pedido:

```
http://<servername>/axis-cgi/jpg/image.cgi?resolution=320x240&compression=25
```

Quando é solicitada uma imagem, o dispositivo retorna a respectiva imagem ou um erro formatado de acordo a norma HTTP:

```
HTTP/1.0 200 OK\r\n
Content-Type: image/jpeg\r\n
 [ Content-Auth: <authorization information>\r\n ]
\r\n
<JPEG image data>\r\n
```

Este dispositivo suporta a transmissão de vídeo (sequência de imagens) através do formato MJPEG (*Motion JPEG image stream*). Como apenas os *browsers* da *Netscape* e o *Mozilla* suportam nativamente a renderização deste formato de vídeo, é necessário fornecer um componente *ActiveX* para permitir a visualização de vídeo nos browsers da *Microsoft*. De forma a tornar a aplicação compatível com diferentes dispositivos optou-se por desenvolver um componente que inclui os diferentes *ActiveX* disponibilizados pelos fabricantes dos diferentes dispositivos. Para não alongar em demasia a presente dissertação, o desenvolvimento deste componente não será descrito neste documento.

Para usar o referido componente é disponibilizado o método `GetHTMLStreaming()` que retorna o código HTML necessário para o instanciar e inicializar.

O método disponibilizado pelo dispositivo para retornar a sequência de imagens em formato MJPEG é o seguinte:

http://<servername>/mjpg.cgi[?<parameter>=<value>[&<parameter>=<value>...]]		
<parameter>=<value>	**Valores**	**Descrição**
resolution=<int>x<int>	*<width>*,*<height>*	Retorna um vídeo com <width> pixeis de largura e <height> pixeis de altura.

compression=<int>	0 - 100	Define a compressão JPEG da imagem. 100 corresponde a compressão máxima, menor dimensão de imagem e pior qualidade.
date=<int>	0, 1	Mostra ou esconde o *timestamp* com a data/hora: 0 = esconde, 1 = mostra
duration=<int>	0, ...	Define, em segundos, o tempo durante o qual o vídeo será enviado para o cliente. Se este parâmetro for omitido ou igual a 0, a duração é ilimitada, terminado apenas quando o pedido for fechado.
nframes=<int>	1, ...	Define quantas imagens serão enviadas para o cliente. Se este parâmetro for omitido ou igual a 0, o *stream* termina quando o pedido for fechado.
fps=<int>	1, …	Define o número de imagens por segundo (desejado) do vídeo a ser retornado. Normalmente este valor será sempre limitado pela largura de banda, mas poderá ser usado para limitar o uso da largura de banda. Poderão ser enviados valores inferiores a 1. Por exemplo, 0,5 -> 1 imagem de dois em dois segundos.

Axis 2130

Este dispositivo, para além de implementar as interfaces `ICamera` e `Isensor` de uma forma idêntica ao dispositivo anterior, implementa a interface `IPtz`. Para permitir movimentar o dispositivo segundo os eixos horizontal e vertical, assim como para controlar o nível de zoom, é disponibilizada a seguinte interface:

http://<servername>/ptz.cgi?<parameter>=<value>[&<parameter>=<value>...]		
<parameter>=<value>	**Valores**	**Descrição**
move=<string>	up, down, left, right, upleft, upright, downleft, downright	Move a câmara um valor pré-definido (aproximadamente 5º, dependendo do dispositivo), numa determinada direcção.
pan=<float>	-180.0 a 180.0	Desloca a câmara para a posição enviada (em graus) segundo o eixo horizontal.
tilt=<float>	-180.0 a 180.0	Desloca a câmara para a posição enviada (em graus) segundo o eixo vertical.
zoom=<int>	1 a 9999	Define o nível de zoom para o valor enviado.
focus=<int>	1 a 9999	Define o nível de focus para o valor enviado.

rpan=<float>	-360.0 a 360.0	Desloca a câmara x graus, segundo o eixo horizontal, relativamente à sua posição.
rtilt=<float>	-360.0 a 360.0	Desloca a câmara x graus, segundo o eixo vertical, relativamente à sua posição.
rzoom=<int>	-9999 a 9999	Aumenta (valores positivos) ou diminuí (valores negativos) o nível de zoom.
rfocus=<int>	-9999 a 9999	Aumenta (valores positivos) ou diminuí (valores negativos) o nível de focus.
autofocus=<string>	on, off	Activa (on) ou desactiva (off) o auto-focus.
autoiris=<string>	on, off	Activa (on) ou desactiva (off) a auto-iris.
autopan=<string>	on, off	Activa (on) ou desactiva (off) o *autopan* – deslocação automática e repetida, segundo o eixo horizontal
setpreset=<int>	1,…	Memoriza a posição actual.
removepreset=<int>	1,…	Remove a posição predefinida.
gotopreset=<int>	1, ...	Desloca a câmara para a posição definida.
query=<string>	position, presetposcam, presetposall	Retorna a informação solicitada. A resposta ao pedido de <position> deve ser do tipo: pan=<int>&tilt=<int>&zoom=<int>&focus=<int>

Para movimentar a câmara podemos usar posições absolutas (comandos *pan* e *tilt*) ou posições relativas à posição actual (*rpan* e *rtilt*). Esta última opção foi a escolhida pois não obriga à manutenção em memória da posição actual da câmara. Para o uso destes comandos é necessário enviar o deslocamento pretendido, medido em graus. Este valor foi inicialmente definido como uma constante fixa mas, com a utilização do dispositivo, chegou-se à conclusão de que o valor do deslocamento deveria ser inversamente proporcional ao nível de zoom. Isto é facilmente perceptível, pois quanto maior o nível de zoom menor deverá ser o deslocamento, correndo-se o risco de cada movimento ser superior à área visível da câmara.

Actualmente não estão implementados o controlo da focagem e da íris, usando-se as opções automáticas, as quais já vêem activadas por omissão. A memorização de posições (*presets*) será uma das funcionalidades a incluir num futuro próximo.

Pixord 4000

Este dispositivo é caracterizado por dispor de 4 entradas de vídeo analógicas, podendo representar até 4 dispositivos independentes na aplicação, ou seja, poderemos criar na aplicação 4 dispositivos diferentes.

A interface deste dispositivo é bastante mais simples e limitada que a interface da Axis, tendo como principal diferença o parâmetro `camid` que permite escolher a câmara que pretendemos visualizar. O dispositivo também difere na forma de escolha da resolução. Para tornar a aplicação flexível e compatível com diversos dispositivos e distintas resoluções, o `driver` armazena 3 conjuntos de resoluções (`ImageSize.Small`, `ImageSize.Medium` e `ImageSize.Big`) as quais são usadas em diferentes áreas da aplicação.

http://<*serverIP*>/images<camid><imageSize>		
Parâmetro	**Valores**	**Descrição**
camid	1-4	Escolhe a entrada de vídeo a partir da qual deve ser obtida a imagem
imageSize	"qsif", "sif", "full"	Define a dimensão da imagem: qsif – NTSC=176x112 PAL=176x144 sif – NTSC=352x240 PAL =352x 288 full - NTSC=704x480 PAL=704x576

Este dispositivo também suporta o envio de *Motion JPEG*, usando para a visualização o mesmo *ActiveX* referido anteriormente.

