
Universidade de Aveiro
2012
Departamento de Eletrónica, Telecomunicações e Informática

Flávio Martinho Simões Ferreira

Home Gateway do Living Usability Lab

Universidade de Aveiro Departamento de Eletrónica, Telecomunicações e Informática
2012

Flávio Martinho Simões Ferreira

Home Gateway do Living Usability Lab

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Sistemas de Informação, realizada sob a orientação científica do Doutor Joaquim Manuel Henriques de Sousa Pinto, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor António Joaquim da Silva Teixeira, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Apoio financeiro da Fundação para a Ciência e Tecnologia no âmbito do Projeto QREN Living Usability Lab.

UNIÃO EUROPEIA
Fundo Europeu de Desenvolvimento regional

Dedico este trabalho a todos os que sempre me apoiaram, família e amigos e especialmente à pessoa que me tem acompanhado nestes últimos anos.

o júri

presidente

Prof. Dr. Osvaldo Manuel da Rocha Pacheco

Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

arguente

Prof. Dr. José Miguel de Oliveira Monteiro Sales Dias,

Professor Associado Convidado do Departamento de Ciências e Tecnologias de Informação do Instituto Superior de Ciências do Trabalho e da Empresa - Instituto Universitário de Lisboa

orientador

Prof. Dr. António Joaquim da Silva Teixeira

Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

orientador

Prof. Dr. Joaquim Manuel Henriques Sousa Pinto

Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos

Agradeço à minha namorada Carolina e família, em especial ao meu Pai, pelo apoio incondicional.

Agradeço ainda aos meus amigos, colegas de curso, namorada e família que me acompanharam ao longo do meu percurso académico, contribuindo para a conclusão desta fase académica.

Por fim, mas não menos importante, agradeço aos meus orientadores, e aos meus colegas no âmbito do projeto LUL por terem sido sempre compreensivos, cooperadores, disponíveis para ajudar, e por demonstrarem total confiança em mim.

palavras-chave Home Gateway, SOA, LUL, Ambient Assisted Living

resumo

A população sénior tem crescido exponencialmente nos últimos anos, o que leva a que haja uma procura de recursos de prestação de serviços de saúde que é maior do que a oferta. De forma a resolver este problema surgiu o *Ambient Assisted Living* para apoiar seniores no seu quotidiano, dentro das suas habitações. Neste documento é descrito o desenvolvimento do *Home Gateway* para apoio aos serviços *Ambient Assisted Living*, do projeto *Living Usability Lab*, e respetiva arquitetura. O *Home Gateway* é a entidade responsável por gerir todos os recursos oferecidos na habitação do sénior. A arquitetura deste sistema é baseada na arquitetura orientada a serviços, utilizando o OSGi. Os recursos são todos disponibilizados através de serviços *Web* pelo que são descritos alguns dos serviços implementados. O Home Gateway desenvolvido responde eficientemente à necessidade de este ser autónomo, de interoperabilidade, de segurança, de escalabilidade, de conter monitorização, de personalização, de adaptação, de ser embebido e distribuído, de providenciar interação explícita, entre outros. O sistema desenvolvido é passível de ser expandido a outras áreas de saúde e a outros leques etários, para além do apoio à população sénior na própria habitação. Em suma e tendo em conta as conclusões e limitações do trabalho desenvolvido torna-se evidente a possibilidade de trabalhos futuros focados em soluções que podem vir a ser implementadas.

keywords Home Gateway, SOA, LUL, Ambient Assisted Living

abstract

The elder population has been growing exponentially in the last years, which results in a greater search of provision of health care resources comparing to the provision of that services. In order to solve this problem the Ambient Assisted Living emerged to support the elderly in their daily life, within their homes. In this document, it's described the development of the Home Gateway to support the services of Ambient Assisted Living, of Living Usability Lab project and its architecture. The Home Gateway is the entity responsible for managing all the resources available in the elderly's house. The Home Gateway architecture is based on service-oriented architecture using OSGi. The resources are all available through Web services, hence some of the implemented services are described. The developed Home Gateway responds effectively to the needs of being autonomous, of interoperability, security, scalability, monitoring, customization, adaptation, being impregnated and distributed, providing explicit interaction, among others. The system that has been developed may be used in other health areas and with different age ranges in addition to supporting the elderly in their own homes. In brief, and taking into account the conclusions and limitations of the developed work, it becomes clear the possibility of future work on the field focused on solutions that can be implemented.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AAL	Ambient Assisted Living
AmI	Ambient Intelligence
LUL	Living Usability Lab
LULMS	LUL Main Server
HS	Home Site
HPS	Health Professional Site
HG	Home Gateway
HSN	Home Site Network
ES	Especialista de Saúde
SOA	Service Oriented Architecture
NGN	New Generation Networks

1. Introdução

1.1. Motivação

O fenómeno do progressivo envelhecimento da população, não só portuguesa, mas mundial, passou a ser, nas últimas décadas, um fator de atenção e de interesse. No ano 2000, numa escala mundial, havia cerca de 605 milhões de pessoas com mais de sessenta anos, sendo esperado que, no ano 2050 este número aumente quase para o quádruplo, ou seja, para dois mil milhões. Nessa altura, o número de cidadãos seniores irá ser, pela primeira vez na história, superior ao número de crianças com menos de catorze anos [11]. Este aumento é bastante acentuado, em especial, na Europa [12]. Em Portugal a situação é semelhante, verificando-se um aumento significativo de seniores correspondente a aproximadamente 20%, entre 2001 e 2011 [13].

Há várias entidades, nomeadamente os centros de prestação de serviços de saúde que, por falta de profissionais de saúde e de lares de idosos para a quantidade de seniores a que é preciso dar resposta, se preocupam cada vez mais com o crescendo desta mesma população. A utilização de tecnologias da informação e comunicação é considerada uma solução bastante viável e exequível para este tipo de problema [6, 14].

Foi neste contexto que surgiu o *Ambient Assisted Living* (AAL) e, consequentemente, o AAL *Joint Programme*, financiado pela União Europeia [15], que tem como objetivo a melhoria da habitação de seniores recorrendo à integração de tecnologias. Deste modo, pretende-se providenciar apoio a seniores nas suas tarefas diárias com o suporte das tecnologias da informação e comunicação [12, 16, 17].

1.1.1. Enquadramento: O Projeto LUL

O Laboratório Vivo de Utilização de Tecnologias Inovadoras para as Redes de Nova Geração (*Living Usability Lab* - LUL) é um projeto que se iniciou em 2010, e que dá ênfase ao desenvolvimento de tecnologias e serviços que possam contribuir para tornar os cidadãos seniores mais saudáveis, ativos e produtivos [18, 19]. Assim, é importante a existência de especialistas na área da saúde, que possam prestar apoio à distância a esta população. Este projeto tem como parceiros principais o Departamento de Eletrónica, Telecomunicações e Informática, o Instituto de Engenharia Eletrónica e Telemática de Aveiro, pertencente à Universidade de Aveiro, a Faculdade de Engenharia da Universidade do Porto e o INESC - Porto, a Microsoft, as empresas Micro I/O e Plux, e é a base do recém-iniciado projeto que lhe dá continuidade: QREN AAL4ALL [18, 20].

O objetivo essencial do LUL é fornecer condições para a criação de serviços e aplicações inovadoras em ALL, utilizando a arquitetura descrita na secção seguinte e, ao mesmo tempo, ter

um conjunto de serviços ao dispor que possam ajudar a criar modelos de negócio[1]. Uma vez que o cenário que se pretende implementar tem especificidades de comunicação e de diversidade de equipamentos (sensores, câmaras de vídeo, atuadores, *set-top boxes*, entre outros) que funcionam de forma desarticulada, torna-se necessário resolver este obstáculo. Para além disso, é crucial permitir gestão remota, e tornar o sistema adequado aos utilizadores.

É crucial que, neste projeto, se considere a usabilidade, a interação multimodal, e a utilização de redes da próxima geração (*Next Generation Networks* - NGN) [1, 19], que assentam numa arquitetura de serviços e formas simples, de baixo custo e automáticas, que interligam um vasto número de casas [19].

1.2. Objetivos do trabalho

Este trabalho tem como principal objetivo desenvolver um HG para apoio a serviços AAL, criando condições e infraestrutura para o desenvolvimento, refinamento e testes de novos serviços.

Os objetivos específicos englobam (1) a análise de requisitos e de soluções no âmbito do desenvolvimento de sistemas AAL, mais especificamente HG, (2) a demonstração do HG no que respeita à segurança da população sénior que vive em habitações pessoais, (3) o desenvolvimento de alguns serviços de apoio às restantes componentes do projeto, e (4) o teste da arquitetura do HG.

O HG deverá cumprir alguns requisitos primordiais, devendo ter:

- Capacidades de instalação/configuração automatizadas;
- Manutenção remota;
- Mecanismos que apoiem o desenvolvimento de serviços e respetiva manutenção;
- A possibilidade de parar, inicializar, reiniciar, atualizar ou eliminar qualquer componente sem que isso obrigue à reinicialização do servidor;
- Uma arquitetura que possibilite a fusão, numa fase posterior, com os equipamentos de operadores de redes de nova geração;
- A propriedade de apoiar a transmissão de vídeo de qualidade e resolução elevadas;
- A capacidade de proporcionar o controlo de todos os dispositivos da habitação, agindo como controlador central da mesma.

1.3. Estrutura da dissertação

No capítulo 2 são apresentados alguns dos projetos mais relevantes para a dissertação de AAL e algumas arquiteturas já implementadas. Neste capítulo que será introduzido o conceito de AAL que está intimamente relacionado ao restante trabalho. São apresentadas algumas das arquiteturas

existentes com respetivas vantagens e desvantagens. Por fim, são apresentadas as tecnologias utilizadas para o desenvolvimento do HG.

No capítulo 3 são apresentados os requisitos obtidos para o desenvolvimento do HG e é descrita a arquitetura do HG com a explicitação das vantagens de cada módulo. Para além disso, pode encontrar-se uma descrição dos serviços desenvolvidos ainda nesse capítulo.

No capítulo 4 são apresentados os resultados obtidos e é desenvolvida a respetiva análise, assim como é apresentado o ambiente ideal de testes do projeto LUL.

Por fim, no capítulo 5, é exposta uma análise genérica ao sistema final e consideram-se algumas limitações do trabalho desenvolvido. É ainda referida um pouco da expansibilidade do HG para outros contextos e do trabalho futuro que ainda pode ser desenvolvido.

1.4. Publicações

No âmbito do projeto LUL e do trabalho desenvolvido foram publicados ou estão em fase de publicação os seguintes artigos:

- *The Living Usability Lab Architecture: Support for the Development and Evaluation of New AAL Services for the Elderly* [1] – capítulo de livro aceite mas ainda não publicado;
- *Health@Home Scenario: Creating a New Support System for Home Telerehabilitation* apresentado na *International Living Usability Lab Workshop on AAL Latest Solutions* [10] – artigo publicado;
- *Cloud Computing Enhanced Service Development Architecture for the Living Usability Lab [2]* – artigo publicado;
- *Cloud Computing Enhanced Service Development Architecture for the Living Usability Lab [21]* – capítulo de livro aceite mas ainda não publicado.

2. Estado da Arte / Trabalho relacionado

2.1. O que é o Ambient Assisted Living (AAL)?

*"*AAL aims to prolongate the time people can live in a decent way in their own home by increasing their autonomy and self-confidence, the discharge of monotonously everyday activities, to monitor and care for the elderly or ill person, to enhance the security and to save resources*"* [12].

Um dos focos do AAL assenta na ideia de aumentar a qualidade de vida, autonomia e independência da população sénior, permitindo que esta usufrua da habitação própria, mesmo que sofra de disfunções incapacitantes ligeiras. Desta forma, o AAL funciona como auxiliar das capacidades funcionais dos seniores, promovendo um padrão de vida mais saudável e controlado, podendo, ainda, ajudar a combater o isolamento social, disponibilizando uma rede social à população-alvo. O AAL deve ser visto como um contributo não só para a comunidade sénior, mas também para os prestadores de cuidados informais (família, amigos,…) e formais (médicos, enfermeiros,...), uma vez que pode aliviar a sobrecarga inerente aos prestadores de cuidados informais e manter os prestadores de cuidados formais mais e melhor informados, aumentando a sua eficiência e facilitando as suas obrigações como profissionais. De uma forma abrangente, a utilização do AAL permite que se reduzam custos no que concerne à saúde. Para além disso, é bastante aliciante encarar o AAL como provedor de novos modelos de negócio que visem a cooperação de várias partes (fabricantes de dispositivos, *developers*, prestadores de serviço, entre outros), de plataformas comuns (*software* e *hardware*) que respondam às expectativas e permitam um fácil desenvolvimento e implementação de soluções AAL [6].

O AAL como prestador de apoio pode ser dividido por vários domínios: envelhecimento saudável em casa, envelhecimento saudável em movimento, envelhecimento saudável na comunidade, e envelhecimento saudável no trabalho [6].

Num estudo preliminar sobre o AAL que teve em conta a opinião da população sénior acerca do mesmo, verificou-se que, de acordo com essa população, o desenvolvimento do projeto deveria considerar a existência de plataformas acessíveis de comunicação e socialização, alarmes para a possibilidade de ocorrência de situações perigosas, lembretes automáticos, e monitorização do estado clínico, entre outras soluções que sirvam de apoio aos seniores no seu quotidiano [6].

Atualmente há evidências e consenso no facto de as tecnologias de um sistema AAL deverem ser [6]:

- Embebidas, isto é, não invasivas e invisíveis;
- Distribuídas uniformemente pelo ambiente;
- Personalizáveis de acordo com as necessidades dos utilizadores;
- Adaptáveis, ou seja, ajustáveis ao contexto, ao ambiente e ao utilizador;

- Antecipatórias, ou seja, devem adiantar-se aos desejos do utilizador o mais rapidamente possível.

Os grandes desafios associados a sistemas AAL prendem-se, essencialmente, com a heterogeneidade (grande e variada quantidade de tecnologias diferentes para o mesmo fim), a interoperabilidade (capacidade de dois programas serem capazes de trocar e interpretar dados, o que pressupõe um mecanismo que é utilizado para superar a heterogeneidade), a estandardização (pretende facilitar os sistemas de interoperabilidade para vários domínios), a segurança e privacidade (necessários devido ao facto de o AAL lidar com dados pessoais sensíveis), os aspetos legais (as aplicações AAL apresentam desafios legais relacionados com a utilização posterior dos dados e proteção do consumidor), a confiança (capacidade de prever e corrigir erros), a *context awareness*, a usabilidade (capacidade de utilizar eficientemente um produto ou um serviço), e a aceitação (obstáculo que pode ser difícil de ultrapassar, uma vez que, de uma maneira geral, é difícil que os utilizadores aceitem as novas tecnologias, especialmente os utilizadores seniores) [6].

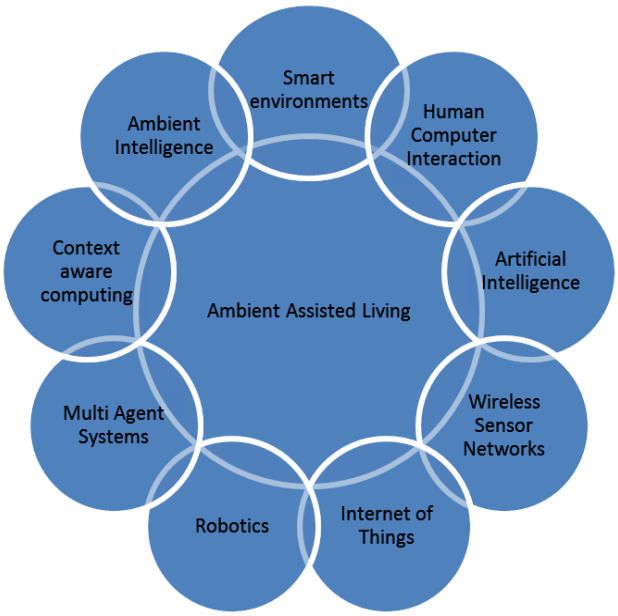


Figura 1 – Áreas de investigação relacionadas com AAL – Retirado de [6]

A Figura 1 apresenta as áreas de investigação relacionadas com AAL, que devem ser compreendidas a favor do desenvolvimento do HG [6].

As duas áreas de investigação mais relevantes para o desenvolvimento deste sistema e, por isso, para a elaboração deste trabalho são a área da *Ambient Intellince* (AmI) e *Internet of Things* (IoT). A primeira porque engloba todos os princípios de automação que são utilizados em AAL. E a segunda porque todos os dispositivos da rede são tratados como recursos tal como acontece no HS.

A *AmI* é definida como um ambiente digital que, de forma pró-ativa e, ao mesmo tempo, sensível, apoia a vida diária das pessoas na sua habitação [6]. A ideia principal de AmI é que o ambiente se deve adaptar às necessidades do habitante e não o contrário [6]. O *Information Society Technologies Advisory Group* (ISTAG)[1] [22] define cinco requisitos tecnológicos essenciais para esta área de investigação:

- O *hardware* deve estar devidamente embutido e discreto, integrando-se de forma total na habitação;
- Deve existir uma boa infraestrutura de comunicação móvel/fixa;
- A rede de dispositivos deve ser dinâmica e massivamente distribuída;
- As interfaces de utilizador devem utilizar a multimodalidade, e devem ser adaptáveis e o mais naturais possível;
- Os sistemas devem providenciar confiança e segurança.

A AmI não deve ser confundida com outras áreas de investigação como o IoT, criticado por alguns por travar a evolução da AmI [6]. No entanto, esta área de investigação pode ser um meio de se alcançarem objetivos na área da AmI [23].

A área de investigação da IoT está muito relacionada com a *Internet*, com as comunicações móveis e com as redes *wireless* de sensores. Esta área de investigação pode ser representada como uma rede baseada nos tradicionais portadores de informação, incluindo a *Internet*, que liga objetos físicos normais, com endereços identificáveis, de forma a providenciar serviços inteligentes [24]. A IoT é caracterizada por [24]:

- Tornar acessiveis todos os objetos, incluindo objetos como copos, rodas de carros, entre outros;
- Providenciar uma interligação de todos os objetos físicos como terminais de rede
- Os serviços universais serem inteligentes.

Como se pode verificar pela sua definição, o AAL está muito relacionado com várias áreas de investigação, pelo que se pode tornar oportuno analisar soluções dessas áreas, de forma a criar uma solução mais eficaz e adequada de um HG para AAL.

Nas secções que se seguem serão apresentados alguns dos projetos mais relevantes de AAL, dando ênfase aos que lidam com o HS.

2.2. Projetos para AAL mais relevantes

Nesta secção são apresentados alguns dos projetos mais relevantes na área de AAL, dando-se principal enfase aos que utilizam HG e/ou que focam a sua implementação na habitação.

[1] Comissão independente preocupada com a criação de conhecimento na área de AmI.

A próxima secção deste trabalho dá ênfase ao projeto *Enhanced Complete Ambient Assisted Living Experiment* (eCAALYX) assim como à sua arquitetura por ser um projeto amadurecido. Seguidamente apresentam-se os projetos de *Middleware Platform for eMPOWERing cognitive disabled and elderly* (MPower) e Persona, bastante relevantes na área de AAL. A apresentação do projeto Persona obriga a uma breve apresentação do projeto Amigo e do projeto *Self-Organizing Dataflow Architectures suPporting Ontology-based problem decomPosition* (SODAPOP) uma vez que a sua arquitetura é baseada nesses projetos. Por fim, apresenta-se o projeto *Ambient Middleware for Context-Awareness* (Amica). Por fim são apresentados outros projetos para AAL.

2.2.1. Projeto eCAALYX

Em [5] é apresentado o projeto *Enhanced Complete Ambient Assisted Living Experiment* (eCAALYX), terminado em 2012 e baseado nos pontos fortes do projeto *Complete Ambient Assisted Living Experiment* (CAALYX). Os objetivos deste projeto foram:

- Monitorizar seniores com doenças crónicas em casa e em movimento
- Aumentar a qualidade de vida da população sénior, dando-lhe liberdade e segurança
- Prevenir a deteorização do estado dos doentes, providenciando-lhes apoio, orientação e cuidados de saúde através da televisão.

Segundo [25] o projeto é composto por três elementos principais: *Home Subsystem*, *Mobile Subsystem* e *Caretaker Subsystem* (Figura 2).

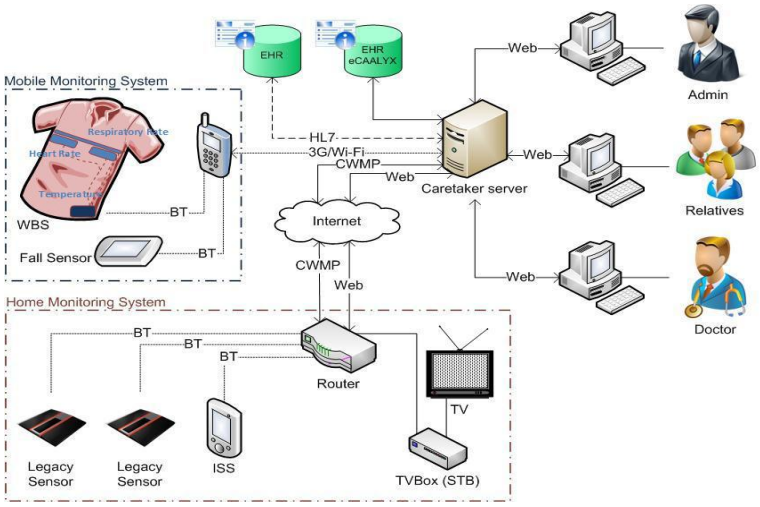


Figura 2 – Visão geral do projeto eCAALYX – Retirado de [5]

Em [26] é apresentado o *Home Subsystem* e está descrita e explicada a implementação do HG. Segundo os autores, esta solução resolve problemas de interoperabilidade e implementa gestão remota de forma a aumentar a flexibilidade, usabilidade e confiança do sistema.

Este subsistema é composto por três elementos principais: o *Caretaker Manager*, que é responsável pela comunicação com os *Caretaker Servers* e por pré-processar os dados médicos obtidos pelos sensores; o *Sensor Manager*, que tem como função a comunicação com os sensores, a coordenação das operações e o relacionamento das características técnicas dos sensores com a lista de sinais vitais a serem monitorizados; e o *CPE Manager*, que lida com as capacidades de gestão remota e que é responsável por monitorizar todos os elementos do sistema e reportar falhas e/ou configurações erradas (Figura 3).