Axis 2460

Este dispositivo é semelhante ao anterior, contendo 4 entradas analógicas, diferindo no facto de dispor de discos internos onde são armazenados os vídeos capturados pelas câmaras. Este dispositivo é o único, dos aqui analisados, que implementa a interface `IVideoServer`.

O dispositivo, seguindo a linha da marca, disponibiliza uma API HTTP a qual nos permite, para além de capturar imagens, listar os vídeos armazenados localmente.

http://<*servername*>/reclist.cgi?<*parameter*>=<*value*>[&<*parameter*>=<*value*>...]		
<parameter>=<value>	**Valores**	**Descrição**
camera=<int>	1, ...	Listar apenas os vídeos da câmara x. Este valor pode ser omitido, sendo listados todos os vídeos de todas as câmaras.
event=<string>	motion, schedule, manual.	Listar os vídeos que ocorreram devido a um determinado tipo de evento.
from=<datetime>	dd/mm/yyyy-hh:mm	Listar apenas os vídeos começados após a data definida.
to=<datetime>	dd/mm/yyyy-hh:mm	Listar apenas os vídeos anteriores à data definida.

Por exemplo, se pretendermos obter todos os vídeos capturados pela câmara 2, após 01/12/2004, deveremos efectuar o seguinte pedido:

```
http://<servername>/reclist.cgi?camera=2&from=01/12/2005-00:00
```

Este pedido retorna uma sequência de variáveis que identificam os vídeos e as suas características:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n

count=<count>\r\n
\r\n

recordingid=<recordingid>\r\n
cameraid=<cameraid>\r\n
eventtype=<eventid>\r\n
length=<length>\r\n
startTime=<dateTime>\r\n
framerate=<framerate>\r\n
secured=<secured>\r\n
quality=<quality>\r\n
stopquality=<stopquality>\r\n
\r\n

... repete-se para cada vídeo retornado
```

Após obtermos uma listagem de todos os vídeos poderemos reproduzi-los usando a seguinte interface:

`http://<servername>/player.cgi?<parameter>=<value>[&<parameter>=<value>... ]`

<parameter>=<value>	**Valores**	**Descrição**
recordingid=<int>	> 0	O ID do vídeo a ser reproduzido.
camera	1, ...	Escolhe a entrada de vídeo a partir de qual deve ser obtida a imagem.
frame=<int>	frame id	Retorna a imagem correspondente à *frame* definida.
direction=<string>	forward, backward	Reproduz o vídeo normalmente ou da frente para trás.
step=<int>	*MJPG video stream* > 0	Indica a velocidade de reprodução: Por omissão: 1 (reproduz todas as *frames*)

Esta interface retorna uma imagem, se especificarmos o parâmetro `frame`, ou um *stream* em *MJPEG*. Conforme referido anteriormente, para visualizar um *stream* é necessário o uso de um *ActiveX*. Neste caso particular, dado que o componente disponibilizado pela Axis dispõe de outras funcionalidades, tais como a definição, em tempo real, da direcção e velocidade de reprodução do vídeo, é implementado o

método `HtmlPlayerUrl()`, que retorna o código HTML necessário para instanciar o *ActiveX*.

Consola de alarme

A consola de alarme da Jablotron, ao incluir um comunicador IP, pode ser incorporada na aplicação. Para este tipo de dispositivos, e conforme referido na secção 5.3, foi criada a interface `IAlarmConsole`.

Este dispositivo, identicamente aos anteriores, disponibiliza uma API HTTP, a qual, por razões de segurança, força sempre o uso de autenticação:

http://<*serverIP*>/alarm?c=<*command*>&p=<*pin*>		
Parâmetro	**Valores**	**Descrição**
command	[0..9, A..Z]	Este parâmetro identifica a função do módulo do alarme: A -> Arma o alarme (p=<codigo>); D -> Desarma o alarme (p=<codigo>); G -> Pede o valor de variáveis. (p=<variavel>) se `p` for omitido, são enviadas todas as variáveis; T -> Envia teclas (p=<teclas>).
Se c=A -> p=<codigo> Se c=D -> p=<codigo> Se c=G -> p=<variável> Se c=T -> p=<teclas>	<codigo>= [0..9] [0..9] [0..9] [0..9] <variavel>= [0..9] [0..9] <teclas>= [0..9,N,F]*	**Código:** Sequência de 4 dígitos que são a chave da consola. **Variável:** Identificador da variável: 00: *Armed*, 01: *Battery*, 02: *Alarm*, 03: *Fault*, 04: *Antena*, 05: *Tamper*, 06: *Power*, 07: *Display* **Teclas:** Teclas a enviar para a consola.

O dispositivo responde à recepção do comando enviando uma variável, denominada de RSP, a qual pode ter os seguintes valores:

- 20 - (OK) comando recebido com sucesso;
- 21 - (OK) comando recebido com sucesso e inserido no buffer de transmissão para consola de alarme;
- 50 - (ERRO) sem memória para receber (tentar mais tarde);
- 51 - (ERRO) comando recebido mas *buffer* de transmissão cheio.

Por exemplo, caso a aplicação necessite de saber o estado da consola, deve enviar o seguinte comando:

```
http://<servername>/alarm?c=G&p=01
```

A resposta do dispositivo poderia ser:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
RSP=20\r\n
Armed=1\r\n
```

Convém salientar que este dispositivo também permite o envio de comandos X10 para a rede eléctrica, sendo a sua interface idêntica ao dispositivo seguinte.

Software local

Conforme referido inicialmente, o software local não é propriamente um dispositivo físico, mas antes um dispositivo virtual.

O software local implementa um pequeno servidor web, que recebe comandos através de pedidos GET. Caso o pedido represente um pedido de imagem, o software comunica com uma webcam, compatível com o WDM (*Windows Driver Model*), para capturar uma imagem ou um stream de vídeo, e envia-o através do protocolo HTTP. Se for um pedido para controlo de um aparelho, o software comunica com o interface X10, através da porta de série, para enviar o comando X10 para a rede eléctrica.

De forma a não alongar em demasia este, o desenvolvimento desta aplicação local não será analisado na presente tese, sendo apenas analisada a sua comunicação com a aplicação no servidor.

Dado que este dispositivo (software) foi totalmente desenvolvido de raiz, desenvolveu-se uma interface semelhante às analisadas, no que respeita à transmissão de imagens e vídeo, não sendo portanto descrita.

Este software permite, através de um dispositivo de comunicação entre a porta de série do PC e a rede X10, o controlo de dispositivos X10, sendo disponibilizada para o efeito a seguinte interface:

http://<*servername*>/x10.cgi[?<parameter>=<value>[&<parameter>=<value> ...]]		
Parâmetro	**Valores**	**Descrição**
address=<houseCode><unitCode>	[A..P] [1..16] ou [A..P] 0	O endereço X10 usado no comando de controlo. Nos comandos ALL_ON e ALL_OFF a segunda parte do endereço (unitCode) é ignorada.

function=<x10Function>	ON, OFF, ALL_ON, ALL_OFF, GET.	Executa o comando X10: • ON – liga o módulo; • OFF – desliga o módulo; • ALL_ON – liga todos os módulos no "houseCode" especificado; • ALL_OFF – desliga todos os módulos no "houseCode" especificado; • GET – retorna informação sobre o estado do módulo endereçado.

Por exemplo, para ligar um dispositivo X10, com o código B5, é necessário enviar a seguinte instrução:

```
http://<servername>/x10.cgi?address=B5&function=on
```

5.7. MECANISMO DE CONFIGURAÇÕES

Durante o desenvolvimento da aplicação, em especial no período final, existiram duas versões em simultâneo da aplicação. Uma interna usada para testes e para desenvolvimento de novas funcionalidades; outra a correr em ambiente real, que era e será a versão oficial do serviço. Será procedimento comum uma versão da aplicação a correr no ambiente de desenvolvimento migrar para o ambiente real, originando uma nova *release*. De modo a facilitar este processo há a necessidade de separar todas as configurações da implementação.

Para o isolamento das configurações o primeiro passo é retirar todas as constantes embutidas no código fonte e colocá-las num repositório separado. De forma a evitar a leitura de um repositório externo cada vez que é requerido um valor das configurações, deverão ser utilizadas classes estáticas cuja função é ler todas as configurações no momento em que são inicializadas. Durante a execução normal da aplicação essas classes poderão ser consultadas de forma a obter qualquer parâmetro de configuração. Este modelo apresenta uma limitação relacionada com a sincronização das classes fornecedoras de valores de configuração e do repositório onde se encontram as configurações. Uma vez inicializada a classe das constantes estas não são mais relidas. A solução para essa limitação está em "reciclar" a memória cada vez que seja detectada uma alteração no repositório de configurações. O processo de reciclagem é automático no caso da plataforma .NET.

O referido repositório de configurações pode ser mantido pelo sistema operativo, isto é, no *registry* no Windows, ou ser apenas um ficheiro em disco. O paradigma mais recente, usado na plataforma .NET, aponta para o uso de um ficheiro uniformizado, estruturado em XML. A vantagem deste ficheiro é ser facilmente alterável, quer por ferramentas automáticas quer por administradores de sistema, e ser facilmente migrável entre máquinas. De seguida é apresentado um pequeno excerto de um dos ficheiros de configurações da aplicação:

```
...
<appSettings>
    <add key="UserHomesDirPath" value="D:\Centralcasa\Visaoweb\Homes\" />
    <add key="UserHomesDirURL" value="/web/Homes/" />
    <add key="UserTextsFile" value="D:\Centralcasa\Visaoweb\Texts.xml" />
    <add key="UserLogsFile" value="D:\Visaoweb\logs_aspx.xml" />
    <add key="DataBaseConnectionString" value="user id=***; password=***;
initial catalog=visaoweb;data source=TAZ;Connect Timeout=30" />
```

```
    <add key="MailServer" value="taz" />
    <add key="MailFrom" value="info@centralcasa.com" />
    <add key="ConnectionTimeout" value="40" />
  </appSettings>
...
```

5.8. ANÁLISE DE DESEMPENHO

Com este capítulo pretende-se descobrir quais os limites de utilização da aplicação e os principais pontos de estrangulamento. Os gráficos aqui apresentados foram obtidos através da aplicação *Web Applications Testing* - versão 3.0, a qual simula diversos pedidos ao servidor, usando utilizadores virtuais.

Para os testes efectuados foram usadas 3 máquinas que simulavam pedidos aos servidores. Para as simulações efectuadas foram definidos testes que simulavam o acesso de 10 a 1000 utilizadores. De forma a estabilizar os resultados, em cada cenário foram realizadas três iterações, e todos os pedidos foram efectuados através de uma rede local de 100MBps.

Uma das limitações da simulação é o facto desta simular apenas o acesso a 3 contas diferentes, não reflectindo uma situação real onde cada utilizador tem as suas próprias credenciais. Esta limitação reflecte-se principalmente na comunicação com os dispositivos, pois é necessário para cada cliente verificar se o mesmo dispositivo está disponível, existindo uma limitação nos poucos dispositivos simulados.

Numa primeira versão, todas as imagens dos dispositivos eram inicialmente enviadas para o servidor e depois reenviadas para o cliente. Esta solução foi criada para satisfazer um dos principais requisitos: a compatibilidade, permitindo a dispositivos que não disponibilizam interface HTTP possam visualizar imagens sem ser necessário usar um *ActiveX* ou *Java Applet*. Para esta primeira versão foi efectuado um teste de carga, sendo apresentado de seguida o gráfico de largura de banda usada em função do número de utilizadores activos.

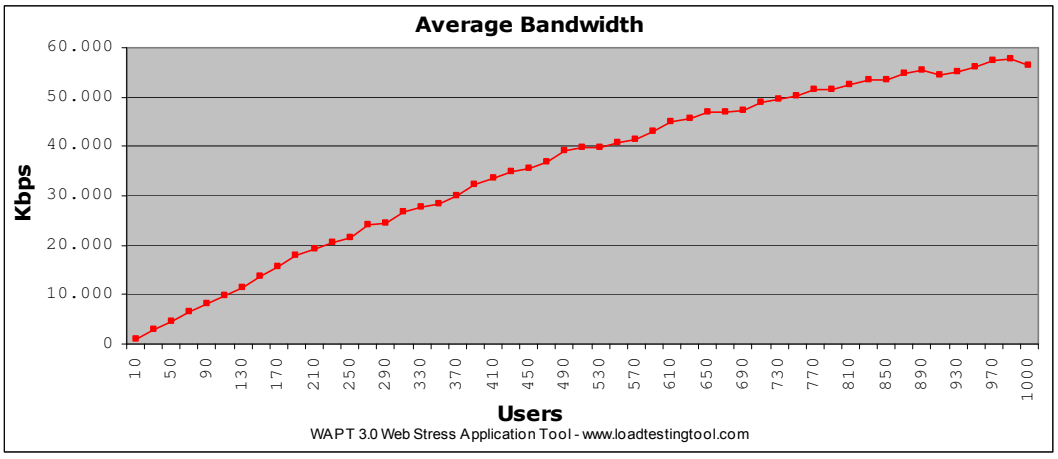

Figura 5.15 - Largura de banda por número de utilizadores (1ª versão)

Conforme se pode verificar, para 10 utilizadores em simultâneo, é necessária uma largura de banda de, aproximadamente, 960Kbps (equivale a 10 utilizadores a visualizarem uma imagem de 12KB por segundo). Esta abordagem é impraticável pois os custos de linhas dedicadas de alto débito são muito elevados.