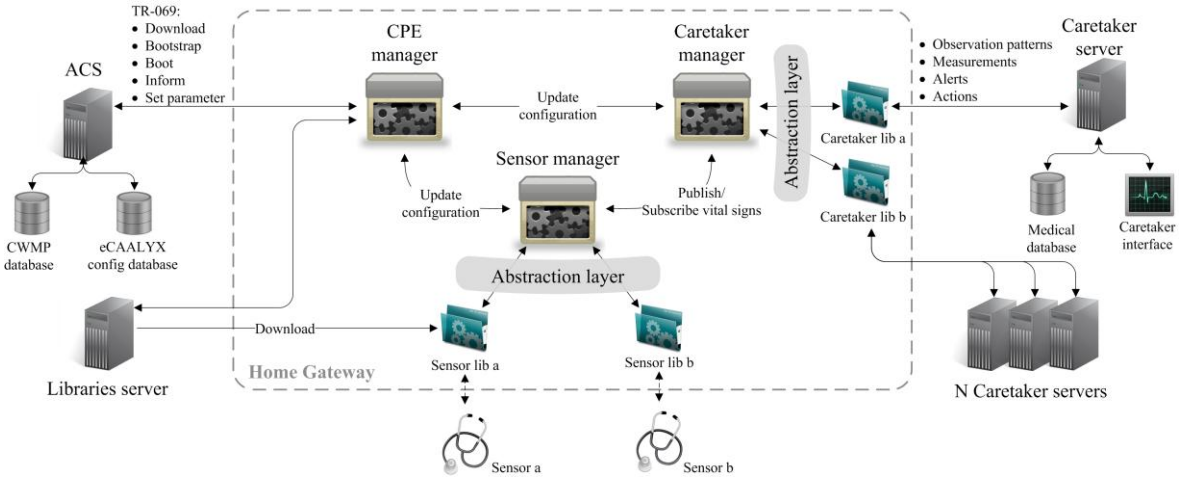


Figura 3 – Arquitetura HG do projeto eCAALYX – Retirado de [26]

2.2.2. Projeto MPower

O projeto *Middleware Platform for eMPOWERing cognitive disabled and elderly* (MPower) é um projeto AAL criado no sentido de desenvolver uma plataforma *middleware* para suporte ao desenvolvimento e distribuição rápida de serviços de cuidados de saúde [27-32]. As principais funcionalidades do sistema são [27-32]:

- A troca de informação entre a habitação;
- O centro prestador de serviços de saúde:
- A monitorização da habitação por parte do centro prestador de serviços de saúde:
- Configuração flexível das características do sistema.

De forma a obter uma solução que responda a todas as necessidades, identificaram-se, numa primeira fase, os requisitos e respetivos atores [27, 29] e uma solução de interoperabilidade [32]. Por fim, criou-se uma arquitetura que respondesse a todos os requisitos identificados [30, 31].

A plataforma desenvolvida é baseada em *Service Oriented Arquitecture* (SOA)[2] e *OMG's Model Driven Architecture* (MDA)[3] [28, 30].

[2] A definição de SOA encontra-se explicitada na secção 2.4.4.
[3] Metodologia de desenvolvimento de *softwate* baseada na importância da modelagem do mesmo.

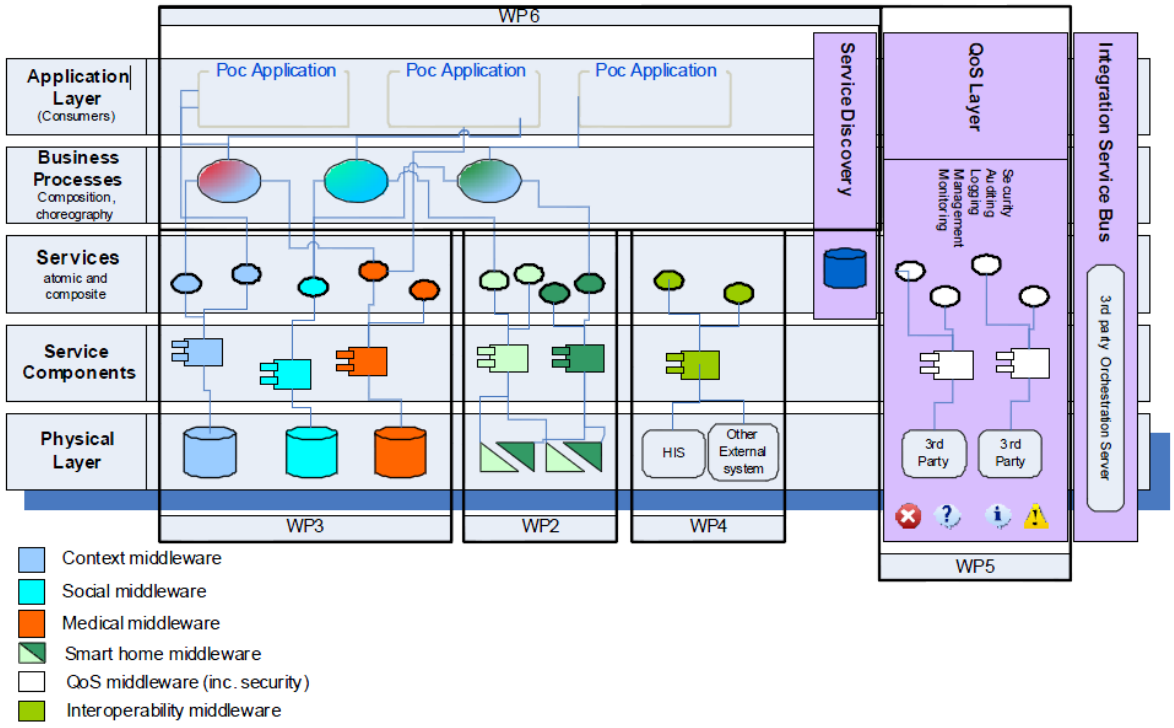


Figura 4 – Arquitetura do projeto MPower – Retirado de [31]

A arquitetura do MPower é baseada na arquitetura proposta pela IBM [33]. Ao analisar a Figura 4 pode verificar-se que a arquitetura está dividia em cinco camadas [30, 31]. A camada física (*Physical*) contém as aplicações criadas como, por exemplo, a base de dados (informação administrativa do paciente, informação de medicação, entre outros) [30, 31]. A camada de componentes de serviços (*Service Components*) expõe as funcionalidades dos componentes físicos e da base de dados da camada física. A camada de componentes de serviços providencia acesso de alto nível à informação e ao controlo dos recursos físicos [30, 31]. A camada de serviços (*Services*) contém as implementações dos serviços desenvolvidos na plataforma MPower. Cada serviço é descrito pelo seu interface e está disponível para desenvolvimento de aplicações. As descrições dos serviços estão guardadas num repositório e podem ser descobertas pelo *Service Discovery* [30, 31]. A camada dos processos de negócio (*Business Processes*) define as regras de negócio das aplicações se são criadas utilizando o MPower *middleware*. Cada processo de negócio define um conjunto de serviços que podem ser orquestrados e coreografados de forma a atuarem de forma independente (por exemplo, a gestão do calendário partilhado, onde este e os seus componentes podem ser utilizados como serviços) [30, 31]. A camada de interface providencia interfaces gráficas de utilizador para os serviços e para os processos de negócio [30, 31].

O *middleware* do MPower está dividido em quatro tipos de serviços [30, 31]:

- Serviços de contexto que estão relacionados com a monitorização e gestão do contexto do paciente [30, 31];
- Serviços de informação (médica e social) que permitem gerir a informação social e os eventos do paciente [30, 31];
- Serviços de interoperabilidade que fornecem a capacidade de interoperabilidade do sistema [32];
- Serviços de qualidade de serviços que providenciam, entre outros, funcionalidades de gestão de serviços, utilizadores, privilégios e segurança (utilização de *Universal Description Discovery and Integration* – UDDI[4]) [30, 31].

2.2.3. Projeto Persona

O Persona é um projeto que tem como base os projetos Amigo e SODAPOP, tornando-se, por isso, necessário descrever, primeiramente, estes dois últimos projetos [8].

O projeto Amigo teve dois objetivos fundamentais que se destinaram a investigar e desenvolver um *middleware* aberto, estandardizado e interoperável, e serviços inteligentes, para ambientes inteligentes, que ofereça aos utilizadores uma interação intuitiva, personalizada e fácil, fornecendo interoperabilidade entre serviços e aplicações [34]. Para cumprir estas metas, foi necessário considerar etapas mais específicas mas bastante importantes [34]:

- Investigar e desenvolver uma plataforma aberta e interoperável capaz de escalar à medida das necessidades do utilizador;
- Investigar e desenvolver serviços inteligentes que combinem interação do utilizador, preferências de utilizador, e *context awareness*;
- Garantir interoperabilidade entre dispositivos, serviços e toda a heterogeneidade da rede;
- Garantir configuração dinâmica e automática;
- Avaliar as soluções utilizando protótipos para obter *feedback* dos utilizadores.

O projeto Amigo desenvolveu um *middleware* que integra dinamicamente sistemas heterogéneos de forma a promover a interoperabilidade entre serviços e dispositivos. A plataforma do Amigo segue o paradigma SOA que permite desenvolver *software* como serviço de forma desacoplada.

A arquitetura do Amigo é constituída por três elementos (Figura 5)[35]:

- A camada base (*Base Middleware*) que é uma solução flexível de *middleware* que pode integrar a maioria das tecnologias mais importantes (plataformas de serviços, protocolos, paradigmas de programação). A comunicação e a descoberta de serviços e dispositivos são

[4] A definição de UDDI encontra-se explicitada na secção 2.6.1.3.

baseadas em semânticas, incluindo as que são baseadas nos padrões de comunicação e descoberta (UPnP, WS ou SLP) [35, 36];

- A camada de serviços inteligentes (*Intelligent User Services*) permite interoperabilidade e integra plataformas heterogéneas de serviços. Esta providencia a utilização de semânticas para representar características funcionais, não funcionais ou de arquitetura, que permitem raciocínio automático sobre os conceitos representados. Os mecanismos de segurança são igualmente providenciados por esta camada [35];
- A outra camada (*Programming & deployment framework*) ajuda as restantes camadas a lidarem com as semânticas, com a gestão da casa, com a descoberta e com a integração de serviços .NET e OSGi [35, 37].

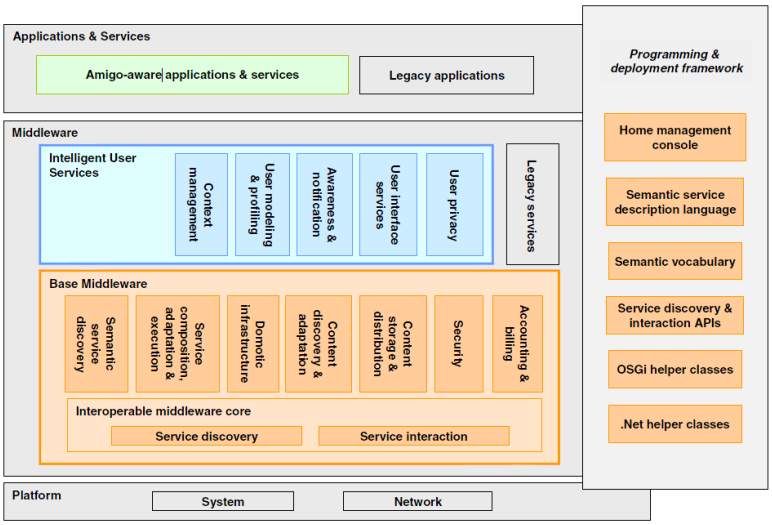


Figura 5 – Arquitetura do projeto Amigo – Retirado de [35]

O sistema SODAPOP (Self-Organizing Dataflow Architectures suPporting Ontology-based problem decomPosition) é um modelo desenvolvido para suportar processamento de fluxo de dados, suportar chamadas remotas de procedimentos, suportar auto-organização dos componentes do sistema, suportar decomposição de problemas descentralizados e resolução conflitos, suportar extensão dinâmica para novos componentes e suportar unificação de sistemas completos. Este sistema introduz dois conceitos fundamentais: (1) canais ou *buses* que leem mensagens ao longo do tempo (podem receber qualquer tipo de mensagem) e que providenciam distribuição espacial de um evento para múltiplos tradutores e (2) tradutores, que podem ler uma ou mais mensagens ao longo do tempo criando uma (ou mais) resposta(s). Os tradutores têm de ter memória, não estão distribuídos e não aceitam qualquer mensagem [38].

O projeto Persona foi criado para ambientes inteligentes através da harmonização das tecnologias e conceitos AAL de modo a desenvolver soluções para providenciar independência a seniores. O

principal objetivo deste projeto é desenvolver uma plataforma estandardizada e escalável que providencia uma variedade de serviços AAL. Os autores subdividiram estes serviços em quatro categorias [8]:

- Serviços de apoio à inclusão social e de troca de experiências;
- Serviços de apoio à comunidade sénior nas suas atividades diárias;
- Serviços de apoio ao sénior, que aumentam a sua confiança e segurança e que os ajudam em situações de risco;
- Serviços que fomentam a mobilidade e suportam os idosos fora das suas habitações.

Os requisitos deste projeto estão divididos em dois grupos: requisitos de utilizador e requisitos técnicos [8].

Os principais requisitos de utilizador que são apontados são os seguintes [8]:

- Os novos serviços têm de ser tão eficazes como a maneira usual de lidar com os problemas;
- O utilizador deve poder desconectar os serviços sempre que quiser;
- O sistema não deve substituir totalmente o tratamento pessoal no que diz respeito a amigos, familiares e vizinhos;
- O sistema deve considerar sempre o contexto do utilizador para evitar inconsistências e assegurar uma prestação de serviços coerente;
- Deve ser possível adicionar, facilmente e sem necessidade de muita configuração, novas funcionalidades, serviços e *hardware* a um determinado espaço físico;
- Os dispositivos da infraestrutura devem ser invisíveis de tal forma que o utilizador praticamente não se aperceba dos mesmos;
- As decisões automáticas do sistema devem pedir confirmação do utilizador e devem ser registadas para que o utilizador as possa verificar;
- O sistema deve ser fácil de utilizar;
- Os interfaces de utilizador devem adaptar-se às necessidades do utilizador à medida que as capacidades se alteram com a idade;
- A comunicação com o utilizador deve ser feita de várias formas (voz, texto, gráficos, sinais, etc.);
- Os dados obtidos têm de ser relevantes para os serviços;
- O utilizador deve poder desativar a recolha de dados dos dispositivos; (13) o sistema deve ser capaz de autenticar o utilizador;
- A transmissão de dados para terceiros deve ser minimizada e segura cumprindo as normas legais;
- O sistema deve verificar se os dados podem ser fornecidos a terceiros.

Já os requisitos técnicos são [8]:

- Garantir alto nível de flexibilidade na distribuição de funcionalidades e facilitar a integração de um número arbitrário de dispositivos;
- Permitir que os componentes atuem imediatamente como nós da infraestrutura;
- Suportar diferentes padrões de comunicação;
- Providenciar mecanismos de descoberta e ligação de serviços;
- Suportar encadeamento de serviços;
- Providenciar mecanismos para agregação de eventos;
- Suportar processamento paralelo;
- Providenciar composição e orquestração de serviços;
- Facilitar a interação explícita entre o utilizador e o sistema e suportar multimodalidade;
- Suportar personalização e consciência de contexto em todas as camadas do sistema e suportar adaptabilidade na camada de apresentação;
- Esconder a complexidade dos serviços aos seus utilizadores;
- Providenciar gestão de qualidade de identidades e privacidade;
- O sistema deve ser modular de forma a suportar extensibilidade e distribuição;
- O sistema deve ser estável, deve avaliar a eficiência e performance, e deve ser simples e completo.

A arquitetura do sistema desenvolvido é baseada na arquitetura *bus* do projeto SODAPOP, é orientada a serviços e utiliza a abordagem ontológica do projeto Amigo. Para além disso, a opção neste projeto foi pela utilização de OSGi como plataforma de serviços [8].

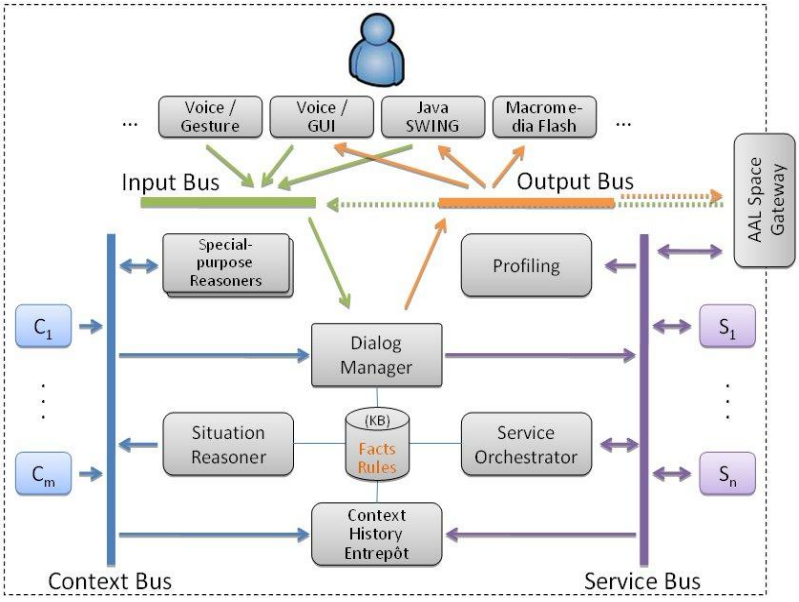


Figura 6 – Arquitetura do projeto Persona – Retirado de [8]

Como se pode verificar pela Figura 6, a arquitetura do projeto Persona é constituída pelos componentes [8]:

- *Dialog Manager* - componente independente da aplicação que lida com os diálogos de todo o sistema e esconde a complexidade de utilizar os serviços das aplicações do utilizador (por exemplo, apresentação hierárquica de serviços disponíveis ao utilizador, suporte a pesquisa de serviços, lidar com logins e etc.) [8];
- *Context History Entrepôt* (CHE) - recolhe a história de todos os eventos do contexto no repositório central, não só para preencher a falha de alguns editores de contexto não providenciarem um interface de pesquisa, mas também para providenciar uma solução de reserva para aqueles os que não conseguem manter todos os dados recolhidos por eles. Garante, ainda, o suporte essencial para se realizar a análise estatística ao longo do tempo e tem uma política que vai eliminando alguns dos dados, para evitar um crescimento exagerado deste repositório.
- *Situation Reasoner* - raciocinador de contexto que utiliza o repositório CHE para inferir novas informações de contexto utilizando o poder lógico do RDF[5]. O *Special-purpose Reasoner*, que é também um componente do projeto Persona tem um papel semelhante ao *Situation Reasoner*, pelo que não é aqui salientado [8];
- *Service Orchestrator* (SO) - é responsável por interpretar os meta-dados[40] que descrevem um serviço, e por realizar instruções com os mesmos. O SO regista os serviços no *bus* tal como cada *service callee* o faria para os seus serviços atómicos. Assim, sempre que um serviço é chamado no *bus*, o único objeto que o implementa será o SO. Posteriormente, o SO executa o serviço correspondente, chamando os seus subserviços, até terminar, devolvendo os resultados para o *bus* [8];
- *Profilling* - garante a adaptabilidade do espaço AAL para os desejos e necessidades do utilizador [8].

Para além dos componentes representados na Figura 6, existe ainda o *Privacy-aware Identity & Security Manager* (PISM) que atua em conjunto com o *middleware* para controlar o acesso aos serviços. Este componente é, também, o provedor de serviços. As principais responsabilidades do PISM são [8]:

- A gestão de identidades e credenciais;
- A gestão de permissões de acesso aos serviços;
- O fornecimento de serviços de autenticação;

[5] Modelo de dados padrão para troca de informação na WEB, baseado em tripletos [39] W. W. W. C. (W3C). (2012). *Resource Description Framework*. Available: http://www.w3.org/RDF/.

- O provisionamento de um mecanismo que controla a divulgação de dados privados.

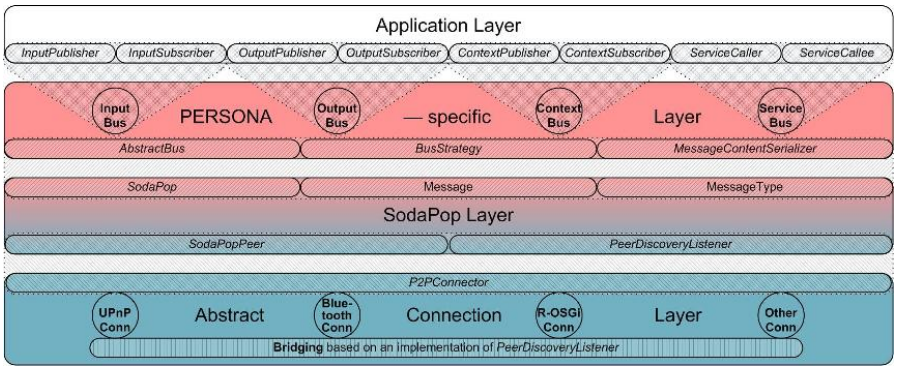


Figura 7 – Camadas do projeto Persona – Retirado de [8]

A arquitetura do Persona é composta por três camadas lógicas (Figura 7). A camada inferior (*Connection layer*) é responsável pela comunicação ponto-a-ponto entre as instâncias do *middleware* (estes utilizam protocolos de descoberta). A camada intermédia (*SodaPop layer)* implementa os interfaces da camada anterior e regista todos os conectores encontrados. A camada superior (*Persona-specific layer*) implementa a entrada, saída, e *service buses* com as suas estratégias distribuídas, utilizando um interpretador de RDF para troca de mensagens [8].

2.2.4. Projeto AMiCa

O *Ambient Middleware for Context-Awareness* (AMiCA), de 2012, é um ambiente de *midleware* em que a arquitetura se baseia em múltiplas camadas, sendo que tem como objetivo fornecer, de forma perfeita e oportuna, a conectividade necessária aos serviços de ambientes dinâmicos como os de serviços AAL [7, 41].

Para além de responder aos requisitos típicos como a heterogeneidade, a escalabilidade, a mobilidade, a comunicação na rede e a segurança, o AMiCA pretende suportar [7]:

- Ambiente inteligente;
- *Context awareness*;
- Descoberta dinâmica dos recursos disponíveis;
- Transparência;
- Comunicação assíncrona;
- Reutilização de componentes;
- Adaptabilidade de ambientes e dispositivos;
- Desenho leve;
- Utilização de *Cloud Computing*.

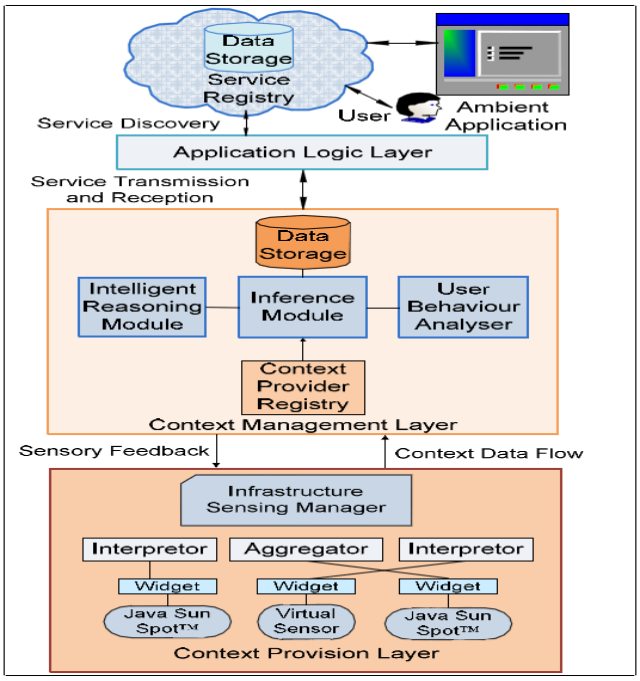


Figura 8 – Visão de alto nível da arquitetura do AMiCA – retirado de [7]

A Figura 8 representa as principais camadas do projeto AMiCA. A camada *Context Provision* da arquitetura tem por base a utilização de sensores *Java Sun SPOT* - e outras fontes de contexto relevantes que permitem a integração dinâmica das fontes de contexto - e deve estar devidamente separada das outras camadas. O *aggregator* combina os dados de contexto obtidos, que são transformados num formato compreendido pelo *interpretor*. A camada *Context Management* guarda os dados detalhados de todas as fontes de contexto que serão posteriormente disponibilizados durante o processo de descoberta. Cada nível da camada de aplicação tem acesso ao seu armazém de dados à medida do tempo, sendo que é nesta camada que se encontra implementada a lógica da aplicação de *context awareness*. É igualmente a camada de aplicação que irá utilizar serviços adicionais que poderão estar na *cloud*. Baseadas nos dados obtidos e tratados nas camadas inferiores, as aplicações da camada superior podem ajustar os seus comportamentos e adaptar os seus serviços para o utilizador. O registo e descoberta de serviços fazem, também, parte da camada de apresentação desta arquitetura [7].