Para diminuir estes requisitos, o módulo `drivers` disponibiliza um método - `GetDirectImageURL()`, o qual permite que o cliente (*browser*) peça as imagens directamente ao dispositivo. Esta solução apenas funciona com dispositivos que disponibilizam uma interface HTTP, o que felizmente acontece para a maior parte dos dispositivos.

Uma das desvantagens desta abordagem é o facto de, quando as câmaras estão configuradas em portos não normalizados (para pedidos HTTP o porto é o 80) e os utilizadores estão a aceder através de uma *firewall* que bloqueia os acessos a estes portos, não se consegue visualizar as imagens. Como forma de resolver este problema foi inserido um pequeno código no cliente, o qual, após receber três erros no pedido de imagens directamente ao dispositivo, notifica o servidor e este passa a disponibilizar as imagens directamente ao cliente através do porto 80.

Esta nova solução permitiu diminuir drasticamente (para menos de 15%) os requisitos em termos de largura de banda, conforme se pode verificar no gráfico seguinte.

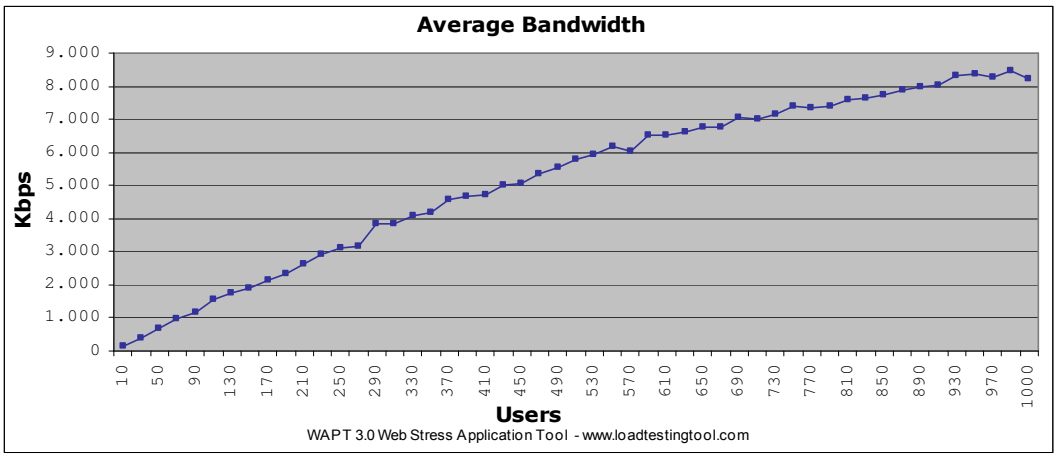

Figura 5.16 - Largura de banda por número de utilizadores (2ª versão)

Convém salientar que, na simulação efectuada, os clientes virtuais não dispõem de *cache* local, funcionalidade disponibilizada em todos os *browsers* comerciais. Num cenário real, caso o acesso à aplicação não seja o primeiro, é muito provável que grande parte das imagens que compõem a interface gráfica já esteja na *cache* do

cliente, diminuindo significativamente o tráfego de informação.

No gráfico da Figura 5.17 estão registados os tempos médios de resposta, sem incluir as imagens transferidas, das principais páginas que compõem a interface da aplicação. Este gráfico permite-nos analisar quais os pedidos que demoram mais tempo a ser processados pelo servidor e a sua relação com o número de utilizadores.

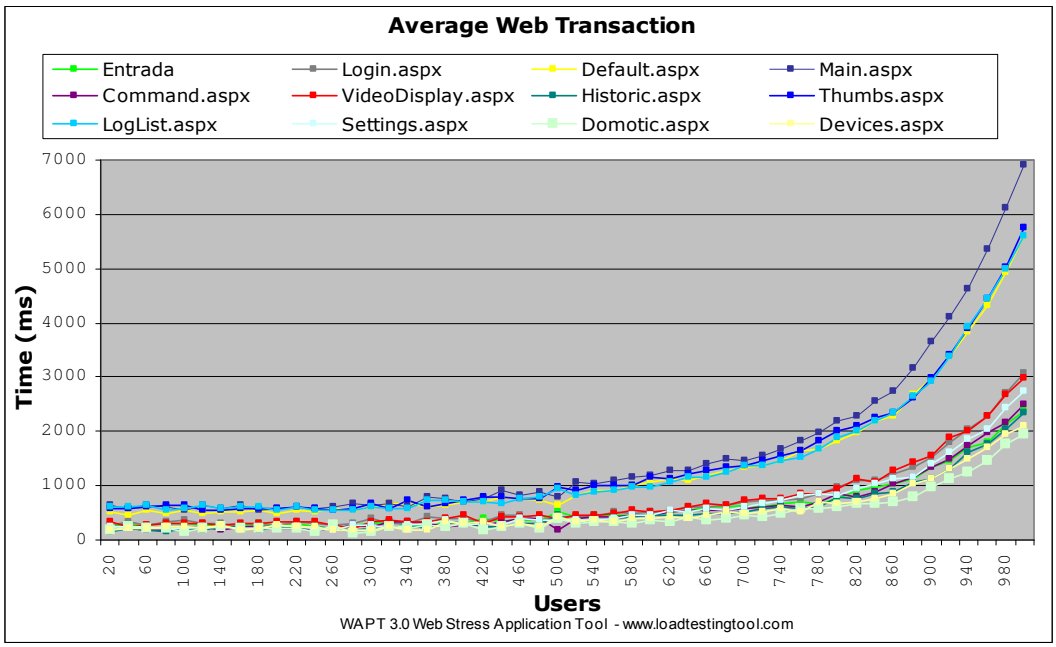

Figura 5.17 - Tempo médio de processamento por página

Estatisticamente, os hábitos de utilização da aplicação, por parte dos utilizadores, não são uniformes para todas as páginas analisadas. De facto, após o utilizador entrar na aplicação, fica uma grande parte do tempo na página de visualização de câmaras e controlo de aparelhos, efectuando comandos para controlo de aparelhos ou controlo da posição da câmara. São poucas as vezes em que ocorrem acessos às páginas de histórico de imagens ou vídeos, sendo ainda mais raros os acessos ao relatório de eventos. As páginas de configurações são também raramente acedidas.

A simulação efectuada simula as interacções de vários utilizadores, admitindo que estes percorrem o mesmo caminho, logo sobrecarregando o servidor efectuando muitos pedidos a páginas que raramente são acedidas.

Para encontrar os limites do servidor em termos de desempenho será usado o gráfico representado na Figura 5.18, o qual indica o número de páginas processadas por número de utilizadores virtuais.