Os autores aconselham a utilização de algumas tecnologias: *Context Toolkit* [42] e *OpenAAL* devido à complexidade associada ao desenvolvimento de uma plataforma deste tipo; *Windows Azure* [43] que é uma plataforma de *Cloud Computing* da Microsoft que providencia um ambiente de desenvolvimento distribuído para aplicações AAL; *Java Sun Spot* [44] que parece dar resultados positivos; tecnologias de semântica *web* como RDF [39] e OWL [7, 45].

2.2.5. Outros projetos de AAL

Em 2012 [46] é apresentado o projeto *UNIVERSal open platform and reference Specifications for Ambient Assisted Living* (UniversAAL), que teve início em 2010, onde foi criada uma plataforma aberta que providencia uma abordagem estandardizada, economicamente sustentável e tecnicamente viável, para desenvolver soluções para AAL. A plataforma é baseada em todo o estado da arte de AAL atual [47]. O projeto teve como objetivos principais [47]:

- Desenhar e estabelecer uma *uStore* baseada na loja de aplicações da *Apple*;
- Consolidar o trabalho realizado na área e integrá-lo;
- Utilizar uma estratégia para resolver os problemas de interoperabilidade da plataforma;
- Desenhar a plataforma para que ela possa trabalhar com os sistemas existentes;
- Aumentar o número de atores envolvidos no desenvolvimento de soluções AAL
- Demonstrar a utilidade prática da solução.

A arquitetura deste projeto é baseada na arquitetura do projeto Persona[6] [48-50].

O *Data Capture and Auto Identification Reference* (DACAR), terminado em 2010, foi um projeto que visa a prestação de cuidados de saúde baseado no paradigma de *Cloud Computing*[51]. O sistema criado no âmbito deste projeto utiliza o Microsoft HealthVault [52], uma vez que que este promove elevados níveis de segurança e de privacidade. O sistema permite que os utilizadores configurem os direitos de acesso aos seus dados clínicos para que posteriormente os especialistas de saúde possam aceder a esses dados sem restrições e de forma eficaz [51].

O *Human Activity Recognition Engine* (HARE), terminado em 2011, foi um projeto que deu origem a um sistema AAL robusto e eficaz para reconhecer e manipular a atividade humana de forma a compreender melhor as situações da vida quotidiana e a tomada de decisão para ambientes de cuidados de saúde à distância. A arquitetura deste projeto é baseada no paradigma de *Cloud Computing*[7] devido à sua elasticidade, escalabilidade e modelo *pay-as-you-go* [53].

Em 2008 surgiu um projeto que visa aumentar a qualidade de vida do grupo etário sénior, criando uma arquitetura aberta para integração e estandardização de serviços (*Open architecture for Accessible Services Integration and Standardisation* - OASIS). A arquitetura proposta é baseada em ontologias e permite *plug* & *play* e redução de custos em interoperabilidade, em conetividade e na partilha de conteúdos entre os serviços atuais e os serviços do futuro (Figura 9) [4].

[6] O projeto Persona está descrito na secção 2.2.3
[7] A definição de *Cloud Computing* encontra-se explicitada na secção 2.4.5.

Figura 9 – OASIS – Retirado de [4]

Em [54] é descrito o projeto Casattenta, terminado em 2010, que aplica uma *Wireless Sensor Network* (WSN) para monitorizar seniores com o objetivo de reconhecer eventos que obriguem a uma assistência imediata. O sistema proposto pelos autores é constituído por três elementos: as *Active Keys*, que são nós móveis que acompanham sempre os habitantes da casa, detetando anomalias nos sinais vitais dos mesmos; os *Monitoring Nodes*, que suportam todas as funcionalidades para manter a segurança na casa (controlo de temperatura, detetor de intrusos, entre outros); e o HG que é, não só o controlador central, mas também a unidade de comunicação que recolhe e processa todos os dados obtidos. O HG é configurável pelo utilizador, permitindo a criação de alarmes que respondem apropriadamente. Foi desenhado para ter um interface intuitivo que mostra os dados obtidos pelos sensores em tempo real. Para além disso, é possível adicionar-lhe extensões de visualização de informação.

O projeto *Service-oriented Programmable Smart Environments for Older Europeans* (SOPRANO), com inicio em 2007 e fim em 2010, foi criado com o intuito de permitir que os seniores europeus lidem de forma mais independente com as suas tarefas diárias. O projeto tem como objetivo desenvolver serviços com interfaces naturais para seniores a baixo custo, seguindo um modelo orientado a ontologias. Isto significa que as ontologias são utilizadas como um modelo dos dados internos dos componentes, criando modelos semânticos de alto nível para comunicação e utilização de serviços [55].

Existem ainda outros projetos como, por exemplo, o projeto *Successful Aging in a Networked Society* (AGNES) que providencia uma plataforma para manter uma rede social de utilização fácil e providenciar aplicações de suporte diário para seniores [56], que têm vindo a contribuir para os sistemas AAL. Grande parte desses projetos é listada pela associação de AAL [57].

2.3. Home Gateways em AAL

Para além das arquiteturas e características dos projetos analisados no capítulo anterior é necessário analisar outro tipo de soluções desenvolvidas nos últimos anos.

Nesta secção irá fazer-se uma revisão geral de grande parte das características necessárias de um HG. Essa revisão será feita tendo em conta o ano de publicação dos artigos de forma a obter uma evolução do conceito dos HG. Seguidamente irá descrever-se os principais tipos de arquiteturas e uma lista de arquiteturas existentes no contexto de AAL.

2.3.1. Evolução do conceito do HG

Em [58] (2000) é proposto um conjunto de características para um HG para propósitos de automação:

- Um interface de utilizador para acesso por parte da rede pública;
- Função de *firewall*;
- Funcionalidade de diretório de serviços que coleciona e apresenta as instancias dos equipamentos e respetivos serviços e conteúdos ao utilizador;
- Funcionalidade de controlo remoto;
- Funcionalidade de tradução de endereços e protocolos;
- Funcionalidade de *router* e de servidor de comunicação;
- Funcionalidades de tradução de multimédia.

Em [59] (2008) o HG é considerado o elemento-chave de uma *Home Area Network* (HAN). É defendido que gerir uma HAN é complexo devido à grande heterogeneidade de dispositivos ligados e de tecnologias utilizadas. Por outro lado, os utilizadores não querem ter trabalho a configurá-los e por esse motivo todos os dispositivos devem ser auto-configuráveis, auto-optimizáveis, auto-restauráveis, auto-protegidos e auto-geridos. Para além disso, o número de serviços é dinâmico e pode aumentar com os dispositivos ligados à casa, o que pode provocar problemas de escalabilidade.

Em [60] (2008) é feita uma análise geral dos dispositivos necessários em AAL e é feita uma análise das características necessárias num ambiente AAL. Os autores afirmam que são necessárias a deteção e a intervenção antecipada de acontecimentos clínicos anormais, a análise e interpretação de dados, a eficiência, a independência dos prestadores de cuidados, e a existência de custos reduzidos.

Em 2008 e 2010 os autores destacam a necessidade de interoperabilidade como uma característica imprescindível num HG, tornando-se este aspeto muito importante para sistemas de AAL[61, 62].

Em [54] (2009) é explicitado o projeto Casattenta em que o HG é, para além de controlador central, a unidade de comunicação que recolhe e processa todos os dados obtidos. O HG é configurável pelo utilizador permitindo a criação de alarmes que respondem apropriadamente e foi desenhado para ter um interface intuitivo que apresenta os dados obtidos pelos sensores em tempo real. Para além disso, é possível adicionar-lhe extensões de visualização de informação.

Ainda em 2009 evidencia-se a necessidade de quatro requisitos imprescindíveis numa plataforma de suporte a ALL: interoperabilidade de sensores/atuadores, por existir uma vasta gama de sensores e atuadores de vendedores diferentes que utilizam tecnologias diferentes; suporte à decisão, sendo que por razões de segurança no estado de saúde dos seniores, este sistema não pode falhar; interação com o utilizador; e troca de dados médicos relevantes do paciente [9].

No ano 2010 os autores referem que um HG de um sistema AAL deve providenciar a possibilidade de efetuar perguntas ao sistema, providenciar alarmes e notificações, detetar anomalias, reconhecer comportamentos e atuar [63]. São ainda, identificados alguns problemas que se prendem com a interoperabilidade, privacidade e autenticação durante a troca de informação, para além de se propor a utilização de tecnologia que suporte IPv6 [14].

Em [26] (2011) é apresentado o *Home Subsystem*, sendo que se descreve e se explica a implementação do HG. Segundo os autores, esta solução resolve problemas de interoperabilidade e implementa gestão remota de forma a aumentar a flexibilidade, usabilidade e confiança do sistema.

Finalmente, ainda em 2011 os autores dão especial enfase à necessidade de existir interoperabilidade, segurança, qualidade de *streaming* de vídeo/voz e a capacidade do armazenamento crescer dinamicamente [64].

Inicialmente, os HG eram vistos como unidades capazes de gerir a comunicação da rede. No entanto, estes tendem, cada vez mais, a ser unidades controladoras de todos os dispositivos e serviços da casa, providenciando interfaces e serviços simples de utilizar, de apoio ao dia-a-dia do utilizador, o que aumenta, tanto a complexidade dos HG, como as necessidades técnicas dos mesmos.

2.4. Arquiteturas de Software

Sendo a arquitetura de *software* uma das partes mais importantes do HG é necessário compreender quais as possibilidades de arquitetura para assim se utilizar a melhor opção.

O estudo de arquitetura de *software* começou no início dos anos 70, com enfâse na importância da estruturação do software. Em [65] os autores definem arquitetura como "*architecture is concerned with the selection of architectural elements, their interactions (…) design is concerned with the modularization and detailed interfaces of the design elements (…)*". Uma definição mais recente de arquitetura de *software* pode ser encontrada em [66].

Pode afirmar-se que a arquitetura de *software* é a estrutura do nosso sistema/programa e que deve ter em conta conceitos como compatibilidade, extensibilidade, confiabilidade, manutenção fácil, disponibilidade, segurança, usabilidade, entre outros. [65-67].

A Microsoft define estilo de arquitetura como um conjunto de princípios que devem ser utilizados para criar um sistema e divide as arquiteturas por quatro categorias: comunicação (*message bus*,

SOA) *deployment (Client/Server, N-Tier, 3-Tier)*, domínio (*Domain Driven Design*) e estrutura (*Component-Based, Object-Oriented, Layered Architecture*). A Microsoft considera ainda que a arquitetura de um sistema pode/deve combinar vários estilos de forma a obter a melhor solução possível [67].

Nesta secção serão abordadas algumas das arquiteturas distribuídas mais populares. Estas arquiteturas são apresentadas numa ordem temporal.

2.4.1. Arquitetura Cliente-Servidor

A abordagem cliente-servidor massificou-se com o crescimento da *Internet*, uma vez que proporcionou o acesso a múltiplos clientes apenas com um único servidor (Figura 10). Normalmente existe um sistema com muita performance (servidor) e vários sistemas com menos performance (clientes). O servidor é ponto central de registo e o único fornecedor de serviços. Já o cliente faz pedidos e não partilha recursos [68]. As principais vantagens deste sistema são o armazenamento de dados centralizado, a flexibilidade (sendo que é possível migrar serviços do servidor sem que os clientes se apercebam), o facto de a manutenção ser feita apenas no servidor, a segurança (havendo só um servidor este pode estar num local seguro), e a redundância. A principal desvantagem é a complexidade do sistema. Se a pessoa que o desenvolve não tiver formação suficiente este pode fracassar totalmente. Por outro lado, pode também ser complicado escalar o serviço. Um sistema mal pensado pode levar a custos extra associados à formação (para resolver o problema), manutenção e suporte [69];

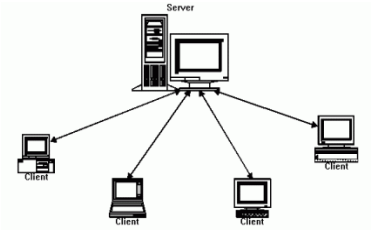


Figura 10 – Modelo Cliente-Servidor

2.4.2. Arquitetura Ponto a ponto (P2P)

No modelo *peer to peer* (P2P) cada nó da rede pode ser simultaneamente cliente e\ou servidor (Figura 11). Cada nó partilha parte do seu *hardware* (processamento, armazenamento, etc.) e acede diretamente a outros nós. Esta abordagem é bastante escalável (dependendo do número de nós) e redundante. No entanto, como existem múltiplos servidores, a manutenção e a realização de *backups* são bastante difíceis [68];

Figura 11 – Modelo P2P

2.4.3. Arquitetura Grid Computing

O *grid computing* é um sistema paralelo e distribuído que permite a troca, seleção e agregação em tempo de execução, de recursos distribuídos geográfica e dinamicamente, dependendo da sua disponibilidade, capacidade, custo, desempenho e dos requisitos de qualidade de serviços do utilizador. Ao contrário de P2P e de cliente/servidor em *grid*, existe um nó responsável por gerir os nós servidores e um ponto de acesso para os clientes. Este sistema tem a desvantagem de, por existirem vários nós, ser difícil fazer a manutenção e mais difícil escalar recursos do que em P2P. Este sistema é essencialmente utilizado para sistemas que necessitem de grande processamento (Figura 12) [70].

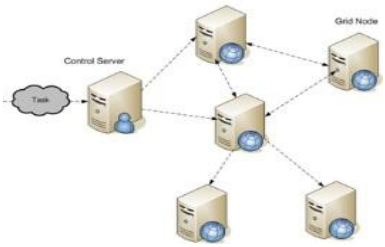


Figura 12 – Modelo Grid Computing

2.4.4. Arquitetura Orientada a Serviços (SOA)

Service Oriented Architecture (SOA) é um paradigma que permite organizar e utilizar capacidades distribuídas que podem ser utilizadas em vários domínios. Geralmente uma solução é feita para resolver ou suportar um determinado propósito. No entanto, essas soluções podem vir a ser necessitadas por outros. A grande vantagem de SOA é providenciar a possibilidade de as necessidades de uns utilizadores serem utilizadas para responder às necessidades de outros utilizadores [71].

Os princípios fundamentais de SOA são a:

- Visibilidade - capacidade de os consumidores com necessidades serem capazes de encontrar as ofertas dos fornecedores;
- Interação - capacidade de utilizar as ofertas dos fornecedores;

- Eficácia.

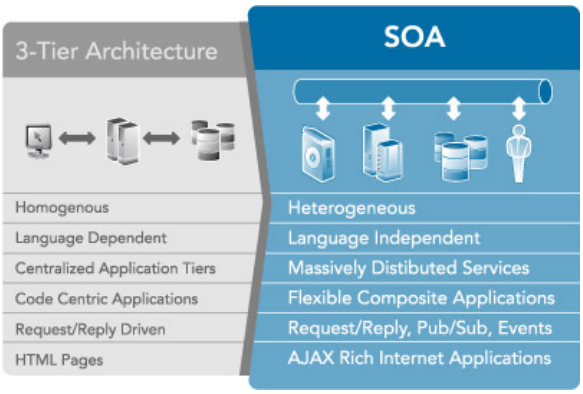


Figura 13 – SOA – Retirado de [71]

O elemento-chave de SOA é o serviço, sendo que SOA é composta apenas por serviços. O serviço pode definir-se pela "satisfação direta de uma necessidade pela utilização de um bem ou de uma prestação de trabalho". Para além do sentido da palavra, em SOA um serviço tem a capacidade de executar uma tarefa para outro adjacente, é a especificação de uma tarefa oferecido por outro serviço, e é a oferta da realização de uma tarefa por outro. Cada serviço deve ser independente dos restantes e ter uma arquitetura própria. Um serviço pode ser privado (utilizado apenas internamente), público (utilizado por terceiros) ou agregado a outros [71].

Outros conceitos fundamentais em SOA são o contexto de execução de um serviço (conjunto de elementos, processos, entidades, políticas de asserção e acordos que são identificados como parte de uma interação de um serviço) e descrição de serviço (que representa a informação necessária para se utilizar o serviço) [71].

O paradigma SOA proporciona algumas vantagens, tais como cada um dos seus serviços ser totalmente independente (e, portanto, reutilizável) e encapsulado (escondendo a complexidade), de ser altamente escalável, de diminuir custos na cooperação entre organizações e de facilitar a complexidade de gestão de sistemas muito complexos [33, 71, 72].

Como todas as arquiteturas de *software*, o SOA tem igualmente alguns princípios e boas práticas que deverão ser respeitados de forma a garantir o sucesso [72, 73]:

- Reutilizável - os seus serviços devem poder ser utilizados por outras aplicações e/ou serviços, no mesmo cenário ou em cenários distintos;
- Granularidade - deve responder especificamente ao problema;
- Modularidade - deve conter módulos capazes de processar um conjunto de instruções de modo autónomo;
- Composability e componentization - deve permitir a construção de serviços mais elaborados através de orquestração ou encadeamento, baseados noutros serviços;

- Interoperabilidade e estandardização - deve ser independente da linguagem de programação, obedecendo a padrões;
- Os serviços deverão ser identificados e catalogados, aprovisionados e entregues, e monitorizados de modo a que se facilite a tarefa de pesquisa, orquestração ou encadeamento, e uso de um determinado serviço para a operação a realizar.

2.4.5. Cloud Computing

A *Cloud Computing* é um modelo de partilha de recursos de diferentes propriedades, diferentes tipos, diferentes regiões e diferentes tempos de recursos computacionais na rede através da virtualização. Estes recursos são depois agrupados, reestruturados ou separados, para serem entregues ao utilizador, segundo as suas exigências [74]. A *Cloud Computing* assegura uma elevada adaptabilidade graças à sua grande capacidade de computação e armazenamento massivo.
Argumenta-se ainda que esta técnica pode diminuir custo associados com investimento, manutenção e gestão, pagando-se à medida do que se necessita [74, 75].
A *Cloud* pode oferecer praticamente tudo como serviço (aplicações, plataformas e infraestrutura). Neste modelo podem utilizar-se aplicações/software como serviços (***Software as a Service*** - SaaS) oferecendo, assim, soluções parciais, componentes, e serviços individuais para que o utilizador possa criar a sua própria solução. Podem ainda utilizar-se plataformas como serviços (***Plataform as a Service*** - PaaS) facilitando, assim, o desenvolvimento de aplicações sem o custo e a complexidade de comprar e gerir as camadas mais inferiores de *hardware* e *software*. Oferece, ainda, infraestrutura como serviço (***Infraestruture as a Service*** - IaaS), que disponibiliza *hardware* (máquinas, processamento, armazenamento, dispositivos, etc.) como serviço [75]. É neste ultimo conceito que se enquadra o conceito ***Device as a Service***, onde um dispositivo é disponibilizado por um serviço.

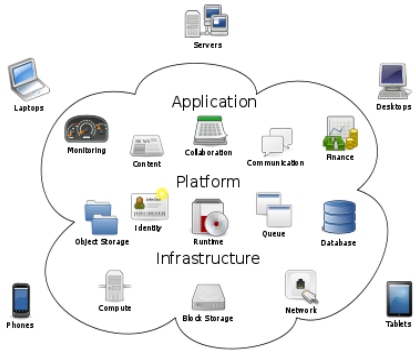


Figura 14 – Cloud Computing

Este modelo oferece a vantagem de qualquer indivíduo se poder servir de forma independente, de estar disponível em qualquer local/dispositivo desde que exista *Internet*, de se obter processamento

e armazenamento balanceado ao longo da infraestrutura sem que nenhum destes esteja associado exclusivamente a um individuo, de ser muito elástico (os consumidores podem aumentar ou diminuir a sua capacidade à medida das necessidades), e de se pagar à medida que os recursos são consumidos. Pode ainda ser bastante vantajoso para as empresas, uma vez que evita o investimento de capital em *hardware*, *software* e serviços, passando estas a pagar somente aquilo que utilizam [75].

No entanto, este tipo de solução prende frequentemente os consumidores aos fornecedores e é bastante complexo em termos legais. Por outro lado, em algumas situações, os custos associados à utilização de *Cloud Computing* podem não ser vantajosos a longo prazo [75].

2.5. Arquiteturas para HG

Para além das arquiteturas expostas nos projetos AAL existem outras propostas de arquiteturas de vários autores para o HG. Nesta secção apresentam-se algumas das propostas que foram analisadas. Em [59] (2008) os autores pretendem provar o conceito de um sistema automático de forma a oferecer alta escalabilidade, implementando-o. Para isso, o HG deve funcionar de forma automática descobrindo serviços e dispositivos sem interferência do utilizador.

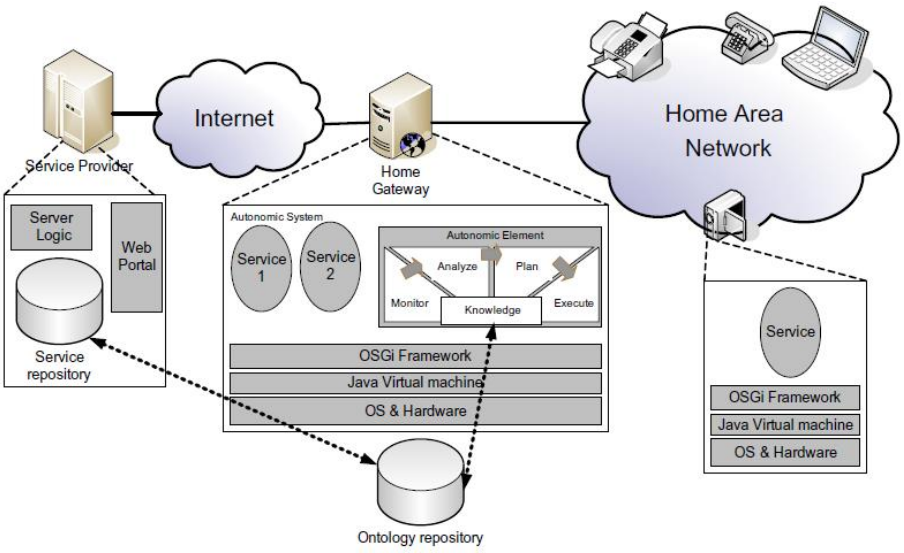


Figura 15 – Arquitetura de sistema automático – Retirado de [59]

Na Figura 15 está representada a arquitetura do sistema acima descrito. O HG é baseado na norma OSGi e desenhado para seguir a descrição de arquitetura da IBM de um elemento automático. O sistema é essencialmente composto por cinco módulos [59]:

- O conhecimento (*Knowledge*) que é necessário para descrever o domínio de que o elemento automático necessita, sendo representado por um modelo conceptual e formal chamado ontologia (obtida a partir do *Ontology repositor*);

- O monitor (*Monitor*) que captura as características técnicas do HG e do ambiente que o rodeia (por exemplo, os dispositivos da Home Area Netwoek (HAN)).
- Analisador (*Analyser*), que é responsável pela funcionalidade de raciocínio. Os dados do sistema são analisados de acordo com determinadas regras
- O plano (*Plan*) que corresponde às decisões inferidas pelo sistema automático.
- A execução (*Execute*) que vai operar sobre os recursos.