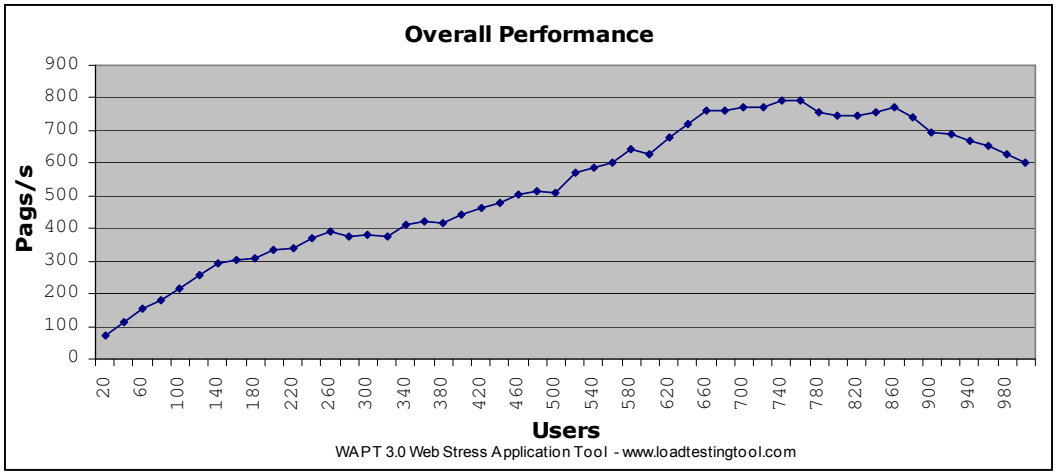

Figura 5.18 - Gráfico de desempenho da aplicação

Ao analisar este gráfico verifica-se que o número de páginas processadas pela aplicação aumenta com o número de utilizadores, até um máximo de 794 páginas/s. Este valor ocorre quando se simula o acesso de 780 utilizadores em simultâneo, significando que, no hardware testado, a aplicação suporta aproximadamente 800 utilizadores em simultâneo, piorando a resposta com a chegada de mais utilizadores.

Podemos constatar que, num universo de 90 clientes, apenas numa única ocasião ocorreu o facto de 14 utilizadores estarem a usar o serviço simultaneamente. Com estes dados poderemos aferir que os picos de utilização terão o valor aproximado de 20% do número de clientes registados.

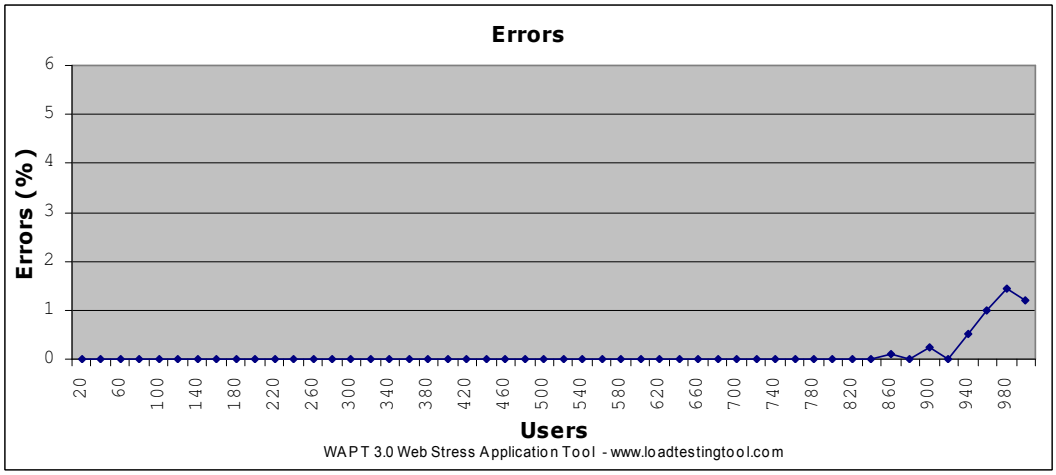

Figura 5.19 - Erros por número de utilizadores

Para finalizar, é apresentado, na Figura 5.19, o gráfico de erros da simulação onde estão representados os erros, em termos percentuais, em função do número de

utilizadores.

Conforme se pode verificar, quando existem na aplicação mais de 900 utilizadores (com a actual configuração de hardware), esta começa a falhar, ainda que numa percentagem muito baixa (máximo de 1,45% para 980 utilizadores). Usando o componente de `debug`, verificou-se que os erros ocorridos devem-se, maioritariamente a erros de comunicação com os dispositivos.

5.9. SÍNTESE

Apesar de não estar totalmente reflectido neste documento, o desenvolvimento não foi totalmente sequencial, seguindo-se uma filosofia iterativa, i.e., após o lançamento de uma versão, melhoravam-se alguns componentes e desenvolviam-se novas funcionalidades. Esta abordagem permitiu corrigir alguns erros iniciais de concepção, os quais foram detectados durante o uso da aplicação em ambientes reais, e descobrir funcionalidades que necessitavam de ser implementadas, sendo de seguida apresentados alguns exemplos.

Uma das primeiras versões da interface forçava a abertura de uma janela do *browser*, após a autenticação do utilizador. Com esta solução pretendia-se controlar a dimensão da janela principal de visualização de forma a optimizar o espaço de visualização. Após a colocação desta versão no servidor de teste, e usando diferentes computadores, descobriu-se que, caso estes tivessem instalado algum software de bloqueio de *popups*, a página principal da aplicação era bloqueada, apresentando-se como um potencial risco para a facilidade de uso.

Um outro exemplo, que dificilmente seria descoberto no planeamento inicial, foi o facto de a interface bloquear enquanto as páginas estão a ser processadas no servidor e não são totalmente enviadas para o cliente. A resolução deste problema passou pela implementação de um sistema *multi-threading*, o qual lança novos processos para efectuar tarefas demoradas, como por exemplo, verificar se um dispositivo está on-line, retornando de imediato uma página de resposta ao cliente. A responsabilidade de voltar a verificar se a operação já foi concluída fica a cargo do cliente, o qual, através de uma *frame* escondida, envia periodicamente ao servidor novos pedidos até que o processamento esteja concluído e a resposta seja enviada para o cliente.

Convém referir novamente que, por uma questão de tempo e de forma a não alongar este documento, foram excluídos alguns pormenores do desenvolvimento.

6. CONCLUSÃO

Para esta dissertação estudaram-se os principais protocolos de domótica actualmente disponíveis no mercado e analisaram-se as principais tecnologias de desenvolvimento de aplicações distribuídas. O objectivo principal era o projecto e desenvolvimento duma aplicação de vigilância e gestão remota dos recursos habitacionais.

Após o estudo dos protocolos de domótica não proprietários, facilmente se concluiu que o melhor posicionado para prevalecer no futuro é o EIB, actualmente abrangido pelo grupo KNX. Durante a fase de planeamento procuraram-se dispositivos EIB/KNX e *gateways* que pudessem ser integradas na aplicação a desenvolver. Esta procura revelou-se fortuita, chegando-se à conclusão que, apesar deste protocolo ser o mais robusto e sólido, a sua aceitação comercial é ainda muito reduzida, talvez devido ao elevado preço. Esta limitação forçou-nos a adoptar o X10, pois este protocolo coloca à nossa disposição variados dispositivos e é de muito mais fácil instalação e uso.

Após uma pesquisa e estudo das principais tecnologias de programação orientadas para ambientes distribuídos, onde foram analisadas as linguagens Java e a recente plataforma .NET, optou-se pelo uso desta última, em grande parte devido ao potencial apresentado, mas também devido ao facto da aplicação que se pretendia desenvolver vir a ser disponibilizada em servidores a executar o sistema operativo e aplicações (*SQL Server* e *IIS*) do mesmo fabricante.

Fazendo uma retrospectiva à escolha da plataforma tecnológica, poder-se-á referir que esta foi uma aposta ganha. Esta plataforma disponibiliza meios para se facilmente criarem aplicações distribuídas e, principalmente, já vem preparada para o desenvolvimento de aplicações web para dispositivos móveis facilitando a satisfação deste requisito inicial.

A aplicação desenvolvida no decorrer desta dissertação foi, desde o início, destinada a ser aplicada num mercado real, sendo lógico que, antes de se definir os seus requisitos funcionais, fosse efectuada uma pesquisa de soluções equivalentes e uma análise dos seus pontos fortes e fracos. Esta pesquisa apresentou-nos algumas soluções bastante interessantes, mas nenhuma que oferecesse a compatibilidade e integração pretendida com diferentes dispositivos.

A análise do mercado a que se destinava a aplicação foi de uma importância significativa para a definição detalhada dos requisitos funcionais da aplicação. Partindo da ideia base, o desenvolvimento de uma aplicação de vigilância e controlo remoto, definiram-se os intervenientes, os actores e detalharam-se as suas interacções com a aplicação. Apesar de se terem definido 4 tipos de actores (utilizador final, assistente técnico, revendedor e administrador), o assistente técnico foi bastante negligenciado durante o desenvolvimento. Este papel e as respectivas funcionalidades serão mais fielmente definidos com o aumento do número de utilizadores finais, sendo esta uma das futuras melhorias a introduzir na actual aplicação.

Para permitir o uso da aplicação é necessário o uso de dispositivos locais que permitam

a captura de imagens, o controlo de aparelhos, etc. A escolha destes dispositivos implicou muitos testes e percalços pois nem todos os fabricantes disponibilizavam as interfaces (*API's*) necessárias para a integração com a aplicação. Esta integração foi, sem dúvida, uma das partes mais difíceis no desenvolvimento. Um outro facto, pouco visível no documento, foi o desenvolvimento de uma interface TCP/IP para a consola de alarme da *Jablotron*, a qual permitiu a sua integração na aplicação e a disponibilização de novas funcionalidades. Esta placa de comunicação foi desenvolvida pela empresa IVV, em parceira com a Universidade do Minho, permitindo-nos definir algumas das funcionalidades, facilitando e potenciando a sua integração nativa com o VisãoWeb.

Durante a fase de testes, com alguns dos clientes da empresa CentralCasa que dispunham do equipamento local necessário, a receptividade destes foi tão elevada que a empresa resolveu apostar comercialmente na aplicação, denominando-a VisãoWeb. A partir desta aplicação foi desenvolvido um plano de negócios, com o qual a empresa conseguiu obter o primeiro prémio do concurso nacional do "Jovem Empreendedor" 2003/2004, promovido pela ANJE – Associação Nacional do Jovem Empresário.

Devido à política de marketing da empresa, a solução inicial foi dirigida para um mercado empresarial. Esta posição afectou ligeiramente o rumo desta dissertação, direccionando-a para o desenvolvimento de funcionalidades adaptadas ao mercado empresarial, como por exemplo, a compatibilidade com vídeo-servidores de gravação local e a visualização em Quad-Cam (esta funcionalidade, apesar de implementada, não foi documentada na presente tese).

Na fase final, com o reconhecimento da importância do mercado residencial e o estabelecimento de parcerias com grandes empresas na área, voltou-se a apostar fortemente em soluções residenciais. Actualmente está em fase final de desenvolvimento a integração com o protocolo EIB abrindo portas à leitura de temperatura, controlo de sistema de aquecimento, entre outras funcionalidades. Estão também em estudo novas interfaces mais orientadas para o controlo e gestão dos recursos habitacionais. Nestas interfaces a organização dos dispositivos é baseada em plantas do imóvel e não nas câmaras de vigilância como nas versões actuais.

Numa perspectiva de evolução da aplicação, e com base na informação obtida a partir da sua utilização diária por um conjunto diversificado de utilizadores, será necessário continuar o trabalho de planeamento e desenvolvimento de novas funcionalidades, destacando-se de seguida as mais procuradas:

- Gravação remota de pequenos vídeos – Pretende-se com esta funcionalidade que um utilizador possa, sem dispor de nenhum equipamento no local com capacidade para armazenar vídeos, gravar vídeos ficando estes armazenados no servidor.

  Esta funcionalidade deve ser independente do tipo de dispositivo, ficando a responsabilidade da codificação do vídeo a cargo do servidor. Esta nova funcionalidade será bastante exigente em termos de largura de banda, devendo ser estudado um mecanismo que permita o uso de linhas económicas alternativas para a obtenção dos vídeos dos dispositivos locais;

- Integração de novos dispositivos – Todos os dispositivos que possuem uma interface de TCP/IP são passíveis de serem integrados na aplicação. Conforme já foi referido, iniciou-se a integração de uma *gateway* de comunicação com redes EIB, mas pretende-se uma integração com outros protocolos e sistemas, como por exemplo, o *HoneyWell* que permite a gestão de sistemas de aquecimento;

- Novas interfaces de visualização – Com as recentes evoluções a nível da televisão digital e interactiva, começam a aparecer *set-top boxes* que permitem o acesso à Internet nas televisões. Este tipo de interface terá de ser bastante simples, já que as *set-top boxes* não têm a mesma capacidade de processamento que um vulgar PC, devendo o design ser também adaptado a televisões onde a resolução e definição são muito menores que num monitor;

- *Streaming* em MPEG4 – Tendo em conta as últimas evoluções tecnológicas a nível de transmissão de dados para dispositivos móveis (3G) e os novos algoritmos de compressão, o desenvolvimento de uma aplicação que permita a recepção de *streams* MPEG4, quer na interface web, quer na interface móvel, irá sem dúvida melhorar significativamente a qualidade de transmissão de vídeo, em especial no número de imagens por segundo, diminuindo drasticamente as exigências a nível de largura de banda.