Resumidamente, o HG obtém os serviços que podem ser utilizados na HAN, utilizando um conjunto de módulos para obter conhecimento autonomamente e acedendo ao repositório de serviços, sendo que, por fim, faz *deploy* dos serviços utilizando o OSGi [59].

Em [60] (2008) os autores explicam como se devem adquirir os dados clínicos do sénior. A arquitetura proposta é baseada num terminal capaz (terminal *InHome*) que é responsável por obter os dados de todos os dispositivos médicos a que está ligado, enviando-os, posteriormente, para o HG (que é responsável por providenciar os acessos para o exterior da rede, e conter serviços personalizados) por WLAN que posteriormente os envia para as aplicações que os guardam, processam e apresentam. O terminal é ainda responsável por monitorizar a atividade do sistema, gerir o ambiente, calendarizar tarefas, gerir aplicações e providenciar monitorização remota.

Em [62] (2010) os autores apresentam uma solução de interoperabilidade baseada em WS (que obriga à criação de novos serviços quando existem novos tipos de dispositivos) para ambientes AAL. Esta solução segue uma abordagem orientada aos serviços. Esta é constituída por um componente central (o *domoNet*) responsável por gerir os restantes componentes que irão controlar os vários tipos de dispositivos do HS. Para comunicação é utilizada a linguagem XML que consegue abstrair os dispositivos num determinado tipo de dispositivo.

Em [14] (2010) os autores defendem uma arquitetura baseada em IoT para proporcionar serviços AAL para pessoas idosas ou com doenças. No entanto, este tipo de arquitetura foi desenhado para baixos custos, consumos e tamanhos (permitindo o desenvolvimento em grande escala) que não conseguem lidar com a mobilidade e segurança do IPv6. Esta arquitetura está baseada em três pilares: providenciar conectividade aos dispositivos (automação residencial, segurança, controlo e comunicação); utilizar 6LoWPAN (IPv6 *based Lower-Power Personal Area Network*) para comunicação ativa; e utilizar RFID (*Radio Frequency Identification*) e NFC (*Near Field Communication*) para comunicação passiva.

Outros problemas identificados por estes autores referem-se à interoperabilidade, privacidade e autenticação durante a troca de informação. A proposta de resolução destes últimos visa a utilização de uma camada de comunicação que utiliza MD5. Os autores defendem, por fim, que desenvolveram uma arquitetura muito modular, flexível, eficiente, escalável e acessível, dando

principal relevância a esta última, já que os utilizadores devem fazer configurações *ad-hoc* que se adequem às tecnologias de que necessitam.

Em [63] (2010) é apresentada uma solução AAL composta por dois elementos: o *Gateway* e o servidor central. O *Gateway* tem a capacidade de obter dados de um grande número de sensores, processá-los e transmiti-los para o servidor central. O servidor central irá guardá-los e prepará-los para serem utilizados pelas aplicações clientes. No caso de os dados serem extremamente sensíveis, o armazenamento de dados é feito no *Gateway* e os resultados podem ser transmitidos diretamente para os utilizadores autorizados. Neste último caso, o servidor central tem apenas o papel de autenticar o utilizador e de fazer manutenção e gerir as configurações de *deployment* do *Gateway*.

Em [64] (2011) os autores propõem um sistema AAL baseado em IoT. Para além de apresentarem várias soluções tecnológicas de interoperabilidade, apresentam uma solução baseada no protocolo RTCP para garantir a qualidade de *streaming* de vídeo e áudio, apresentam uma solução baseada em MD5 para aumentar a segurança do sistema e apresentam uma solução baseada em *Cloud Computing* para providenciar elasticidade no armazenamento do sistema.

2.5.1. SOA em AAL

Em [9] (2009) refere-se que um sistema AAL é caracterizado por necessitar de uma grande configuração, ser muito dinâmico ao longo do tempo e incluir muita variedade de componentes e subsistemas de diferentes vendedores e indústrias que comunicam de forma diferente, dando uso a um amplo espectro de tecnologias. Para além disso, devem ainda ser considerados aspetos de usabilidade. Ainda no artigo acima referido, os autores mencionam que os princípios de SOA parecem emparelhar com os requisitos de AAL. Os princípios de encapsulamento, contrato de serviços, e abstração ajudam a lidar com a complexidade associada aos vários sensores e tecnologias. O princípio de fraco acoplamento facilita a implementação de um sistema de apoio a decisões que pode ser ligado a um sistema existente baseado nas necessidades do utilizador. A descoberta de serviços facilita a auto-configuração do sistema. Existem, também, *frameworks* que providenciam suporte para manutenção remota [9].

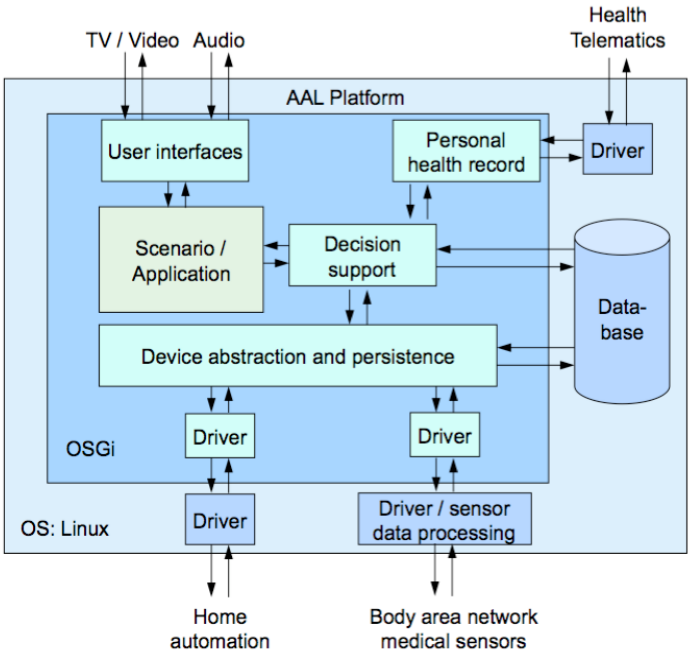


Figura 16 – Arquitetura SOA e OSGi framework – Retirado de [9]

A arquitetura proposta pelos autores utiliza o OSGi *framework* e é constituída por várias camadas (Figura 16). A camada de abstração e persistência (*Device abstraction and persistence*) permite que o algoritmo aceda a dados históricos e atuais dos sensores, utilizando um único interface que abstrai as características únicas dos dispositivos sempre que possível. A camada de interface de utilizador (*User interfaces*) implementa a interação entre o sistema e o habitante (através de vários dispositivos). A camada de suporte à decisão (*Decision support*) processa e interpreta os dados dos sensores e depende, geralmente, do contexto do utilizador. A camada de registo médico pessoal (*Personal health record*) segue a informação médica relevante do utilizador, que pode ser gerada pela camada de suporte à decisão ou fornecida pelo especialista de saúde. Por fim, a camada de aplicação (*scenario/application*) é o *software* que interage com o utilizador quando necessário através de interface multimodal, produzindo decisões como alarmes, avisos ou recomendações sempre que algo anormal acontece, e controla os dispositivos da casa [9].

2.6. Tecnologias de base para desenvolvimento de HG

Nesta secção são descritas as principais tecnologias utilizadas no desenvolvimento do HG.

2.6.1. WEB Services

Web Services (WS) são componentes de uma aplicação e comunicam utilizando o protocolo livre *HTTP* (SOAP ou REST). Para além de serem autossuficientes e auto-descritos, podendo ser

descobertos utilizando UDDI, através do seu WSDL que os descreve, podem ser utilizados por outras aplicações/serviços, e têm como base a utilização de XML [76].
As mensagens entre o cliente e o servidor poderão ser trocadas em formato *Extensible Markup Language* (XML) ou *JavaScript Object Notation* (JSON).

2.6.1.1. Web Service Description Language (WSDL)

O WSDL é uma derivação do formato XML utilizada para descrever WS (como um conjunto de operações, as suas entradas e saídas) e para localizar WS. Define o serviço com um conjunto de operações presente na rede. A abstração na definição do conjunto de operações e das mensagens é separada da implementação dos protocolos de rede sobre os quais assenta a comunicação com o serviço [76].

2.6.1.2. Simple Object Acess Protocol (SOAP)

O SOAP é descrito pela W3C como um protocolo de comunicação leve e extensível, que permite a troca de informação descentralizada, numa arquitetura distribuída. É independente da linguagem e da plataforma, é baseada em XML e é um *standard* W3C [77].
Este protocolo é constituído por 3 partes: (1) o envelope constituído pelo *Header* e pelo *Body*, sendo que é nele que está toda a informação que irá ser trocada; (2) as regras de codificação dos parâmetros para XML, uma vez que poderão ser objetos compostos; (3) o *Remote Procedure Call* (RCP) que é uma tecnologia de comunicação entre processos e permite a invocação remota de um método [77].
O uso de SOAP tem vantagens, uma vez que é baseado em XML e permite ser usado sobre diferentes protocolos de comunicação. O protocolo padrão é o HTTP, embora seja possível usar, por exemplo, o protocolo SMTP. Como grande parte das implementações de SOAP são feitas sobre o protocolo HTTP são evitados problemas de *firewalls* e *proxies*, uma vez que se está a considerar um dos protocolos de rede mais usado. No entanto, não usa os métodos disponíveis no protocolo HTTP (PUT, POST, GET, DELETE). Este protocolo tem a desvantagem de diminuir o desempenho devido, essencialmente, à grande complexidade associada às suas mensagens [77].

2.6.1.3. Universal Description, Discovery and Integration (UDDI)

A *Universal Description, Discovery and Integration* (UDDI) é considerada um protocolo padrão pela OASIS, especificando um método para publicar e descobrir serviços num diretório para uma arquitetura orientada a serviços [78], também designado *broker*.
A UDDI foi desenvolvida para ser interrogada via SOAP e providenciar o WSDL, que descreve as operações, os protocolos e as mensagens que um serviço suporta. Assim, um serviço poderia,

facilmente, alterar a sua URI, uma vez que, existindo a interrogação por parte do cliente à UDDI, este saberia sempre qual a sua localização nesse momento [78].

2.6.1.4. REpresentational State Transfer (REST)

Sendo este um protocolo cliente-servidor, pressupõe que não existe um estado, isto é, cada mensagem trocada contém toda a informação necessária para a sua execução. Desta forma, as partes envolvidas não necessitam de gravar a informação sobre o estado das comunicações efetuadas. Este protocolo de comunicação assume que tudo na *Internet* é uma representação de um recurso, logo, tudo pode ser acedido remotamente, sendo que cada representação tem a sua URI. Assim, é possível navegar entre as diversas representações bastando, para isso, seguir as ligações sem ser necessário, por exemplo, o uso de um registo. Para além disso, o REST utiliza os métodos disponíveis no protocolo HTTP (PUT, POST, GET, DELETE) [79].

Para funcionar corretamente é necessário que REST seja cliente-servidor, isto é, que a comunicação seja feita através de uma interface conhecida. Isto permite a separação de conceitos, uma vez que não interessa ao cliente o modo como o servidor armazena os dados, aumentando a portabilidade do código do cliente. Por outro lado o servidor não tem preocupação sobre o estado do utilizador, aumentado assim a sua escalabilidade. REST deverá ser *stateless*, isto é, o contexto do cliente não é relevante para o servidor. Como cada pedido contém toda a informação necessária para a sua execução, o estado é da responsabilidade do cliente, facilitando, assim, a monotorização dos pedidos por parte do servidor, e aumentando a confiança no servidor face a falhas na rede. REST providencia sistemas de *cache* das respostas por parte do cliente, uma vez que implícita ou explicitamente estas definem as suas propriedades de *cache*, podendo, assim, diminuir o número de pedidos entre o cliente e o servidor [79].

2.6.1.5. Extensible Markup Language (XML)

O XML pressupõe a definição do *schema* associado à interface que define o recurso em questão. Como tal, acarreta um processamento elevado, uma vez que é necessária a serialização e desserialização dos objetos contidos nas mensagens trocadas. Para usar XML como formato para a comunicação com um serviço, é necessário ter conhecimentos noutras tecnologias, nomeadamente: XPath, *Schema* XML, XSLT, XML *Namespace*, DOM [80].

2.6.1.6. JavaScript Object Notation (JSON)

O JSON é um formato leve, aberto, baseado em texto para protocolos de comunicação. Tal como o XML é lido por humanos e máquinas. É derivado do JavaScript para a representação de estruturas de dados e listas associativas, mas diferencia-se do JavaScript uma vez que é independente da

linguagem de programação, e existem *parsers* para quase todas as linguagens de programação. O JSON é frequentemente usado para a serialização (e desserialização) e transmissão de estruturas de dados sobre protolocos de comunicação. É usado como alternativa ao XML no SOA. Está estruturado como uma lista de objetos, sendo que estes são constituídos por uma coleção de chave/valor. A chave é uma cadeia de caracteres, e o valor podem referir-se aos tipos suportados pelo JavaScript, ou ainda a um objeto com a sua coleção de chave/valor. O uso de JSON é mais vantajoso que o uso de XML nos serviços, porque permite que a informação que é trocada entre o servidor e o cliente seja reduzida ao mínimo, uma vez que é apenas enviada a informação necessária. A utilização de JSON apresenta vantagens na curva de aprendizagem, uma vez que para os programadores que programam no paradigma orientado aos objetos, os conceitos mantêm inalterados, em oposição à curva de aprendizagem do XML descrita anteriormente [81].

2.6.2. Open Services Gateway initiative (OSGi)

O *Open Services Gateway initiative* (OSGi) é um conjunto de especificações que define um sistema dinâmico para Java que visa reduzir a complexidade, providenciando uma arquitetura modular, orientada a serviços. As especificações permitem que os componentes escondam a complexidade associada aos outros componentes, fazendo com que estes comuniquem como serviços. Este modelo é utilizado em muitas e variadas soluções, inclusivamente soluções de AAL [3].

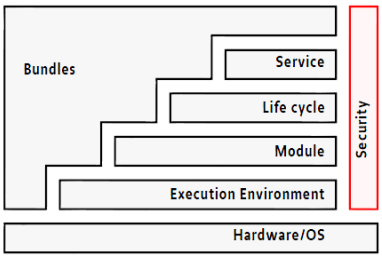


Figura 17 – Camadas OSGi – retirado de [3]

O OSGi está dividido em várias camadas, como se pode verificar na Figura 17 [3]:

- A *Security Layer*, que se baseia na segurança de *Java 2* [82] embora adicione um número de restrições e preencha algumas falhas do mesmo [3];
- A *Module Layer*, que define o modelo de modularização para o Java. Esta contém um conjunto de restrições associadas à partilha de módulos/serviços noutros módulos/serviços [3];
- A *Life Cycle Layer*, que providencia uma *API* para gerir o ciclo de vida dos módulos/serviços;
- A *Service Layer*, que providencia um modelo de programação conciso, dinâmico e coerente que simplifica o desenvolvimento e *deploy* de módulos/serviços [3].

De acordo com [83], este modelo beneficia todos os intervenientes principalmente os modelos de negócio, visto que proporciona a redução da complexidade, reutilização e uma adaptação rápida ao mundo real (por ser extremamente modular). Proporciona um fácil desenvolvimento e atualizações dinâmicas. Os vários módulos/serviços do OSGi podem ser instalados, descarregados, removidos e atualizados de uma forma dinâmica e escalável sem que isso afete o restante sistema, sendo que o OSGi é o responsável por gerir as dependências entre os vários serviços. O OSGi proporciona ainda transparência, utilização de versões, simplicidade, rapidez e segurança. É não intrusivo, muito utilizado e apoiado por algumas das companhias mais importantes na área.

2.6.3. Axis2

O projeto *Axis2* da *Apache* é implementado em *Java*. Este projeto é uma evolução do projeto *Axis1* tirando vantagem da experiencia já adquirida. Este projeto providencia um modelo de objetos e uma arquitetura modular que torna fácil adicionar e suportar novos serviços *Web* [84].
O *Axis2* facilita o envio, a receção e o processamento de mensagens SOAP, o retorno do WSDL do serviço, a criação e utilização de REST, a criação e utilização de serviços que utilizem *WS-Security*, *WS-ReliableMessaging*, *WS-Addressing*, *WS-Coordination* e *WS-Atomic Transaction*, a criação de um serviço a partir de uma classe normal e a criação de classes para o cliente e para o servidor utilizando o WSDL [84].

2.6.4. Glassfish Server

O *Glassfish Server* é um servidor de aplicações gratuito, apoiado pela comunidade, e que suporta a plataforma Java EE 6. Por forma a este sistema providenciar modularidade e extensibilidade, foi desenvolvido utilizando o modelo OSGi. O *Glassfish Server* é também utilizado para desenvolver aplicações orientadas ao serviço, modulares, dinâmicas e extensíveis. O servidor permite a interação entre os serviços OSGi e os componentes do *Java EE*, permitindo, assim, que as aplicações beneficiem tanto das vantagens associadas ao *Java EE* como das vantagens associadas ao modelo OSGi [85].

3. Home Gateway e Serviços do LUL – Conceptualização e Desenvolvimento

Neste capítulo introduz-se a arquitetura de desenvolvimento do projeto LUL, e respetivos elementos. Seguidamente apresenta-se um cenário de utilização e listam-se os requisitos de desenvolvimento do HG. Finalmente, explica-se a arquitetura de software do HG, e respetivos serviços.

3.1. Modelo Conceptual do projeto LUL

A arquitetura de desenvolvimento do LUL está dividida em quatro camadas principais: a camada de apresentação (*Applicational Layer*), a camada dos serviços de *Living Lab* (*Living Lab Services Layer*), a camada de serviços comuns (*Common Services Layer*) e a camada de infraestrutura (*Infrastructural Layer*) (Figura 18) [1, 10].

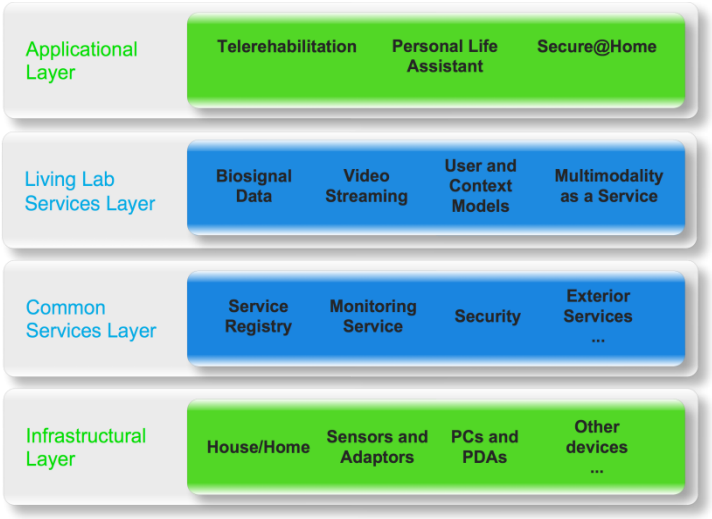


Figura 18 – Modelo conceptual do projeto LUL – Adaptado de [1, 10]

A camada de apresentação incorpora as aplicações desenvolvidas, que serão utilizadas tanto pelo sénior como pelo especialista de saúde. Sendo esta a camada responsável por conter grande parte da interface humano-computador, é onde a usabilidade do sistema, nomeadamente a multimodalidade do mesmo, é fulcral [1, 10].

A camada de serviços *Living Lab* visa envolver todos os serviços específicos para o utilizador, desde serviços para a utilização de certos dispositivos, como serviços para a recolha de dados e, ainda, outros serviços cruciais para o desenvolvimento de determinadas aplicações. Alguns exemplos destes serviços são aqueles que obtêm contexto e conseguem fornecer adaptações de interação, ou que permitem o controlo de câmaras de vídeo dentro da habitação do indivíduo sénior [1, 10].

A camada de serviços comuns é responsável por fornecer serviços que permitem gerir acessos entre os vários nós, monitorizar conexões e sessões, aumentar a segurança e gerir utilizadores [1, 10].
Por fim, a camada de infraestrutura é composta por dispositivos que serão utilizados pelas aplicações, ou, na maioria dos casos, por serviços das restantes camadas. Como a adição e remoção dos dispositivos são feitas de forma dinâmica, é esperado que estes sejam acessíveis através de *Web Services* (WS) [1, 10].
A camada de serviços de *Living Lab* e de infraestrutura correspondem ao *Home Site* (HS), pelo que é onde o *Home Gateway* (HG) atua.
A heterogeneidade do projeto LUL obriga a que se recorra às NGN como solução para a camada de comunicação, devido à necessidade de providenciar apoio a um leque variado de serviços, aplicações e mecanismos [1, 10].

3.2. Componentes principais do projeto LUL

O projeto LUL está associado a três noções essenciais: (1) o HS diz respeito à localização da população sénior (no cenário utilizado, este corresponde à sua habitação) e contém um variado conjunto de recursos que estão disponíveis como serviços através do HG, elemento responsável pelo controlo de todos os recursos do HS; (2) o *Health Professional Site* (HPS) representa a localização do especialista de saúde, que, por sua vez, tem ao seu dispor um conjunto de recursos que age como um cliente do HS; (3) o LUL *Main Server* (LULMS) tem como função manter uma lista completa dos especialistas de saúde e dos seniores registados no sistema, sendo ainda responsável por identificar o HS e o HPS, por listar os recursos em cada local e por gerir os eventos [1, 2].