Outra perspectiva muito interessante de evolução desta aplicação, numa vertente mais tecnológica, prende-se com a utilização do MPEG7 para inserir meta-informação nos vídeos capturados e armazenados pela aplicação. Através do *Multimedia Content Description Interface*, poderemos armazenar desde informação básica, como a data de captura, a duração, o tipo de codificação ou a câmara que capturou o vídeo, assim como informação mais complexa, como a existência de movimento ou som num vídeo. Esta melhoria permitirá uma melhor indexação dos dados do utilizador e facilitará posteriores pesquisas.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[ASP.NET Mobile, 04]Mobile ASP.NET Web Applications – http://www.asp.net/mobile (verificado a 10/03/2005)

[Bernstein, 97] Philip A. Bernstein and Eric Newcomer, **Principles of Transaction Processing**, Janeiro de 1997, Morgan Kaufmann

[CasaFuturo, 04] **Casa do Futuro Interactiva**, http://www.casadofuturointeractiva.com.pt/ (verificado a 10/03/2005)

[CeBus, 05] **CEBus Industry Council, Inc**, http://www.cebus.org (verificado em 10/03/2005)

[CLISpec, 02] Standard ECMA – 335, **Common Language Infrastructure (CLI) 2nd edition**, Dezembro de 2002 http://www.ecma-international.org/publications/standards/Ecma-335.htm (verificado a 10/03/2005)

[Coulouris, 00] George Coulouris, Jean Dollimore, Tim Kindberg, **Distributed Systems - Concepts and Design**, Agosto de 2000, Addison Wesley

[Dietmar, 01] Loy Dietmar, Dietrich Dietmar, **Open Control Networks: Lonworks/Eia 709 Technology**, Outubro de 2001, Kluwer Academic Publishers

[Dutt, 99] Amitava Dutta-Roy, "**Networks for Homes**" in Spectrum, IEEE, Dezembro de 1999

[EIBHandbook, 99] **EIBA Handbook Series – Release 3.0**, Março de 1999

[EIBKit, 00] **EIB Information Kit**, Maio 2000, http://www.eiba.com (verificado a 10/03/2005)

[Evans, 01] Grayson Evans, **CEBus Demystified: The ANSI/EIA 600 User's Guide**, Março de 2001, McGraw-Hill Professional Publishing

[Graziani, 00] Javier Lamas Graziani, **Domotica - Sistemas de Control Para Viviendas**, Janeiro de 2000, Paraninfo

[J2EESpec, 03] JSR Expert Group, **Java 2 Plataform Enterprise Edition Specification v1.4**, 2003

[Jacobson, 00] R. Jacobson, **SQL Server 2000 Analysis Step by Step**, Janeiro de 2000, Microsoft Press International

[Jeronimo, 03] Michael Jeronimo, **UPnP Design by Example: A Software**

Developer's Guide to Universal Plug and Play, Maio de 2003, Intel Press

[konnex, 05] http://www.konnex-knx.com/ (verificado a 10/03/2005)

[Lonworks, 00] **Introduction to the LonWorks System**, Março de 2000, Echelon Corporation

[MacCarty, 99] Bill MacCarty e Luke Cassady-Dorion, **Java Distributed Objects**, 1999

[MacDonald, 03] Matthew MacDonald, **.NET Distributed Applications**, Fevereiro de 2003, Microsoft Press

[McConnell, 03] Steve McConnell, **Professional Software Development**, Junho de 2003, Addison Wesley

[Milroy, 02] Steve Milroy, Ken Cox, **.NET Mobile Web Developer's Guide**, Fevereiro de 2002, Syngress

[MobileControls, 04] ASP.NET Mobile Controls Device Updates - http://msdn.microsoft.com/mobility/othertech/asp.netmc/mobileweb/aspmobiledrivers/default.aspx (verificado a 10/03/2005)

[Monteiro, 00] Edmundo Monteiro e Fernando Boavida, **Engenharia de Redes Informáticas**, Abril de 2000

[MSFT.NET, 03] **Application Architecture for .NET: Designing Applications and Services**, Abril de 2003, Microsoft Press

[Nielsen, 00] Jakob Nielsen's, **Designing Web Usability: The Practice of Simplicity**, Janeiro de 2000, New Riders

[Nielsen, 04] Jakob Nielsen's, http://www.useit.com (verificado a 10/03/2005)

[Raji, 99] Reza Raji, "**The Lonworks Solution and Your Home**" in Home Toys Article, Outubro de 1999

[Rye, 97] Dave Rye, "**X-10 Ltd. Group History and Overview**" in Home Toys Article, June 1997 http://www.hometoys.com/htinews/jun97/articles/x10.htm (verificado a 10/03/2005)

[Sauter, 02] Thilo Sauter, Dietmar Dietrich, Wolfgang Kastner, **EIB: Installation Bus System**, Março de 2002, Wiley-VCH

[Siegel, 00] Jon Siegel, **CORBA 3 Fundamentals and Programming, 2º**

Edition, Abril de 2000, John Wiley & Sons

[Wack, 97] Kenneth Wacks, **"Introduction to the LonTalk Communications Protocol**" in Home Toys Article, Outubro de 1997

[Webb, 99] Warren Webb, "**Consumer Bus Defends Home Turf**" in EDN Access, Agosto de 1999

[Weiss, 02] S.W. Weiss, **Handheld Usability**, Julho de 2002, John Wiley and Sons