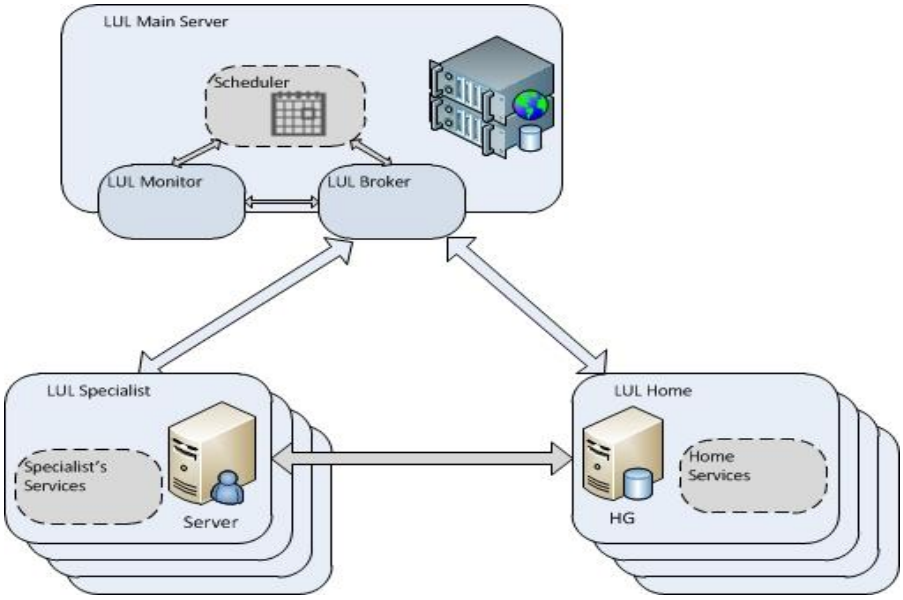


Figura 19 – Componentes principais do LUL – Adaptado de [1, 2]

Para além disso, e relativamente a esta última conceção apresentada, tal como é possível observar na Figura 19, é o LULMS que fornece informação ao HPS, que se conecta ao HS. Na HS o HG gere a disponibilização dos vários serviços disponíveis, providenciando, deste modo, uma comunicação eficaz e segura[1, 2].

3.3. Visão Geral do HG

O HG é o elemento que interliga o HS à *Internet* e, consequentemente, ao LULMS e à HPS. É importante referir que, por essa razão, a *Home Site Network* (HSN) é uma rede isolada pelo HG, sendo este o único responsável pelas interações com a mesma. Assim, é fácil perceber que o HG tem um papel fulcral na HSN (Figura 20).

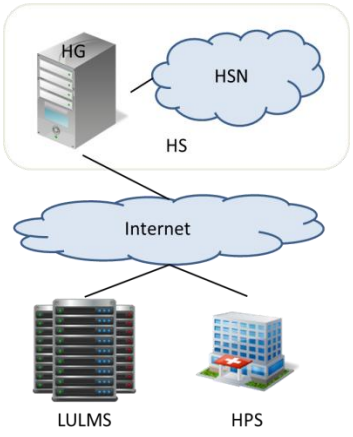


Figura 20 – Home Gateway do LUL - Visão Geral

Num cenário próximo da realidade, todos os dispositivos de uma habitação vão estar localizados na HSN. Estes dispositivos irão ser controlados pelo HG ou, em alguns casos, irão enviar informação para o mesmo. Os controlos poderão ser decisões autónomas do HG, poderão ser pedidos de algum dos serviços, ou poderão ser pedidos efetuados por utilizadores autorizados fora da HSN. Os dispositivos na HSN podem, igualmente, enviar informação pertinente para o HG, para que esta possa ser analisada/utilizada pelos diversos serviços do sistema.

Esta secção inicia-se com uma pequena apresentação de um cenário de utilização do HG, seguida de um resumo, e devida explicação dos requisitos obtidos para o desenvolvimento do HG e respetivos serviços. Detém, ainda, uma descrição da arquitetura do HG, dos seus módulos e de como estes podem ajudar a responder a cada um dos requisitos. Por fim, descrevem-se alguns dos serviços que estão a correr no HG.

3.4. Cenário de Utilização do HG

O HG pode ser utilizado em vários cenários. Nesta situação é descrito um possível cenário que se aplica ao projeto LUL (Figura 21).

Tome-se em consideração a situação hipotética em que há uma aplicação situada no HPS e outra aplicação situada no HS, e que o especialista de saúde e o sénior vão estabelecer uma sessão remota de prestação de serviços de saúde.

Momentos antes de a sessão ter início, a aplicação do sénior e a aplicação do especialista de saúde vão estabelecer uma conexão com o LULMS e informá-lo de que estão *online*. Nesse momento, o especialista de saúde passará a ter acesso à informação de conexão do sénior, podendo deixar de comunicar com o LULMS e passando apenas a comunicar com o HG.

Quando a comunicação entre o HS e o HPS é direta, o HG é responsável por fornecer ao especialista de saúde os serviços que este pretende. Para isso, terá de verificar se o mesmo tem privilégios, e terá de informar o sénior de que alguém quer efetuar uma sessão com ele (podendo este negar a realização da sessão). Se a sessão puder proceder normalmente, é responsabilidade do HG disponibilizar os serviços para utilização externa (limitando o acesso a serviços cujo especialista de saúde não tem privilégios) e de informar o LULMS quais os recursos que estão em utilização que, por sua vez, os vai associar à sessão. Assim, caso algum dos intervenientes perca a conexão, o LULMS pode recuperá-la facilmente, informando o HG que deve disponibilizar os recursos que estavam em utilização. É igualmente responsabilidade do HG informar o LULMS sempre que um serviço/recurso deixa de ser utilizado.

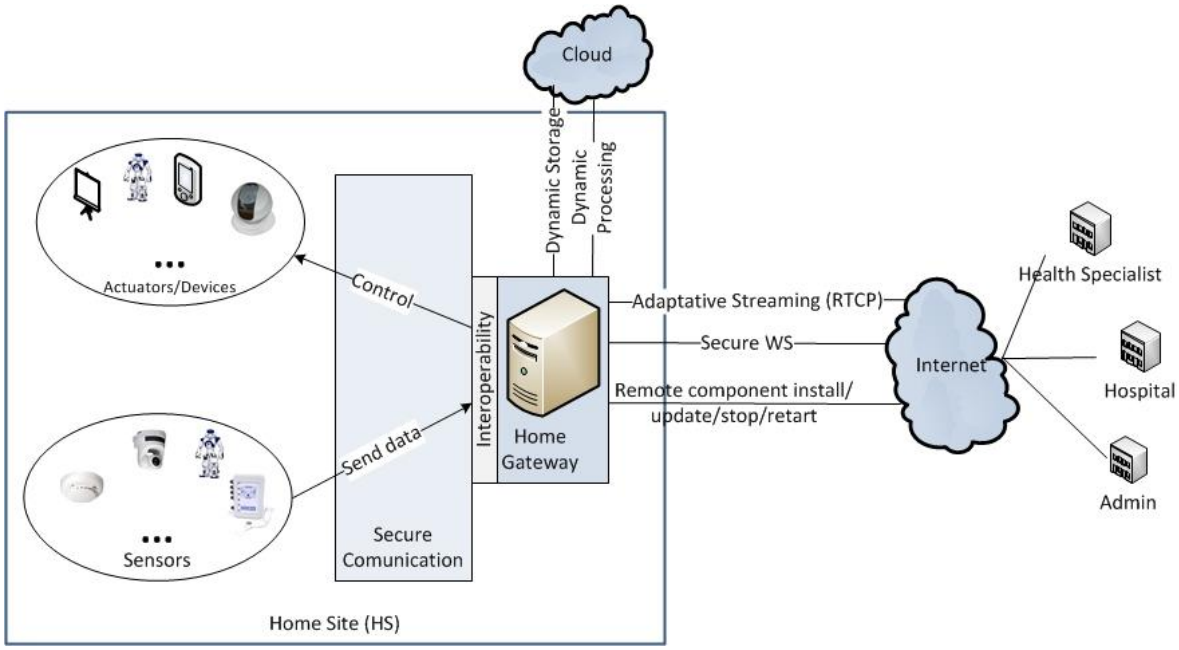


Figura 21 – Cenário de utilização do Home Gateway

Os serviços podem incluir o controlo de dispositivos, de aplicações ou apenas ler dados obtidos dentro da casa, em tempo real ou dos últimos dias.

Ao longo da sessão é transferida uma grande quantidade de informação para o exterior, nomeadamente no caso de a sessão estar a ser efetuada com recursos multimédia. Neste caso, para uma comunicação segura e eficaz, o HG deve ser capaz de adaptar os dados multimédia e os seus recursos de tal forma que a comunicação continue a fluir normalmente.

Pode ainda acontecer que, no momento em que um sénior está em sessão, um administrador tenha a necessidade de aceder aos serviços da HS. Nesta situação, o sistema pode avisar o sénior que o administrador está a aceder ao sistema. No entanto, esse procedimento não pode afetar os recursos que estão a ser utilizados. A razão deste acesso pode dever-se à necessidade de manutenção que pode incluir instalação ou remoção de novos serviços, ou pode ser apenas para mudar algumas configurações base do sistema/serviços.

No caso extremo de o HG estar a utilizar uma grande quantidade dos seus recursos em simultâneo, pode ser necessário aceder a recursos externos ao HS, como é o caso da *Cloud*, para que o sistema não falhe. O HG tem a responsabilidade de fazer esta gestão de recursos e decidir, autonomamente, quando devem utilizar os recursos externos.

Caso algum dos serviços detete uma situação de emergência, o especialista de saúde (se tiver privilégios suficientes para isso) pode aceder aos serviços da habitação por forma a verificar o sucedido. Neste último caso, o sénior pode igualmente barrar o acesso para prevenir falhas do sistema, ou seja, se o sistema detetar uma situação de emergência, o sénior saberá que isso aconteceu e, caso não seja verdade, pode cancelar esse sinal.

3.5. Requisitos do HG

Da análise de vários projetos AAL anteriormente em conjunto com a análise feita pelos membros do projeto LUL surgiram vários requisitos. Alguns desses requisitos não se enquadram no âmbito do desenvolvimento do HG, pelo que esta secção apenas terá como ênfase os requisitos do HG e respetivos serviços.

Um sistema de AAL tem uma grande variedade de requisitos tanto técnicos como de utilizador. É importante lembrar que estes sistemas serão utilizados por pessoas seniores que, na sua maioria, não têm facilidade em lidar com novas tecnologias e que, portanto, merecem especial atenção na análise de requisitos.

3.5.1. Requisitos Técnicos do HG e Serviços

Os requisitos técnicos obtidos na fase de análise de requisitos para o HG são:

- **Interoperabilidade** – A integração de vários dispositivos que utilizam protocolos de comunicação diferentes e de fornecedores diferentes é crucial. Quando se idealiza um sistema de AAL não se pode considerar apenas os dispositivos que vão ser utilizados no momento do desenvolvimento do sistema, mas também os dispositivos em geral. Sem esta característica o sistema torna-se rapidamente obsoleto, uma vez que não permite a fácil integração de vários dispositivos, ficando, muito provavelmente, preso a fornecedores/protocolos de comunicação;
- **Segurança** – No contexto de AAL existe troca, armazenamento e processamento de dados bastante sensíveis. É portanto crucial manter os níveis de segurança maximizados. Para isso, deve ser garantida a disponibilidade, integridade e confidencialidade dos dados. A disponibilidade dos dados assegura que os mesmos estarão acessíveis em qualquer momento. A integridade e a confidencialidade garantem que os dados não são alterados por terceiros ou corrompidos e que os dados não são acedidos por terceiros, respetivamente;
- **Escalabilidade** – As necessidades do utilizador do sistema não serão sempre iguais, por isso, a quantidade de recursos necessários pode crescer ou diminuir. Por esse motivo, é importante garantir que o sistema tem a capacidade de escalar os seus recursos, tanto a nível de armazenamento como a nível de processamento;
- **Integração de um número arbitrário de dispositivos** – Inicialmente o sistema pode ter um número limitado/bem definido de dispositivos. No entanto, é conveniente que esse número possa aumentar de forma ilimitada sendo, por isso, necessário que o sistema seja capaz de integrar tantos dispositivos quantos forem necessários;

- **Sistema automático** – De forma a facilitar a configuração do sistema é conveniente que o sistema seja capaz de descobrir, de forma automática, quais os dispositivos que estão na rede, e de instalar e configurar automaticamente os serviços associados a esses dispositivo;
- **Processamento paralelo** – Num sistema com um grande número de serviços é crucial a existência de processamento paralelo, embora este deva ser efetuado de forma contida para diminuir os problemas de segurança associados a serviços que requerem alta disponibilidade;
- **Modularidade** – Usualmente, um sistema AAL é bastante complexo pelo que é importante que este seja modular, de forma a suportar extensibilidade, distribuição e reutilização;
- **Monitorização** – O sistema deve ter as capacidades de auto-monitorização e de recuperação de falhas. É importante que um sistema AAL seja capaz de autonomamente monitorizar os seus componentes e também os seus serviços. Deve também monitorizar o seu desempenho e eficiência. Esta característica é crucial para que o sistema aumente a sua segurança e escalabilidade;
- ***Firewall*** – O HG deve fazer função de *firewall*, controlando tudo o que entra e sai da HSN;
- **Gestão de serviços** – O HG deve providenciar mecanismos de descoberta de serviços, suportar o encadeamento de serviços, suportar mecanismos de ligação de serviços, providenciar orquestração e providenciar composição de serviços, facilitando, assim, a utilização dos mesmos;
- **Recolha de dados** – O sistema deve ter serviços com a capacidade de obter dados dos dispositivos;
- **Processamento de dados** – O sistema deve ter serviços com a capacidade de processar os dados, transformando-os em informação;
- **Comunicação síncrona e assíncrona** – O sistema deve ser capaz de providenciar comunicação síncrona mas também assíncrona;
- ***Cloud Computing*** – É importante que o sistema seja capaz de suportar este paradigma, uma vez que esta é uma tendência cada vez mais utilizada e proporciona a capacidade de escalar o sistema em todos os aspetos;
- **Gestão remota** – O sistema deve permitir gestão remota, de forma a facilitar a resolução de eventuais falhas à distância;
- **Instalar/reinicializar/parar/desinstalar** – O sistema deve ser capaz de instalar/reinicializar/parar/desinstalar serviços e componentes sem que o restante sistema seja reinicializado e, sempre que possível, sem provocar impacto noutros serviços/componentes do sistema;

- **Qualidade de Serviços** – Garantir a qualidade de todos os serviços, principalmente dos serviços de *streaming* de multimédia;
- **Multiplataforma** – O HG deve, se possível, suportar o *deployment* de serviços desenvolvidos sobre várias linguagens de programação;
- **Redes de nova geração** – A utilização de redes de nova geração é nuclear ao projeto LUL, sendo, portanto, fundamental que o sistema funcione segundo este paradigma.

Entre os requisitos técnicos listados destacam-se a necessidade de **interoperabilidade**, de **segurança**, de **monitorização**, de **modularidade** e de instalar, reinicializar, parar e desinstalar serviços em tempo de execução.

3.5.2. Requisitos de Utilizador do HG e Serviços

Tal como já foi referido, os utilizadores do sistema serão seniores, o que pode acarretar a dificuldades em interagir com novas tecnologias. Por esse motivo, os requisitos de utilizador são bastante importantes:

- **Personalização** – O sistema deve providenciar aos utilizadores a capacidade de configuração para que este responda mais adequadamente às necessidades do utilizador;
- **Interação explícita** – O sistema deve facilitar a interação explícita entre o sistema e o utilizador. O sistema deve igualmente ser capaz de comunicar de várias formas (através de voz, texto, gráficos, vídeo, etc.);
- **Adaptação** – O sistema deve ser capaz de se adaptar ao contexto, principalmente na camada de apresentação. Para isso, é necessário que o sistema tenha consciência do contexto nas várias camadas do sistema;
- **Simplicidade** – O sistema deve ser simples, escondendo a complexidade dos serviços dos seus utilizadores;
- **Embebido e distribuído** – O sistema e os seus componentes (por exemplo, dispositivos) devem ser não invasivos e invisíveis no ambiente da casa, de tal forma que passem despercebidos ao utilizador, não havendo o risco de o ambiente da habitação se tornar desconfortável. Por outro lado, o sistema e os seus componentes devem estar distribuídos para que possa responder em toda a casa de forma eficaz;
- **Dados relevantes** – Apenas os dados relevantes para a saúde do sénior devem ser transmitidos e acedidos pelo exterior do HSN. Estes dados devem estar de acordo com os princípios legais de privacidade;
- **Ser antecipatório** – O sistema deve, se possível, antecipar-se aos desejos do utilizador;
- **Eficazes** – Os novos serviços devem ser tão eficazes como a maneira tradicional de lidar com os problemas, embora estes serviços não devam substituir o tratamento pessoal;

- **Confirmação/verificação** – O sistema deve pedir confirmação do utilizador sempre que novas configurações ou novos serviços forem instalados de forma automática. Deve, ainda, permitir ao utilizador desligar/ligar um serviço em qualquer momento e informar o utilizador acerca dos dados que estão a ser acedidos para o exterior.

Entre os requisitos de utilizador listados destacam-se a necessidade de **adaptação**, de **simplicidade** e de **confirmação**.

3.6. Arquitetura do HG

O HG do projeto LUL utiliza uma abordagem orientada aos serviços, que foi igualmente aplicada aos restantes componentes do projeto, sendo a tendência dos sistemas de AAL. Sendo assim, a arquitetura do sistema do HG beneficia de todas as vantagens das arquiteturas SOA. Para além disso, a arquitetura do HG é do tipo com n-camadas.

A arquitetura está dividida em cinco camadas principais. A camada física, a camada de serviços, a camada de apresentação, a camada de qualidade de serviços e a camada de gestão de serviços.

Figura 22 – Home Gateway camadas

Três dessas camadas (a camada de apresentação, a camada de serviços e a camada física) estão representadas hierarquicamente, uma vez que a camada superior apenas pode interagir com a que está imediatamente a baixo e, por sua vez, a inferior apenas pode interagir com a que está imediatamente acima. Não existe a possibilidade de uma camada interagir com uma outra camada que não lhe esteja adjacente, ou seja, a camada de aplicação não vai interagir diretamente com a camada física. As restantes duas camadas estão representadas de forma diferente uma vez que interagem com as várias camadas (Figura 22).

A **camada física** contém os dispositivos do HS, as bases de dados, e respetivos interfaces de utilização. Esta camada interage diretamente com a camada de serviços e com a camada de qualidade de serviços.

A **camada de serviços** inclui todos os serviços do HG. Alguns serviços podem ser composição de outros serviços. Esta camada é a responsável por todo o modelo de negócio do sistema AAL e interage com todas as camadas do sistema.

A **camada de apresentação** é composta pelas aplicações que serão utilizadas pelo sénior. Estas aplicações devem ter especial atenção a aspetos de usabilidade e interface.

A **camada de qualidade de serviços** é responsável pela garantia de segurança no sistema. É nesta camada que estão implementadas soluções para aumentar a confidencialidade, integridade e disponibilidade dos serviços do sistema. É ainda responsável por analisar o desempenho e por escalar os recursos do HG, caso seja necessário.

A **camada de gestão de serviços** é responsável por disponibilizar novos serviços sempre que o utilizador o pretender ou sempre que um novo dispositivo é instalado na rede. Deve ainda disponibilizar todos os serviços públicos para o interior e exterior do HSN. Assim, as aplicações conseguem aceder aos serviços de forma mais fácil e eficaz.

3.7. Implementação da arquitetura do HG

O HG está implementado sobre a estrutura base do *Glassfish* que corre sobre a plataforma de serviços OSGi (que permite parar/atualizar/reiniciar/instalar/desinstalar serviços sem que o sistema tenha de reiniciar). Para providenciar a possibilidade de suportar várias plataformas de serviços, o HG utiliza o sistema operativo *MS Windows Server 2003*.

A Figura 23 pretende ilustrar os módulos principais e a respetiva camada onde estes se encontram. Cada módulo é disponibilizado na forma de *Web Service* e pode ser reutilizado por outras aplicações. A comunicação entre os módulos é feita por HTTP, mais concretamente REST e SOAP.

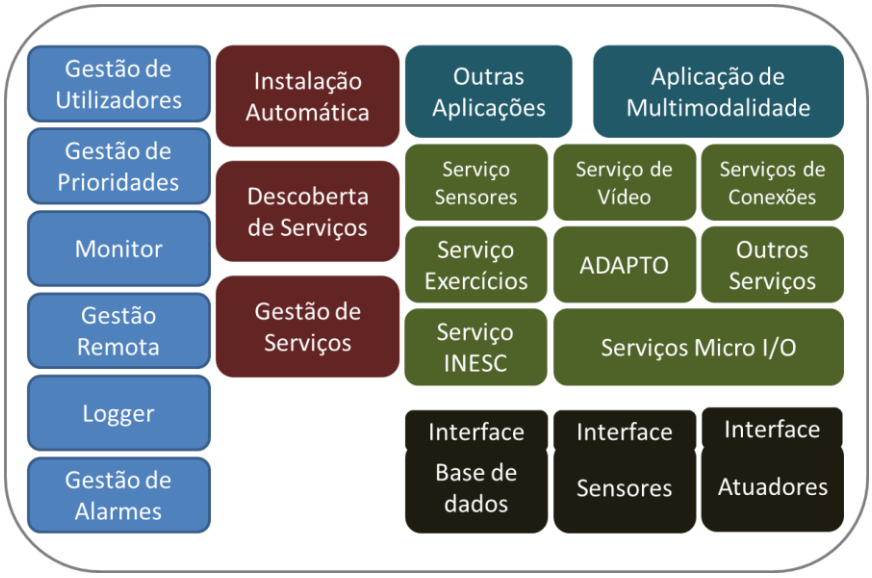


Figura 23 – Home Gateway - Módulos por camada

A camada física é essencialmente composta por três elementos: as bases de dados, os sensores e os atuadores. Estes têm interfaces de utilização que estão disponibilizados nesta camada. Os sensores são os elementos que irão capturar dados que serão utilizados por alguns serviços. Os atuadores executam determinadas tarefas de acordo com os pedidos dos serviços. Por sua vez, as bases de dados podem ser de vários tipos e são as responsáveis por guardar a informação do sistema de forma persistente.

A camada de aplicação é composta por um conjunto de aplicações. Atualmente existe apenas a aplicação de multimodalidade desenvolvida no âmbito do projeto LUL. Esta aplicação utiliza grande parte dos serviços que estão na camada de serviços do sistema.

As restantes três camadas são as mais importantes para este documento, sendo que é nelas que estão os serviços e os módulos do sistema do HG. Estas camadas e respetivos módulos serão descritos nas secções seguintes.

3.8. Camadas de gestão e de qualidade de serviços

Nesta secção estão descritos os módulos da camada de gestão de serviços e da camada de qualidade de serviços. É igualmente nesta secção, e ao longo da descrição das camadas, que se apresenta a explicação do modo como cada um destes módulos pode responder aos requisitos do HG.

3.8.1. Gestão de utilizadores

O módulo de gestão de utilizadores, tal como o nome indica, permite gerir utilizadores. Este módulo permite associar diferentes privilégios a diferentes utilizadores, sejam estes utilizadores no interior do HS ou no exterior.

O principal objetivo deste módulo é aumentar a privacidade dos habitantes, possibilitando ao habitante a configuração dos privilégios de acesso. Ao mesmo tempo, quando algum utilizador não autorizado tenta aceder a um recurso do sistema ou quando um utilizador autorizado tenta aceder a um recurso privado, este vai notificar o habitante, dando-lhe *feedback* e a oportunidade de barrar o acesso. Assim, o sénior sente-se mais à vontade com a tecnologia uma vez que tem controlo sobre a mesma.

No interior do HS, várias pessoas poderão ter acesso aos serviços. No entanto, em alguns cenários, pode não ser plausível que essas pessoas tenham acesso a todos os serviços. Um exemplo disso é um serviço capaz de lidar com informação sensível/privada de um dos habitantes da casa. Pode não fazer sentido que os restantes habitantes tenham acesso a essa informação. Outro exemplo é a manipulação de atuadores dentro do quarto do indivíduo sénior.

Para além de existir a necessidade de diferenciar seniores dentro da habitação é crucial que se diferenciem os utilizadores exteriores ao HS. Este módulo é responsável por gerir os acessos exteriores ao HG e seus serviços.

Em alguns casos o sistema de gestão de utilizadores pode enviar alertas para o módulo de alarmes, para que os seniores do HS possam dar permissões especiais a um determinado utilizador. Para serviços mais privados, mesmo que o utilizador tenha permissões, pode ser enviado um alarme ao habitante da casa com a informação de que há dados privados que estão a ser acedidos.

Para além disso, este módulo contém operações de configurações de privilégios e operações para criar/ler/remover/atualizar utilizadores. As regras associadas a cada *role* são definidas neste módulo.

Todos os acessos são registados no sistema de registos (*Logger*) da aplicação para que mais tarde se possa verificar se ocorreu alguma anomalia no sistema. A informação dos utilizadores é guardada persistentemente recorrendo a ficheiros XML para não se criar dependências tecnológicas associadas a sistemas de gestão de base de dados (SGBD) e para que o sistema não necessite de grandes especificações técnicas (uma vez que os SGBD necessitam de bastantes recursos). O acesso a ficheiros é, normalmente, mais lento do que o acesso a informação de SGBD. No entanto, a quantidade de acessos persistentes aos ficheiros não será muito elevada. Mesmo assim, de forma a colmatar o problema associado à velocidade de acesso, foi implementado um sistema de *cache* na função de autenticação que depende do tempo e da data de alteração do ficheiro. A *password* do utilizador é encriptada recorrendo a uma *one-way function (hash function)*[86] de forma a aumentar a segurança do sistema (Anexo E – Serviços do HG).

3.8.2. Gestão de prioridades

Este módulo serve de auxílio aos restantes módulos ajudando a aumentar a segurança do sistema. Por exemplo, acedendo a este módulo, o módulo de gestão de privilégios consegue perceber quais os serviços que requerem mais privacidade, e é também acendendo a este módulo que o monitor vai decidir quais os serviços que pode desligar quando não tem recursos suficientes disponíveis.

A prioridade é uma medida baseada nos três principais fatores de segurança: privacidade, disponibilidade e integridade. Estes valores são guardados de forma separada, podendo obter-se um dos seguintes valores para cada fator: 0 – baixo; 1 – médio; 2 – alto. É, ainda, possível obter o valor de prioridade que é uma combinação destes fatores, sendo que o valor mais alto de prioridade é igual a 6, e o mais baixo igual a 0.

O módulo de gestão de prioridades permite associar aos vários serviços/módulos do sistema uma prioridade. Cada serviço deve estar associado a uma prioridade para que o sistema possa atuar segundo essa prioridade. Quando um serviço é automaticamente instalado é necessário que na descrição do mesmo já esteja associada a prioridade. Se o serviço for instalado manualmente o utilizador tem de aceder a este módulo e configurar a prioridade do novo serviço.

Este módulo é, portanto, composto por uma lista de serviços, com respetivas prioridades (que estão guardados persistentemente utilizando ficheiros XML). Um serviço que não tenha sido configurado (por não ter a descrição ou porque foi instalado manualmente e ninguém o configurou) fica automaticamente com o nível mais baixo de prioridade (Anexo E – Serviços do HG).

3.8.3. Gestão remota

O módulo de gestão remota está disponível para o exterior e permite a um utilizador autorizado efetuar operações de administração do sistema, como desligar serviços, instalar serviços, alterar configurações, adicionar utilizadores, entre outras. Este serviço é um conjunto variado de serviços do sistema e está disponível para o exterior da HSN. Idealmente está registado no LULMS de forma a facilitar a obtenção do URL de acesso aos administradores.

É importante referenciar que os servidores têm igualmente a capacidade de acesso remoto, que deve ser utilizada em cooperação com este serviço.

Este serviço pode igualmente ser acedido dentro da casa, permitindo efetuar as mesmas operações.

A vantagem deste serviço é providenciar mecanismos extra de controlo remoto, permitindo assim a manutenção e administração do sistema à distância (Anexo E – Serviços do HG).

3.8.4. Logger

Este módulo é responsável por registar todas as ações efetuadas no sistema. Todas as aplicações/serviços/módulos devem fazer registos neste sistema. A informação é guardada de forma persistente recorrendo a ficheiros XML (Anexo E – Serviços do HG).
A grande vantagem deste serviço é poder analisar-se o comportamento do sistema, verificar os acontecimentos anteriores a uma falha, consultar acessos, erros, entre outros.

3.8.5. Gestão de Alarme

A vantagem deste módulo é o facto de providenciar um sistema comum de comunicação, permitindo comunicação assíncrona e síncrona entre o sistema e o utilizador.
O módulo de gestão de alarmes permite guardar alarmes para o utilizador. O objetivo deste módulo é que todos os serviços coloquem os seus alarmes, incluindo verificações de ações.
Os alarmes podem ser colocados por qualquer aplicação ou por um utilizador. Podem ser colocados para serem apresentados no momento ou para serem apresentados numa determinada data/hora. Os alarmes podem, ainda, estar associados a um tipo de alarme, a uma pessoa ou a uma aplicação. A função do módulo é enviar os alarmes a serem apresentados ao longo do tempo, sendo responsabilidade da aplicação a obtenção do contexto para decidir se deve mostrar o alarme (uma vez que só faz sentido mostrar o alarme se/quando estiver alguém a vê-lo).
Idealmente, todas as aplicações capazes de receber/apresentar alarmes devem registar-se neste serviço, especificando (ou não) qual o tipo de alarme a receber. Caso não exista a especificação, a aplicação receberá todo o tipo de alarmes.
Os alarmes são guardados persistentemente em ficheiros XML até serem enviados. A lista de aplicações registadas para receber alarmes é igualmente guardada da mesma forma. Este módulo é composto por um processo que envia um alarme no momento em que este deve ser enviado (Anexo E – Serviços do HG).

3.8.6. Instalação Automática

O módulo de instalação automática tem a responsabilidade de instalar, desinstalar, atualizar e configurar serviços novos de forma automática. Este módulo, com a ajuda do serviço de obtenção de serviços (serviços de distribuição de JARs) do LULMS, vai obter os serviços que estão disponíveis para uma habitação com determinados dispositivos.
No caso de se verificar um novo dispositivo instalado na HSN, é feito um pedido ao LULMS para obter os serviços correspondentes. Com essa informação, o módulo vai comparar os serviços obtidos com os serviços existentes no HG. Se algum dos serviços ainda não estiver instalado, o HG

instala-o automaticamente no sistema. Para o caso de novos serviços e de atualizações, o módulo vai obter, de forma frequente (por exemplo diariamente; isto dependerá da frequência de instalação de novos serviços), a lista de todos os serviços para cada um dos dispositivos, instalando os novos serviços e atualizando aqueles que necessitem de atualização.

Este módulo é composto por processos que verificam de forma constante se existem novos dispositivos na HSN, se existem novos serviços para os dispositivos instalados na HSN e se existem atualizações para os serviços já instalados na HG. Assim, sempre que um novo dispositivo é instalado ou sempre que um novo serviço é disponibilizado para um dispositivo, o HG conseguirá obter esse serviço de forma automática (Anexo E – Serviços do HG).

Os dipositivos estão divididos por um conjunto de categorias. Estas categorias são importantes uma vez que pode haver um serviço que só funcione com vídeo mas que não dependa de nenhuma câmara, ou seja, que seja totalmente independente do *hardware*. No entanto, é necessário haver serviços específicos da câmara que permitem o controlo da mesma.

Este módulo proporciona várias vantagens:

- Como funciona de forma automática torna a integração de novo *hardware* no sistema mais simples e fácil;
- Como utiliza o LULMS para obter os serviços permite a instalação distribuída e massiva de um serviço, sendo apenas necessário colocar esse serviço no repositório do LULMS;
- Aumenta a interoperabilidade do sistema. Os fornecedores de *hardware* podem colocar serviços que utilizam o seu *hardware* de forma simples e eficiente no LULMS. Quando instalarem o novo *hardware* no sistema ele fica automaticamente disponível para utilização na forma de serviço;
- Permite atualização distribuída dos serviços, ou seja, sempre que um serviço é atualizado torna-se apenas necessário atualizar o LULMS. O HG responsabiliza-se pela atualização do serviço em cada nó da rede.

3.8.7. Gestão de serviços

Este módulo permite aos utilizadores ativar/desativar serviços em qualquer momento. A vantagem deste módulo é permitir ao utilizador desligar um serviço sempre que pretende e permitir-lhe fazer configurações sobre os serviços, dando-lhe assim mais controlo sobre o sistema. Permite igualmente alterar as configurações de um determinado serviço ou mudar o seu grau de prioridade (Anexo E – Serviços do HG).

3.8.8. Descoberta de serviços

Este módulo é responsável por fornecer às aplicações/serviços a informação dos serviços disponíveis. Assim, torna-se fácil para uma aplicação/serviço obter um determinado serviço do sistema. Este módulo aumenta a flexibilidade do sistema uma vez que permite obter a lista de serviços disponíveis de forma dinâmica.

A lista de serviços está guardada num UDDI interno do sistema e é acedida por todos os módulos da camada de gestão de serviços e também pelo monitor. Se não existir um sistema de UDDI configurado, este serviço guarda a lista de serviços e respetivas informações persistentemente num ficheiro XML (Anexo E – Serviços do HG).

3.9. Camada de Serviços

Nesta secção são descritos e explicados alguns dos serviços implementados no âmbito do projeto LUL. Todos os serviços estão alojados no HG e respondem a requisitos que foram impostos pelos parceiros responsáveis por desenvolver as aplicações da camada de apresentação do projeto. Estes serviços são ao mesmo tempo uma ferramenta de teste da arquitetura implementada.

No âmbito do projeto desenvolvi o serviço de sensores (adaptado de um serviço já existente), desenvolvi o serviço de câmara de vídeo, o serviço de conexões e o serviço de exercícios (adaptado de um serviço já existentes). Os restantes serviços desta camada foram desenvolvidos por outros membros do projeto.

Por questões procedimentais e de qualidade de serviços, todos os serviços desenvolvidos têm como resposta um objeto do mesmo tipo, chamado *Operation Status*. Este objeto é composto por três elementos: sucesso, resultado e mensagem de erro. Assim, sempre que um método de um serviço termina com sucesso, o elemento sucesso apresenta o valor verdadeiro, e caso exista resultado, esse resultado é preenchido no parâmetro resultado. Caso o método falhe, o resultado do parâmetro sucesso apresenta valor falso e é associado ao parâmetro mensagem de erro a mensagem que explica o motivo de a operação ter falhado. Este procedimento ajuda o *developer* de aplicações/serviços a perceber quando um serviço corre conforme previsto ou quando contém um erro. Nos casos de existir um erro, o *developer* tem informação para lidar com ele, sem que este tenha de ser propagado para o exterior do serviço.

3.9.1. Serviço de Conexão

O serviço de gestão de conexões faz a ligação entre o HS e o LULMS (Figura 24). O principal objetivo do serviço é encapsular todos os serviços do LULMS. Assim, não existe a necessidade de se conhecer o endereço do LULMS nem dos seus serviços. Este serviço é utilizado pelas aplicações

dentro do HS para efetuar todas as operações no LULMS: para efetuar autenticação com o LULMS, para associar os recursos que estão a ser utilizados, e para gerir as informações especificas de conexão do HG (Anexo C – Serviço de Conexão).

Com a utilização do módulo automático e de descoberta de serviços, caso o endereço do LULMS mude, é apenas necessário atualizar este serviço no LULMS, não sendo posta em risco a utilização do LULMS por parte dos outros serviços/aplicações e evitando qualquer tipo de modificação nesses serviços/aplicações.

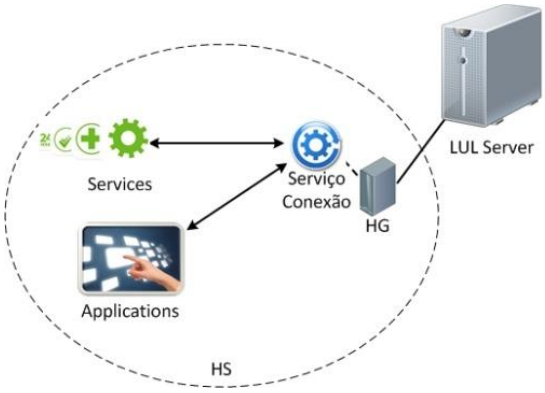


Figura 24 – Serviço de Conexão

3.9.2. Serviço de Sensores

Este serviço tem como principal objetivo disponibilizar para o exterior/interior do HS dados em tempo real dos sensores, podendo assim utilizar-se esses dados para criar informação relevante para as aplicações. Este serviço contém dados de sensores, que são enviados por determinados serviços que os obtêm e, em alguns casos, os tratam e transformam. Por este serviço poder ter dados muito sensíveis (principalmente quando estamos a falar de dados clínicos) é crucial manter a privacidade, integridade e disponibilidade dos mesmos, sendo considerado um serviço de alta prioridade (valor seis).

Este serviço tem a particularidade de pretender ser universal, ou seja, servir para qualquer dispositivo/*hardware*/fornecedor. Assim, se um fornecedor de um determinado sensor pretender, pode criar um serviço que coloque os dados do seu sensor neste serviço (tratados ou não), aumentando assim a flexibilidade e interoperabilidade do sistema, uma vez que remove todas as dependências relacionadas com fornecedores. Outra potencialidade deste serviço é poder ter várias fontes de dados a inserir dados para o mesmo propósito, por exemplo, dados de um sensor que avalia o batimento cardíaco e de um robot que obtém à distância a mesma informação, aumentando assim a fiabilidade e redundância de dados do sistema, criando tolerância a falhas (já que existe redundância de dados).

Sendo o principal objetivo deste serviço a obtenção de dados em tempo real, o armazenamento de dados não é feito persistentemente no sistema, obrigando assim cada canal do serviço a conter um número máximo de dados. Cada sensor/serviço pode colocar uma determinada quantidade de informação/dados num determinado canal, permitindo, que se faça uma separação da informação/dados de cada serviço. Como o serviço aceita um número limitado de dados que pode ser configurado manual ou automaticamente pelo HG, é possível a controlar a quantidade de dados. Por outro lado, os dados mais antigos serão automaticamente substituídos pelos dados mais recentes.

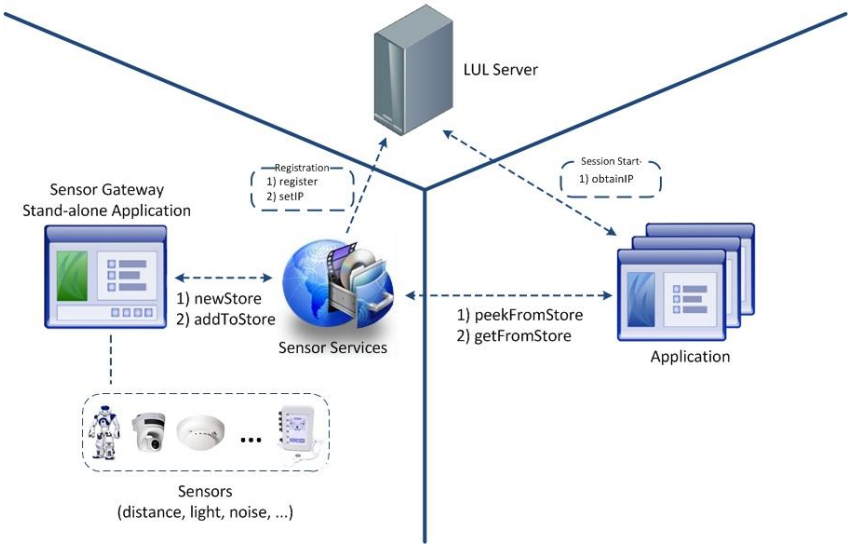


Figura 25 – Serviço de Sensores – Comunicação externa

A Figura 25 representa, de forma geral, a comunicação externa com o serviço de sensores. Esta encontra-se subdivida nas três zonas do projeto LUL (LULMS, HPS e HS). Primeiramente, é necessário fazer o registo do serviço de sensores no LULMS. Aquando esse registo, deve ser dada a informação sobre quais os sensores que estão a enviar informação/dados para o serviço de sensores, e, se possível, os respetivos canais. Depois de devidamente registado, o serviço pode ser acedido externamente por uma aplicação (desde que esta tenha autorização para isso). Para estabelecer a ligação com o serviço, a aplicação necessita de obter toda a informação sobre o mesmo (como, por exemplo, o IP). Depois de passadas estas fases, a aplicação vai comunicar diretamente com o serviço (que está no HG), podendo obter todos os dados que lhe são úteis.

Como especificado acima, o serviço de sensores é um serviço genérico que pretende abstrair o acesso a todos os sensores. Para isso, é necessário que cada sensor vá atualizando os seus dados, inserindo-os no serviço. Na Figura 25 pode verificar-se a existência de uma aplicação que comunica diretamente com os sensores e que insere os dados do mesmo no serviço.

A comunicação interna funciona de igual forma, não existindo, no entanto, a necessidade de acesso ao LULMS, sendo que a aplicação (caso tenha autorização) acede diretamente ao serviço. Para

obter informação sobre o serviço, a aplicação deve aceder ao registo de serviços do HG e obter o serviço de forma autónoma (Anexo B – Serviço de Sensores).

3.9.3. Serviço de Vídeo

O serviço de vídeo tem como principal objetivo disponibilizar para o exterior/interior do HS uma câmara de vídeo e som, podendo desta forma visualizar-se o interior da habitação (mais concretamente o paciente) de forma autónoma, controlada e segura. Por esse motivo, o serviço deve ser capaz de aceder a pelo menos uma câmara de vídeo localizada na HSN, deve providenciar uma comunicação segura e permitir executar ações de controlo da câmara (por exemplo, ampliar ou mover a câmara para os lados). Por questões de privacidade torna-se muito importante que este serviço seja extremamente seguro, e o seu acesso muito restrito.

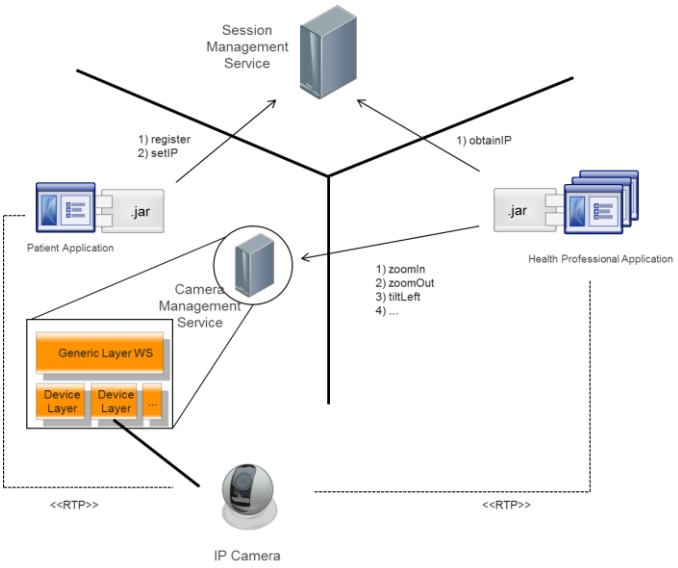


Figura 26 – Serviço de Vídeo - Comunicação externa

Este serviço segue o paradigma *Device as a Service*. Isto significa, que todas as funcionalidades do dispositivo são disponibilizadas como serviço.

Para aceder à câmara de vídeo a partir do exterior do HS é necessário comunicar com o LULMS e, portanto, com o serviço que permite gerir e visualizar recursos do mesmo. Já o acesso interno (dentro da HSN) é feito de forma mais facilitada, uma vez que é necessário obter o serviço, utilizando o módulo de descoberta de serviços, e utilizá-lo.

No que diz respeito à comunicação externa (Figura 26), depois de obtidas algumas informações, a aplicação localizada no HPS passa a poder comunicar diretamente com o serviço da câmara de vídeo que está disponível no HG, podendo assim controlá-la sem qualquer restrição (desde que tenha privilégios para isso e que o *hardware* o permita). A partir desse momento passa a ser igualmente possível fazer *streaming* de vídeo utilizando o protocolo RTCP [64], que é feito

diretamente entre a câmara e a aplicação. Este protocolo irá garantir a qualidade de transferência de vídeo e voz criando, assim, uma comunicação síncrona eficaz (Anexo A – Serviço de Vídeo).

3.9.4. Serviço de Exercícios

Este serviço permite a adição de uma lista de exercícios, que são posteriormente carregados pelas aplicações, colocar os exercícios a correr (*play*) ou parar a apresentação dos exercícios (*stop*). Permite ainda avançar exercícios à frente ou ir para uma posição da lista, começando a execução a partir dessa posição.

Com este serviço, o especialista de saúde pode colocar e controlar a realização de exercícios e, ao mesmo tempo, utilizar serviços para obter dados clínicos em tempo real (Anexo D – Serviço de Exercícios).

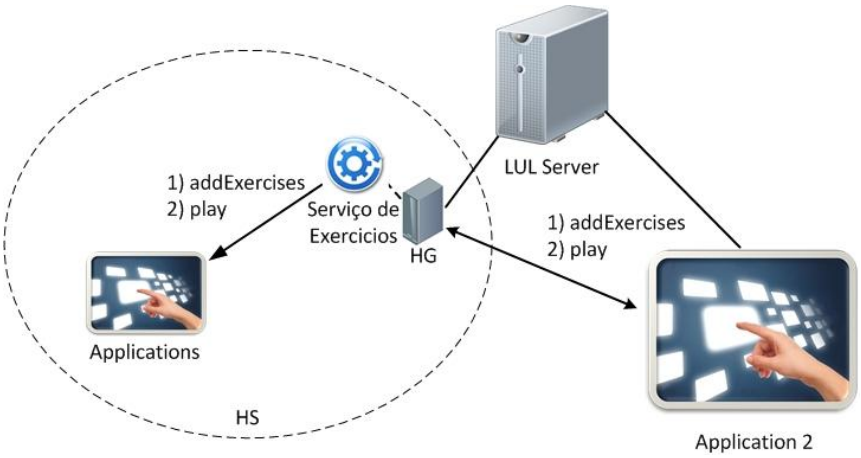


Figura 27 – Serviço de Exercícios

Este serviço segue o paradigma de *Application as a Service*. O objetivo deste serviço é permitir a um utilizador externo à HS (mas com privilégios) gerir a apresentação de exercícios do sénior.

Este serviço tem a função de cliente e de servidor. Ao analisar a Figura 27 pode verificar-se que o especialista de saúde comunica com o serviço que, posteriormente, vai comunicar com uma ou várias aplicações via HTTP.