[Wu, 99] Jie Wu, **Distributed System Design**, Agosto de 1998, CRC Press

8. ANEXOS

8.1. ANEXO A - EXTRACTO DO FICHEIRO DE LOGS

```
<log>
   <texto>
      Aplicação iniciada. (Texts.xml: D:\Visaoweb_Homes\Texts.xml; logs_aspx.xml:
D:\pontopr\visaoweb\web\logs_web3.xml; UserHomesDirURL: /Homes/; UserHomesDirPath:
D:\Visaoweb_Homes\; DataBaseConnectionString: user id=cca;password=Cartoon;initial
catalog=visaoweb;data source=TAZ;Connect Timeout=30)
   </texto>
   <data>30-03-2005 10:19:34</data>
   <eventType>INFO</eventType>
</log>
<log>
   <texto>
      Erro inesperado!
   </texto>
   <data>30-03-2005 10:19:34</data>
   <eventType>ERROR</eventType>
   <erro>
      System.Web.HttpUnhandledException: Exception of type
System.Web.HttpUnhandledException was thrown. ---> System.IO.FileNotFoundException:
d:\no_picture.JPG
   at System.Drawing.Image.FromFile(String filename, Boolean useEmbeddedColorManagement)
   at System.Drawing.Image.FromFile(String filename)
   at VisaoWebLogic.Core.Device.get_ImgNoPicture() in
logic\VisaoWebLogic\Core\Devices\Device.cs:line 442
   at WebAppCCInterface.CurrentImage.Page_Load(Object sender, EventArgs e) in
d:\debug\web133\Home\CurrentImage.aspx.cs:line 29
   at System.Web.UI.Control.OnLoad(EventArgs e)
   at System.Web.UI.Control.LoadRecursive()
   at System.Web.UI.Page.ProcessRequestMain()
   --- End of inner exception stack trace ---
   at System.Web.UI.Page.HandleError(Exception e)
   at System.Web.UI.Page.ProcessRequestMain()
   at System.Web.UI.Page.ProcessRequest()
   at System.Web.UI.Page.ProcessRequest(HttpContext context)
   at System.Web.CallHandlerExecutionStep.System.Web.HttpApplication
   +IExecutionStep.Execute()
   at System.Web.HttpApplication.ExecuteStep(IExecutionStep step, Boolean&
completedSynchronously)
   </erro>
</log>
<log>
   <texto>
      Actualizei o parametro (DevID: 225 Key: 'IpAddress' Value: '195.23.20.165')
   </texto>
   <data>30-03-2005 11:43:31</data>
   <eventType>SUCCESS</eventType>
</log>
<log>
```

```
   <texto>
      CheckDirsOnDisk (user: HULK\IUSR_visaoweb; dir: D:\Visaoweb_Homes\Home_1)
   </texto>
   <data>30-03-2005 11:44:34</data>
   <eventType>WARNING</eventType>
</log>
<log>
   <texto>
      Falha ao controlar x10 no dispositivo (ID: 92; HouseCode: B; UnitCode: 9)
   </texto>
   <data>30-03-2005 11:45:10</data>
   <eventType>ERROR</eventType>
   <erro>
      System.Exception: Falha ao obter controlar x10 com a Consola da IVV (ip:
192.168.1.50:80)  ---> System.Exception: Erro de timeout na execução!
   at WebComunication.Core.ExecutadorAssincrono.Executa(Int32 timeout) in
logic\WebComunication\Core\ExecutadorAssincrono.cs:line 69
   at VisaoWebLogic.Drivers.ConsoleIVVDriver.ExecutaComandoX10(String strHouse, String
strKey, Int16 function) in logic\VisaoWebLogic\Drivers\IVV\ConsoleIVVDriver.cs:line 241
   --- End of inner exception stack trace ---
   at VisaoWebLogic.Drivers.ConsoleIVVDriver.ExecutaComandoX10(String strHouse, String
strKey, Int16 function) in logic\VisaoWebLogic\Drivers\IVV\ConsoleIVVDriver.cs:line 246
   at VisaoWebLogic.Drivers.ConsoleIVVDriver.SetOff(String strHouse, String strKey) in
logic\VisaoWebLogic\Drivers\IVV\ConsoleIVVDriver.cs:line 221
   at VisaoWebLogic.Core.Device.ControlX10(DeviceModule pX10Dev, X10Command x10) in
logic\VisaoWebLogic\Core\Devices\Device.cs:line 722
   </erro>
   <fich>Device.cs:ControlX10()</fich>
</log>
<log>
   <texto>
      Actualizei o parametro (DevID: 177 Key: 'IpAddress' Value: '195.23.155.118')
   </texto>
   <data>30-03-2005 11:45:18</data>
   <eventType>SUCCESS</eventType>
</log>
<log>
   <texto>
      Falha ao obter imagem do dispositivo (ID: 92)
   </texto>
   <data>30-03-2005 12:10:52</data>
   <eventType>ERROR</eventType>
   <erro>
      System.Exception: Dispositivo não suporta a funcionalidade
   at VisaoWebLogic.Core.Device.CaptureImage() in
logic\VisaoWebLogic\Core\Devices\Device.cs:line 569
   </erro>
   <fich>Device.cs:CaptureImage()</fich>
</log>
<log>
   <texto>
```

```
      Actualizei o parametro (DevID: 201 Key: 'IpAddress' Value: '195.23.20.165')
   </texto>
   <data>30-03-2005 12:13:31</data>
   <eventType>SUCCESS</eventType>
</log>
...
<log>
   <texto>
      Falha ao carregar utilizador (ID: 0)
   </texto>
   <data>31-05-2005 5:54:50</data>
   <eventType>ERROR</eventType>
   <erro>
      System.Data.SqlClient.SqlException: Timeout expired.  The timeout period elapsed
prior to completion of the operation or the server is not responding.
   at System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior cmdBehavior,
RunBehavior runBehavior, Boolean returnStream)
   at System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior)
   at
System.Data.SqlClient.SqlCommand.System.Data.IDbCommand.ExecuteReader(CommandBehavior
behavior)
   at System.Data.Common.DbDataAdapter.FillFromCommand(Object data, Int32 startRecord,
Int32 maxRecords, String srcTable, IDbCommand command, CommandBehavior behavior)
   at System.Data.Common.DbDataAdapter.Fill(DataTable dataTable, IDbCommand command,
CommandBehavior behavior)
   at System.Data.Common.DbDataAdapter.Fill(DataTable dataTable)
   at DataBase.DBService.FillDataTable(String tabela, String filtro) in
logic\DataBase\DBService.cs:line 296
   at VisaoWebLogic.Core.User.Load(Int32 id) in logic\VisaoWebLogic\Core\User.cs:line
756
   </erro>
   <fich>User.cs</fich>
</log>
...
```

8.2. ANEXO B – PARÂMETROS DE EXECUÇÃO DA ANÁLISE DE DESEMPENHO

Test run execution data

Run started at: *Mon Jan 24 10:34:23 2005*

Scenario name: *VisaoWeb.wts*

Test run comment: *VisãoWeb - Hulk - 10-1000 users - 21 Jan*

Test executed by: *André Silva*

Test executed at: *Garfield*

Test run parameters

Virtual users *batch run from 10 to 1000 step 10*

Iterations *1*

Delays between pages: *Yes*

Load level limit: N/A

User connection speed: 256000 bits/sec

Timing mode: *Web transaction (without images)*

Virtual users logs: *are saved to C:\Programas\WAPT\Logs*

Reports: *are saved to C:\Programas\WAPT\Reports*

Main test sequence

Name	Page	Delay
Entrada	www.visaoweb.com/web/default.aspx	0
Login.aspx	www.visaoweb.com/web/login.aspx	0
Main.aspx	www.visaoweb.com/web/Home/Main.aspx	0

Command.aspx	www.visaoweb.com/web/command.aspx?cmd=3	1
VideoDisplay.aspx	www.visaoweb.com/web/Home/videodisplay.aspx	0
Historic.aspx	www.visaoweb.com/web/historics/historic.aspx	1
Thumbs.aspx	www.visaoweb.com/web/historics/thumbs.aspx?cmd=filter&dateInicial=2004-09-01&dateFinal=2005-01-19&camera=-1&ddlEvent=-1	0
LogList.aspx	www.visaoweb.com/web/historics/logList.aspx?cmd=filter&event=0&source=&dayi=04&monthi=11&yeari=2004&dayf=28&monthf=01&yearf=2005	0
Settings.aspx	www.visaoweb.com/web/Settings/settings.aspx	0
Domotic.aspx	www.visaoweb.com/web/Settings/domoticMain.aspx	0
Devices.aspx	www.visaoweb.com/web/Settings/camerasMain.aspx	0