3.9.5. Serviços desenvolvidos por parceiros

3.9.5.1. Serviços de automação desenvolvidos pela Micro I/O

Os serviços da Micro I/O estão focados no controlo de sensores e atuadores dentro da HSN. A Figura 28 corresponde à arquitetura interna dos serviços desenvolvidos pela Micro I/O.

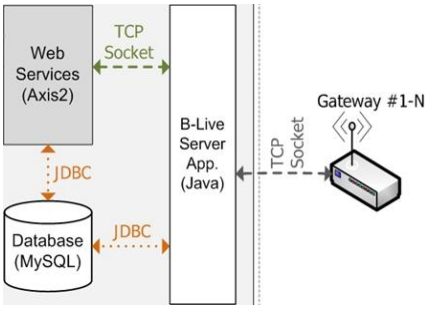


Figura 28 – Serviços Micro I/O

A comunicação entre os *gateways* e o "*Server App*" é bilateral e, portanto, esta deverá permanecer sempre ativa. A comunicação é feita por *HTTP* utilizando SOAP. A comunicação com a base de dados é conseguida através da ferramenta JDBC. Para além disso, o sistema de gestão de base de dados utilizado é o *MySQL*. O Serviço regista a informação de todos os sensores e atuadores presentes no sistema B-Live. É importante referir que os serviços estão implementados sobre *stack Axis2*.

A Micro I/O elaborou um conjunto variado de serviços:

- *SectorService* - serve para gerir os vários setores da casa;
- *RoomService* - permite obter os quartos da HS, adicionar nome aos quartos e remover quartos;
- *DeviceService* - permite obter os dispositivos/sensores/atuadores localizados na HSN, obter o nome de um dispositivo/sensor/atuador, obter o estado de bateria dos dispositivos/sensores/atuadores e obter os dados dos dispositivos/sensores/atuadores;
- *SiteService* - permite gerir a HS, ou seja, adicionar setores, quartos e dispositivos;
- *LampService* - serviço de atuadores para controlar lâmpadas, que permite ligar, desligar ou mudar a intensidade de uma lâmpada;
- *DoorService* - serviço de atuadores que permite controlar as portas/janelas (abrir, verificar se está aberta e fechar);
- *OutletService* - serviço de atuadores que permite acender a apagar tomadas/eletrodomésticos;
- *CurrentSensingService* - serviço que permite obter dados relacionados com os sensores de corrente de tomadas;
- *DistanceSensingService* - serviço que permite obter dados de sensores de distância;
- *TemperatureSensingService* - serviço que permite obter dados de sensores de temperatura;
- *MovementDetectionService* - serviço que permite obter dados de um sensor de movimento;
- *SmokeService* - serviço que permite obter dados de um sensor de fumo;
- *FurnitureSensorService* - é um serviço que permite obter dados de um sensor de pressão;

- *LighSensingService* - serviço que permite obter dados de um sensor luminosidade;
- *WaterService* - serviço que permite obter dados de sensores que detetam se uma torneira está aberta, ou se existe caudal de água a passar num determinado local;
- *BuzzerService* - serviço que permite listar e reproduzir musica;
- *AirConditionerService* - serviço que permite controlar um dispositivo de ar condicionado;
- *LocalisationService* - serviço que permite obter a posição dos habitantes da HS. Este serviço pode ser utilizado em cooperação com o serviço da INESC aumentando, assim, a fiabilidade do sistema;
- *RealTimeService* - serviço que obtém dados em tempo real de sensores de sinais vitais.

A Figura 29 exemplifica as ligações de alguns dos serviços descritos acima. Demonstra ainda que o HG é o local onde todos esses serviços estarão alojados.

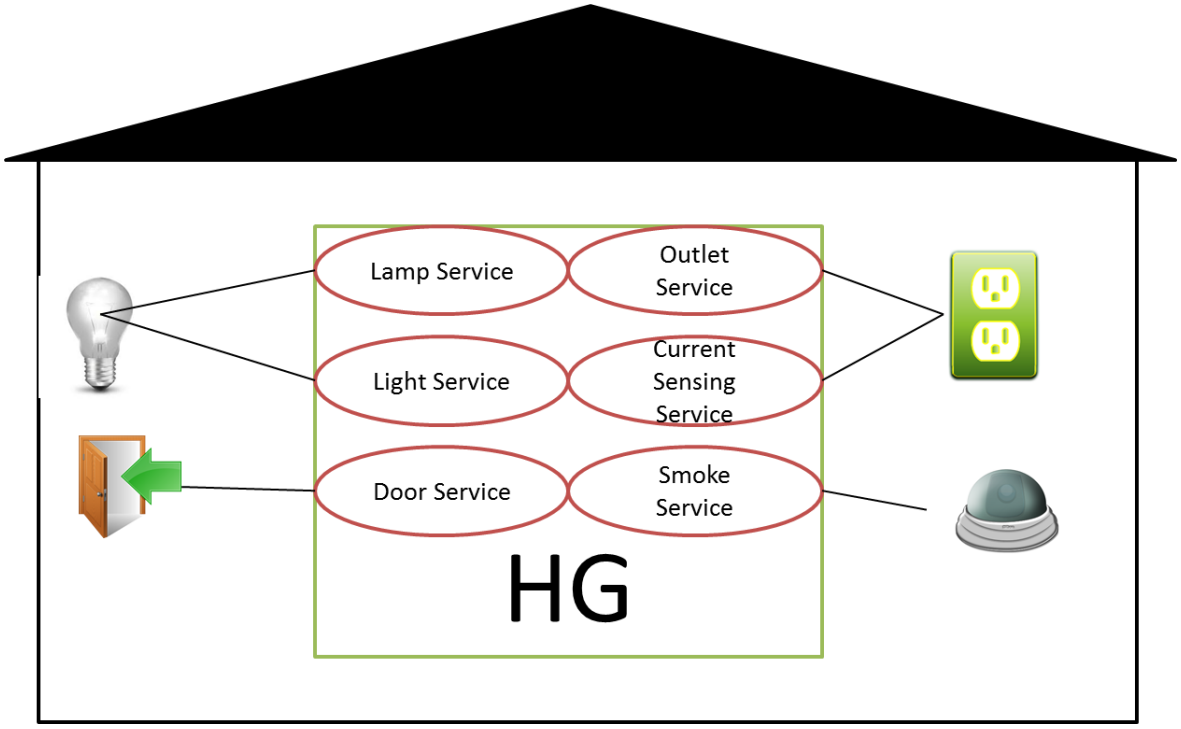


Figura 29 - Serviços Micro I/O - Exemplos

Estes serviços devem ser colocados no LULMS para que, quando ocorrer a instalação dos dispositivos/atuadores/sensores no HS, o HG seja capaz de obter os serviços e instalá-los de forma automática.

Estes serviços estão desenvolvidos sobre *Axis2* pelo que houve a necessidade de instalar o serviço no servidor. Foi igualmente essencial a configuração da base de dados necessária para colocar os serviços a correr (o que obrigou à instalação do MySQL). Num cenário ideal não deveria haver dependências nos serviços, como é o caso do sistema de gestão de base de dados. No entanto, caso

isso aconteça, os *developers* devem instalar as dependências dos seus serviços quando instalam os dispositivos na HS.

3.9.5.2. Serviços de localização desenvolvidos pela INESC

O serviço desenvolvido pelo parceiro INESC do projeto LUL permite obter o contexto do utilizador a partir de câmaras de vídeo. Este serviço será utilizado, em conjunto com o serviço AdaptO[8], pelas aplicações, de forma a adaptarem-se ao contexto dos utilizadores.

O serviço tem a capacidade de:

- Obter a posição do habitante dentro do HS (pode ser utilizado pelas aplicações para decidirem se faz sentido estarem ativas);
- Obter a distância entre um habitante e um objeto da HS (por exemplo, entre um sénior e uma televisão, providenciando assim às aplicações a funcionalidade de aumentarem o tamanho da fonte caso os sénior esteja demasiado afastado);
- Obter um número total de pessoas na HS (que pode ser utilizado para deteção de intrusos), de obter a informação de uma divisão e de pessoas nessa divisão (que pode ser utilizado quando existe mais de um habitante na HS);
- Obter a trajetória de um habitante (que pode ser utilizado por um robot para seguir o habitante).

O serviço foi desenvolvido em *.NET*, o que aumentou a complexidade do problema associado ao HG, já que houve a necessidade de colocar o*.NET* e o *Java* a funcionarem em conjunto. No entanto, como a arquitetura foi desenvolvida de forma modular, orientada a serviços, e a contar com multiplataforma, não existem limitações na utilização e publicação do serviço. Tal como os outros serviços, este pode ser obtido utilizando o serviço de descoberta de serviços descrito acima. É apenas necessário registar os serviços desenvolvidos em *.NET* na UDDI interna.

3.9.6. Outros serviços

Para além dos serviços implementados no âmbito deste trabalho, outros serviços foram implementados por parceiros do projeto. Por motivos diversos estes serviços não seguem todas as recomendações descritas no início deste capítulo. Estes serviços estarão igualmente alojados no HS, mais concretamente no HG.. Entre esses serviços destacam-se o AdaptO [87], o serviço do robot [88] e o PLA [89].

O AdaptO é um serviço desenvolvido pela Universidade de Aveiro para apoiar seniores com problemas de audição ou de visão. Este serviço é capaz de se adaptar às necessidades do utilizador de forma automática, proporcionando valores características de volume de som e de fonte do texto.

[8] Ver secção 3.9.6.

Assim, as aplicações poderão utilizar este serviço para adaptar as suas características de interface. Outra aplicação viável para este serviço é providenciar uma adaptação de interface à medida que o utilizador se move, por exemplo, quanto mais afastado está do ecrã maior é o volume do som [87].
O serviço de robot é um serviço desenvolvido pela Universidade de Aveiro para melhorar a qualidade de vida dos seniores através da utilização de um robot. O robot é disponibilizado, assim como os restantes recursos da HS, na forma de serviço. O robot providencia segurança aos utilizadores e ao ambiente da habitação, é capaz de evitar obstáculos parados ou que se movimentem, fornece medicamentos ou lembra o sénior de que os tem de tomar, transmite vídeo e som, recebe informação de sensores externos (através da utilização do serviço de sensores), estaciona numa estação de carregamento quando a bateria está em baixo, executa ordens externas, segue o sénior e responde quando é chamado [88].
O PLA é um serviço/aplicação multimodal e multi-plataforma (*desktop* ou *mobile*) que integra o acesso a *e-mail* e agenda, permite a realização de conferências de áudio e vídeo e integra serviços multimédia sociais. O PLA contém uma interface desenvolvida especialmente para pessoas com dificuldades de mobilidade.

3.10. Integração do Home Gateway na habitação

De forma a aumentar a integração do sistema no ambiente da habitação, foi feita investigação no sentido de obter soluções que permitem instalar todo o sistema do HG de forma rápida e simples e de obter solução para integrar a HG com os equipamentos de operadores de redes de nova geração.
O processo de instalação automática consiste na utilização de um instalador *Java* que instalará, de forma automática, os serviços-base do sistema e que os colocará em execução.
Depois de os vários módulos estarem a funcionar no sistema e, mais especificamente, depois de o módulo de instalação automática da camada de gestão de serviço estar a funcionar, o sistema será capaz de, automaticamente, obter os dispositivos e instalar os vários serviços de acordo com as necessidades do utilizador.
Uma vez que o HG é constituído exclusivamente por serviços, é possível colocar todo o sistema a correr noutros dispositivos, como é o caso das *set-top-box* que recebem informação proveniente de uma fonte exterior (mais concretamente os provedores de serviços de televisão) [90] ou das *Xbox* que permitem a utilização de serviços externos para apresentar dados e a utilização de *streaming* de dados [91] .

4. Testes e Resultados

Neste capítulo é apresentada uma possível configuração do ambiente do HG. Sobre essa configuração foram efetuados testes ao HG e aos seus serviços e procedeu-se à realização de testes unitários e testes de carga. São, também, apresentados os resultados mais importantes desses testes. Ainda neste capítulo, apresenta-se a aplicação de tele-reabiliatação, que é utilizada para testar a arquitetura e os serviços do sistema. Essa apresentação demonstra o modo como o HG facilita a integração e utilização dos serviços. Por fim, é apresentada a análise do teste integrado no projeto LUL. Essa análise inclui a análise de tipo de ligações, de equipamentos e de locais que devem ser testados de forma a obter um conjunto de resultados eficientes.

4.1. Configuração de Teste

Existem vários tipos de configurações possíveis do ambiente onde o HG está inserido. Nesta secção é apresentado em detalhe um exemplo de configuração desse ambiente. Inicialmente relata-se o cenário de utilização do HG, referindo os dispositivos que o rodeiam e as suas ligações. De seguida, apresentam-se as características da máquina onde está a correr o HG. Por fim, apresentam-se as duas aplicações utilizadas para testar o HG e os seus serviços. A configuração detalhada representa o cenário real que foi utilizado para a realização dos testes descritos na secção seguinte.

4.1.1. Cenário de utilização

Na Figura 30 está esquematizado o cenário lógico de utilização do HG que foi utilizado para testes. No lado direito da figura estão representados os nós correspondentes ao HPS. No lado esquerdo da figura está representado o HS e os seus dispositivos.

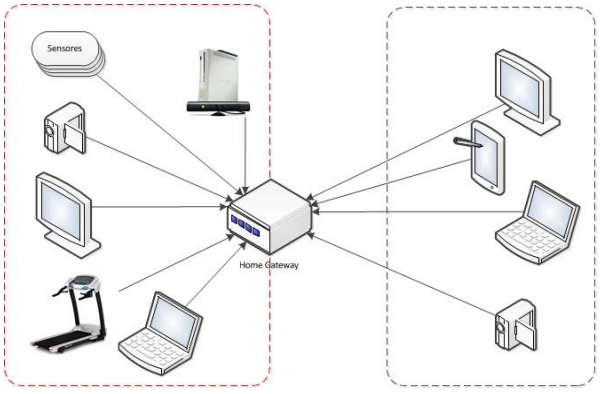


Figura 30 - Exemplo de utilização

Como se pode verificar na figura, quando os dois nós (HS e HPS) estão em comunicação, toda a informação é trocada tendo como intermediário o HG.

Os elementos (aplicações, dispositivos, entre outros) do HS começam a comunicação com o HG enviando-lhe dados importantes. No entanto, em alguns casos, o HG pode ser o responsável por pedir a informação a esses dispositivos ou enviar-lhes instruções. Um exemplo desta última situação é o serviço de câmara de vídeo que envia comandos de controlo para a respetiva câmara.
Os nós HPS comunicam diretamente com o HG, que posteriormente lhes fornece a informação que pretendem. Estes são os responsáveis por pedir a informação, sendo que o HG nunca inicia a comunicação com estes. Num cenário normal existem vários HPS a aceder a um único HS, pelo que é necessário garantir a disponibilidade de serviços para todos eles. É por esse motivo que estão representadas várias ligações do lado direito da Figura 30.

4.1.2. Configurações do Home Gateway

A máquina que possui o sistema do HG contém o *MS Windows Server 2003* instalado. Este inclui o *Internet Information Services*, que é utilizado para os serviços desenvolvidos em *.NET*, o *Glassfish*, que utiliza o modelo OSGi onde estão os módulos do HG, e também alguns serviços. Em execução no *Glassfish* está a *stack* de serviço *Axis2* que contém, essencialmente, os serviços desenvolvidos pela Micro I/O. Para além disso, a máquina tem instalada a Microsoft UDDI e o sistema de gestão de base de dados MySQL que é utilizado para guardar informação relativa aos serviços da Micro I/O de forma persistente.
O HG e os seus serviços foram testados numa máquina com os requisitos mínimos de funcionamento. Esta máquina tem as seguintes configurações:

- 512 MB de memória RAM;
- Processador Intel Pentium 4 CPU 2800 GHz;
- Disco rígido mínimo 3 GB.

4.1.3. Ferramentas de teste

Todos os módulos (que estão disponíveis como serviço) e todos os serviços da camada de serviços da arquitetura foram testados recorrendo às ferramentas *soapUI* e *loadUI*. A escolha destas ferramentas deveu-se ao facto de estas permitirem de forma rápida, fácil e eficaz realizar vários tipos de testes a serviços. Por outro lado, devido à sua popularidade e tempo de vida (desde 2005) são ferramentas muito testadas e, portanto, bastante fiáveis.
O *soapUI* é uma ferramenta gratuita que permite fazer testes funcionais a serviços e oferece uma interface simples, permitindo executar um conjunto de métodos de um ou mais serviços de forma rápida. Permite, ainda, fazer pequenos testes de carga a cada serviço [92].
O *loadUI* é uma ferramenta gratuita que faz testes de carga. Esta ferramenta oferece uma interface bastante intuitiva e permite criar, configurar e redistribuir os testes de forma interativa e em tempo-

real. Providencia ainda uma cobertura total de testes e suporta os protocolos e tecnologias padronizadas. Esta gera cargas escaláveis, reais, e em grande escala para qualquer local ou computador [93].

4.2. Testes

Partindo do princípio que o objetivo desta dissertação é criar condições para a utilização rápida e eficaz de serviços, em muitos casos não era requerida a criação de interfaces, mas sim o teste dos serviços e o teste aos limites de carga do servidor. Por esse motivo os serviços foram testados recorrendo a ferramentas que permitem testar o funcionamento dos mesmos. Foram igualmente testados recorrendo-se a ferramentas que permitem sobrecarregar os serviços de forma a averiguar comportamentos anormais.

A realização de testes foi dividida em três fases diferentes. Inicialmente foram feitos testes unitários de forma a testar cada serviço e respetivos métodos individualmente. Quando todos os serviços passaram com sucesso nos testes unitários, passou-se à segunda fase de testes: os testes integrados. Os testes integrados combinam os vários módulos/serviços e testam-nos em grupo. Em alguns casos os testes integrados são cruciais para o bom funcionamento de um serviço, uma vez que existem serviços que dependem de outros serviços.

Na terceira fase foram efetuados testes de carga. Os testes de carga unitários, onde cada serviço e respetivos métodos foram submetidos a uma carga de pedidos anormal, permitem averiguar problemas nos serviços. Por fim, foi feito um teste de carga à máquina onde se simulou a existência de vários nós HPS que utilizaram vários serviços num curto intervalo de tempo.

4.2.1. Testes unitários e integrados

Com o *soapUI* foi possível testar, não só se os serviços e respetivos métodos estavam a funcionar corretamente, mas também verificar se estes estavam a responder em tempo útil.

Para cada método de cada serviço foram criados pedidos em XML (testes). Estes testes cobrem grande parte das possibilidades, ou seja, para cada método foram feitos testes onde acontece o esperado, foram testados os limites e foram testadas situações onde é suposto ocorrer erro. Este procedimento foi feito de forma iterativa repetindo os testes para cada método até que todos passassem com sucesso. Quando todos os serviços estavam a funcionar, foram feitos pequenos testes de carga de forma a averiguar os tempos de resposta dos mesmos. Este processo foi, igualmente, repetido de forma iterativa até que os tempos de resposta dos métodos de cada serviço fossem próximos dos aceitáveis.

De seguida, foram feitos testes de integração dos métodos que dependiam de outros serviços (por exemplo, os métodos que efetuam registos no *logger*).

Como exemplo são apresentados os resultados dos testes realizados ao serviço de exercícios explicado nas secções anteriores. Foi escolhido este serviço por ter sido o serviço que deu origem a mais problemas, e por ter sido aquele que proporcionou tempos de resposta superiores.

Começa-se por averiguar em que situações se pretende fazer os testes. Os testes feitos foram: (1) inserir todos os dados de forma correta sem que ocorra nenhuma situação anormal; (2) inserir os dados todos de forma correta mas inserir um *URL* de comunicação errado; (3) inserir dados de forma aleatória; (4) apagar dados quando estes não existem; (5) mandar pedidos de ações associadas aos exercícios (começar, parar e mudar de exercício) quando não existem exercícios ou quando não existe nenhum *URL* para fazer o pedido.

Como exemplo são mostrados os resultados para o primeiro teste listado acima. Na Figura 31 está representado o exemplo do pedido XML de inserção de três exercícios (do lado esquerdo) e a respetiva resposta (do lado direito).

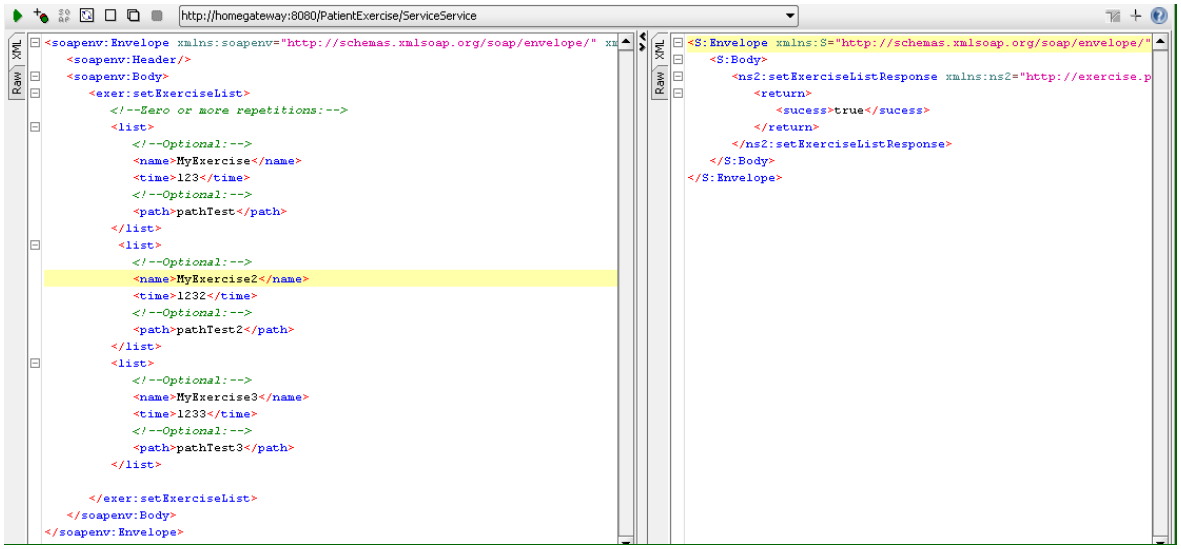


Figura 31 – SoapUI - Inserção de exercícios

A Figura 32 prova que todos os testes tiveram sucesso, ou seja, todas as operações estão a funcionar como previso. Se alguma das operações não tivesse chegado ao fim ou se não estivesse a verde isso significaria que ocorreu um erro inesperado.

Na Figura 32 é feito um teste de carga aos métodos de inserção de um *URL*, de inserção de exercícios e obtenção de exercícios. Como se pode verificar foram efetuados 112 pedidos a cada um dos métodos durante os 60 segundos de teste. Para o primeiro caso obteve-se um tempo médio de resposta de 502 milissegundos tendo sido transferidos 29224 *bytes* a uma taxa de 485 *bytes* por segundo. Para o segundo caso obteve-se o tempo de 1180 milissegundos tendo sido transferidos 30344 bytes a uma taxa de 504 bytes por segundo. Para o terceiro caso o tempo médio obtido foi de 211 milissegundos tendo sido transferidos 67088 *bytes* a uma taxa de 1114 *bytes* por segundo. Durante a execução dos testes nunca houve ocorrência de erros, o que é positivo. O tempo do

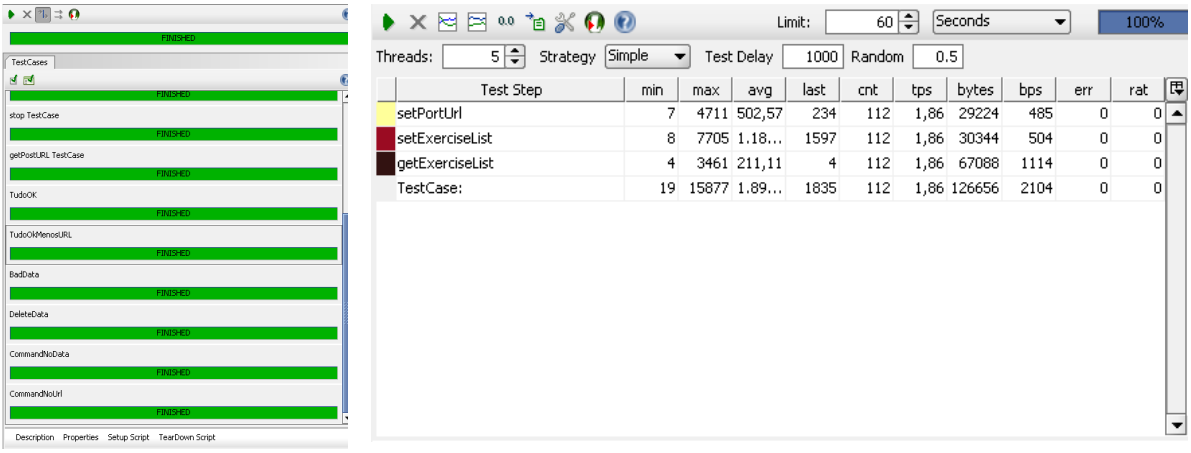


Figura 32 – SoapUI – Testes

primeiro teste foi razoável e o tempo do terceiro foi aceitável tendo em conta os dados transferidos. No entanto, o tempo do segundo teste foi elevado, o que é justificável, uma vez que este pedido depende também do tempo associado ao envio de mensagens POST para os URL listados.

A Tabela 1 apresenta os valores totais de utilização de todos os métodos dos serviços. Todos os serviços estão a funcionar como esperado, sem ocorrer nenhum erro. Os serviços com respostas mais lentas são: (1) o serviço de exercícios, o que se deve ao facto de estar a ser utilizado um endereço que não está dentro da rede, o que num caso real não irá acontecer, (2) o serviço de conexão, por depender da velocidade de resposta do LULMC, e por fim (3) os serviços que utilizam informação guardada de forma persistente.

Tabela 1 - Testes unitários e integrados - Tempos de resposta

Serviços	*Tempo médio (ms)*	*Nº de erros*	*Nº de métodos*	*Tempo Médio por método (ms)*
Exercícios	6593	0	7	941
Sensores	2443	0	7	350
Vídeo	2192	0	5	438
Conexões	7612	0	9	845
Gestão de utilizadores	3138	0	6	523
Gestão de alarmes	2863	0	6	477
Logger	1638	0	3	546
Gestão de prioridades	1146	0	2	573
Instalação automática	1526	0	3	508
Gestão de serviços	3618	0	6	603
Descoberta serviços	723	0	2	362
Monitor	356	0	1	356

4.2.2. Testes de carga

Depois de criados os testes utilizando a ferramenta SoapUI, é possível utilizar o loadUI para fazer testes de carga aproximados a um ambiente real. Inicialmente para testar a os serviços foram efetuados testes de carga a todos eles. Os serviços foram testados em termos de carga para uma situação onde se utilizam todos os métodos oferecidos pelos mesmos. Seguidamente foram feitos testes a duas situações concretas: (1) teste onde vários nós HPS utilizam todos os serviços do sistema num curto intervalo de tempo, de forma a obter qual o número máximo de utilizadores são suportados em simultâneo, e (2) teste onde várias aplicações utilizam os mesmos recursos que a aplicação de tele-reabilitação.

4.2.2.1. Testes de carga por serviço

Foram efetuados testes de carga a cada serviço. Nestes testes todos os métodos do serviço são invocados. Os testes foram efetuados utilizando um rácio inicial de um novo conjunto completo de pedidos por segundo que vai aumentado de forma aleatória. Como é improvável que todos os métodos sejam chamados sequencialmente (sem um intervalo de tempo), e para aumentar o realismo dos testes, foram colocados intervalos de tempo lógicos entre as invocações de cada método.

Na Figura 33 está demonstrado um teste de carga ao serviço de exercícios, porque este mostrou ser o serviço com menor desempenho. No teste, existe um novo pedido a um rácio inicial de um por segundo, ou seja, simula-se a existência de uma nova aplicação a utilizar o serviço em cada segundo da seguinte maneira:

- Começa-se por se inserir um endereço para onde o serviço envia as mensagens *HTTP*;
- Passado 500 milissegundos obtém-se esse *URL* (por questões de teste);
- 200 Milissegundos depois inserem-se os exercícios;
- Passados 1500 milissegundos obtém-se os exercícios e começa-se a correr os mesmos;
- Passado mais 2000 milissegundos avança-se para outro exercício;
- Por fim, e passados mais 1700 milissegundos, pára-se a execução de exercícios.

Os intervalos entre os vários métodos foram utilizados de forma a aumentar o realismo do teste. É improvável que todos os métodos sejam chamados sem um intervalo de tempo pela mesma aplicação/serviço.

Como se pode constatar na Figura 33, depois de 1 minuto e 33 segundos foram efetuados 529 pedidos, tendo ocorrido zero erros. Logo, todos os pedidos foram efetuados com sucesso, pelo que o serviço está capaz de suportar uma carga bastante razoável para o contexto de utilização do mesmo.

Figura 33 – Teste de carga ao serviço de exercícios

Ao mesmo tempo que estes testes foram efetuados, foi realizada uma recolha dos tempos de vários métodos do serviço. A Figura 34 mostra os tempos associados ao método que muda para um exercício específico e ao método que interrompe a execução de exercícios. Em alguns momentos, existem alguns picos de tempo de resposta que estão relacionados com o processamento em simultâneo excessivo no servidor (uma vez que este tem recursos muito limitados) ou ao facto do URL utilizado não estar na rede local do servidor. No entanto, e apesar de não serem tempos muito bons, são aceitáveis para o serviço mais lento do servidor e para uma quantidade tão excessiva de pedidos em simultâneo.

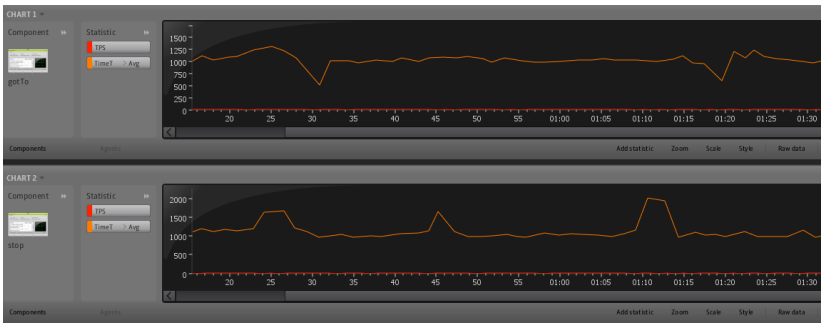


Figura 34 – Tempos de resposta do serviço de exercícios

A Tabela 2 apresenta o número total de pedidos e erros num determinado tempo para cada um dos serviços. Todos os serviços estão a responder eficientemente. Em alguns casos ocorreram um ou

dois erros, o que não é preocupante, uma vez que se deve ao fraco processamento da máquina onde está o HG. Em suma, o HG e os seus serviços têm a capacidade de responder eficientemente a uma capacidade relativamente elevada de pedidos.

Tabela 2 - Testes de carga por serviço

Serviços	*Tempo Ocorrido*	*Nº de erros*	*Nº de pedidos*
Exercícios	1m 33s	0	529
Sensores	1m 32s	1	733
Vídeo	1m 32s	0	423
Conexões	1m 32s	1	397
Gestão de utilizadores	1m 33s	2	602
Gestão de alarmes	1m 33s	0	501
Logger	1m 32s	0	671
Gestão de prioridades	1m 33s	0	279
Instalação automática	1m 33s	0	352
Gestão de serviços	1m 32s	0	643
Descoberta serviços	1m 33s	1	232
Monitor	1m 33s	0	157

4.2.2.2. Simulação de uma aplicação que utiliza todos os serviços

Neste fase foi feito um teste de carga que simula a utilização de todos os serviços do sistema (exceto os serviços da Micro I/O e do INESC) em simultâneo. Este teste, à semelhança do anterior, vai correr todos os métodos de cada um dos serviços, embora vá executar todos os serviços do sistema a um rácio de um pedido por segundo.

A grande vantagem deste teste é verificar como é que o servidor/serviços se comportam com uma carga bastante grande de pedidos a vários dos seus serviços/métodos. Uma carga tão grande num intervalo de tempo tão pequeno é algo raro. No entanto, poderá acontecer e, por isso, é crucial testar este cenário.

Nesta situação, verifica-se que alguns serviços falham e os tempos de resposta tornam-se, também, bastante altos, mesmo para serviços que individualmente mostraram ser bastante eficientes. Partindo do pressuposto que os erros não se devem a problemas relacionados com os serviços (porque estes já foram testados com sucesso) acredita-se que o problema está na capacidade de resposta do servidor (que tem uma capacidade relativamente baixas) que em 1 minuto e 19

segundos recebeu 1138 pedidos, não tendo conseguido responder com sucesso 14% deles. No que diz respeito ao tempo de resposta, este valor passou, em alguns casos, de um valor de tempo médio de resposta de aproximadamente 2500 milissegundos (para executar todos os serviços) para quase 5500 milissegundos, o que é um valor bastante mais elevado (Figura 35). As falhas começaram a ocorrer quando foram alcançados os 902 pedidos, nos 57 segundos do teste.

Figura 35 – Teste de carga a todos os serviços

Pode concluir-se que não existe capacidade no servidor/serviços para responder a tantos pedidos num tão curto espaço de tempo, durante tanto tempo, o que, para a máquina em questão, já era previsível e não crítico, uma vez que é muito pouco provável haver tantos pedidos em tão pouco tempo.

4.2.2.3. Simulação de utilização do nº de serviços da aplicação de tele-reabilitação

Por fim, foi efetuado um teste que simula a aplicação de tele-reabilitação[9] e que utiliza o serviço de conexões, o serviço de exercícios, o serviço de vídeo, o serviço de sensores, o serviço adapto, o serviço de gestão de utilizadores, o serviço de alarmes e o *logger*.

De forma a aumentar a carga foi simulada a existência de várias aplicações simultâneas. Inicialmente começou-se por realizar a simulação com sete utilizadores, que foram aumentando até ao momento em que ocorreram falhas.

[9] Explicado na secção 4.3.

Tabela 3 - Testes de carga - Máximo de utilizadores

Nº de Utilizadores	*Tempo Médio de execução*	*Nº de erros*	*Nº de pedidos*	*% de Erros*
7	9s	0	133	0%
14	9,5s	0	266	0%
20	10s	0	380	0%
30	12s	2	570	0,3%
33	14s	17	627	2,7%
35	18s	101	665	15,2%

Ao analisar a Tabela 3 pode verificar-se que quando existem trinta e cinco utilizadores, o número de falhas aumenta exponencialmente. Em suma, o servidor/serviços têm a capacidade de responder eficazmente quando trinta e três utilizadores usufruem, simultaneamente, de uma aplicação equiparada à aplicação de tele-reabilitação.

4.3. O HG e a aplicação tele-reabilitação

Durante o decorrer do projeto LUL foram desenvolvidas, na Universidade de Aveiro, em paralelo com o HG e o LULMS, duas aplicações com multimodalidade: a aplicação do especialista de saúde (aplicação HPS) que estão localizadas no HPS, e a aplicação do paciente (aplicação HS) que estão localizada no HS. Estas aplicações, para além de representarem o resultado final do projeto LUL, vão utilizar os serviços disponíveis tanto no HG como no LULMS. Assim, são as aplicações que são utilizadas para testar a arquitetura e respetivos serviços do HG. Esta seção pretende descrever o modo como o HG é utilizado por esta aplicação.

De forma a compreender o papel do HG, será explicitada a aplicação e apresentado um diagrama sequencial que demonstrará a utilização do HG. Para além do que será explicado nas seguintes secções, estas aplicações podem ainda vir a integrar alguma da informação obtida pelos serviços da INESC e da Micro I/O.

4.3.1. A aplicação HPS

A aplicação do profissional de saúde (Figura 36) tem como objetivo obter e providenciar dados da e para a casa [10].

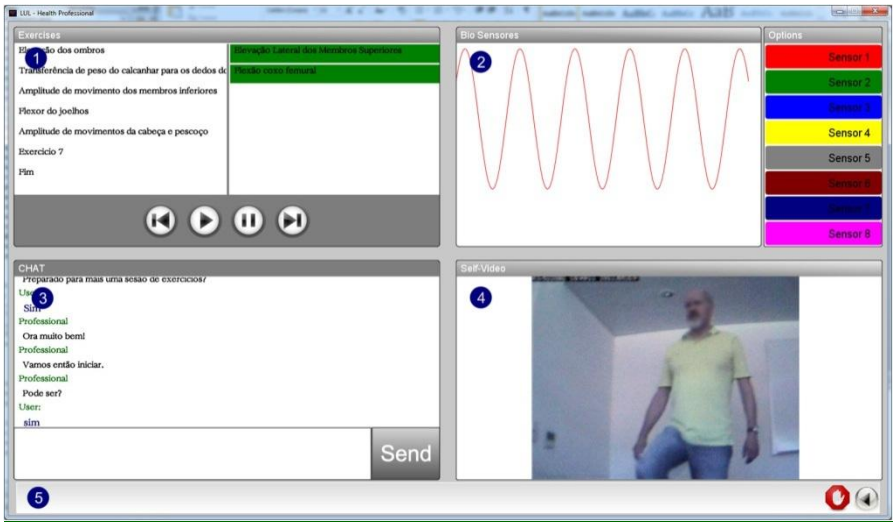


Figura 36 – Aplicação do profissional de saúde

Esta aplicação permite monitorizar o paciente através de vídeo e sensores vitais [10]. Para a utilização do vídeo é utilizado o protocolo de comunicação RTCP que permite estabelecer uma comunicação de vídeo/som em tempo real de alta qualidade. Também é utilizado o serviço de vídeo para controlo total da câmara de vídeo, tornando possível ao utilizador controlá-la, interagindo diretamente sobre a imagem do vídeo. Os dados dos sensores são obtidos do serviço de sensores e, posteriormente, transformados em informação pela aplicação que os vais apresentar.

Permite, também, enviar alguma informação para o utilizador. Essa informação pode referir-se a mensagens, para que ambos os lados estejam em comunicação, mas pode também ser o envio de dados relativos aos exercícios [10]. Estes dados são enviados para o serviço de exercícios descrito no capítulo anterior. Como se pode constatar na figura, o profissional de saúde tem a possibilidade de avançar ou recuar para outro exercício, de parar a execução de exercícios, e de os colocar a correr. Isto permite ao profissional de saúde planear, controlar e monitorizar os exercícios eficientemente.

A aplicação permite, ainda, que o utilizador guarde os dados da sessão (por exemplo, o exercício em que ficou) [10].

4.3.2. A aplicação HS

Esta aplicação, por ser específica para seniores, que, tal como já foi referido, têm algumas limitações em lidar com tecnologia, foi alvo de uma maior atenção ao nível de usabilidade. Por esse motivo, o interface foi desenvolvido para ecrãs com maior dimensão e para permitir uma navegação simples e de rápida aprendizagem [10].

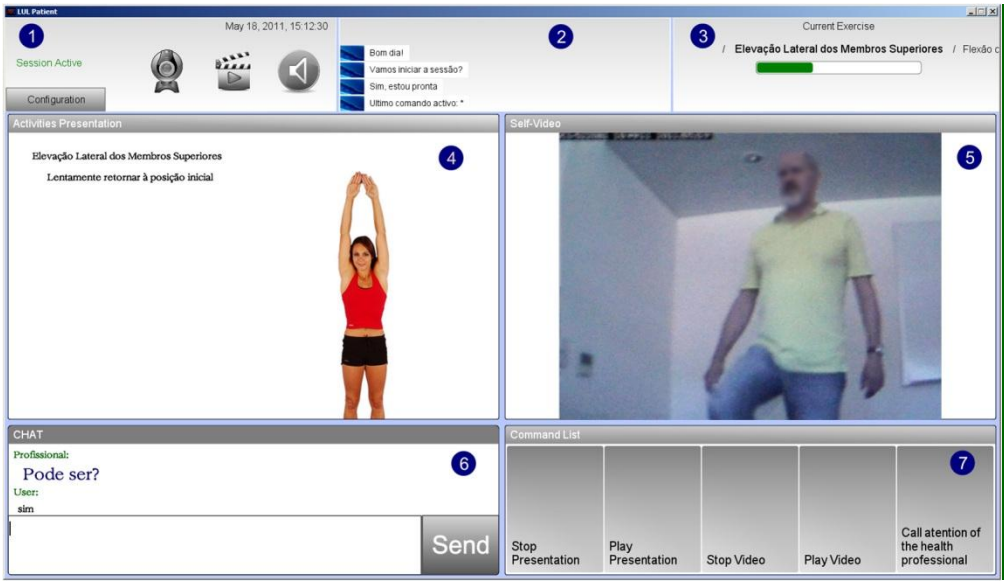


Figura 37 – Aplicação do Sénior

A Figura 37 representa os vários blocos constituintes da aplicação [10]. Os três blocos superiores sumarizam a informação relativa à sessão, sendo o seu o principal objetivo a apresentação do estado da sessão ao utilizador [10]. Para isso, é exposta a hora e a data no primeiro componente, um registo das últimas ações do profissional de saúde e do paciente no segundo componente e a lista de exercícios planeados para a sessão no terceiro componente [10]. Nos dois componentes do meio da aplicação (4 e 5) é apresentado, respetivamente, o exercício a realizar (esta informação é enviada para a aplicação por parte do serviço de exercícios) e é apresentada uma imagem do paciente para que este possa avaliar o seu desempenho. Os últimos dois blocos permitem o envio de mensagens do sénior para o especialista.

Esta aplicação utiliza o serviço AdaptO para se adaptar ao contexto do utilizador, aumentando, por exemplo, o tamanho de letra quando o paciente se afasta ou diminuindo-o quando ele se aproxima do dispositivo que apresenta a aplicação. Para além disso, utiliza o serviço de conexão do HG para se conectar ao LULMS, e informá-lo acerca dos recursos está a utilizar.

4.3.3. Papel do HG

Nesta secção estão descritas em detalhe as várias etapas associadas à comunicação entre a aplicação HS e a aplicação HPS. Esta comunicação está dividida em várias fases: a fase de conexão e de alocação de recursos, a fase de comunicação vídeo, e a fase de obtenção de dados de sensores. A fase de conexão e de alocação de recursos é a primeira fase do processo de comunicação. As duas restantes fases ocorrem simultaneamente em conjunto com a comunicação direta via *chat*.

Na primeira etapa, a fase de conexão e de alocação de recursos, os dois intervenientes, a aplicação HPS e a aplicação HS, têm de se conectar com o LULMS. A aplicação HS tem de se conectar

recorrendo ao serviço de conexão do HG. Depois de devidamente conectados, a aplicação HPS irá obter dados para se conectar diretamente com o HS. Ao conectar-se tem de se autenticar no HS. O processo de autenticação é responsabilidade da aplicação HS que utilizará o HG para descobrir se a aplicação HPS tem autorização para avançar com a comunicação através do serviço de gestão de utilizadores. De seguida, a aplicação HS regista os recursos que vão ser utilizados utilizando o serviço de conexão. Finalmente, a aplicação HS informa a aplicação HPS se tem privilégios de acesso (Figura 38). O que se pode verificar é que nesta fase a aplicação HS é capaz de obter informação do exterior e de se autenticar com o exterior sem nunca comunicar diretamente para o exterior.

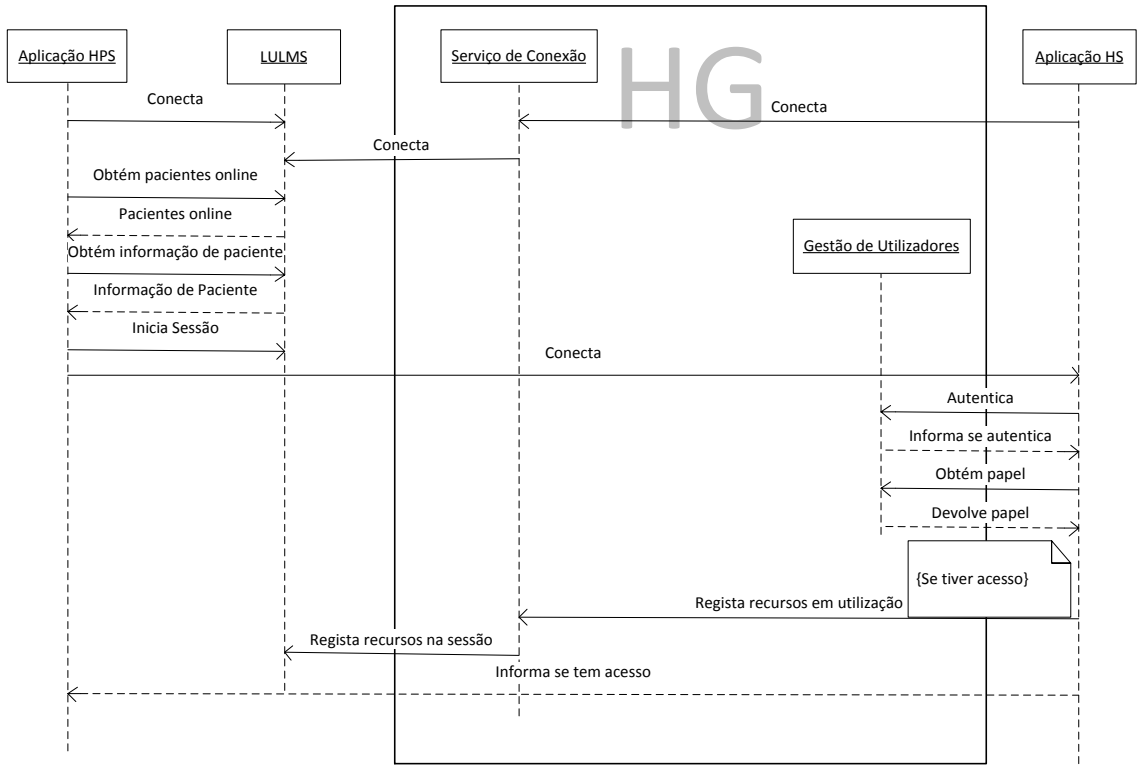


Figura 38 – O HG e a aplicação - fase de conexão

Na fase de obtenção de dados são pedidas as informações do serviço de sensores ao LULMS. Depois de obtidas as informações, a aplicação HPS comunica diretamente com o serviço de sensores do HG e obtém os dados pretendidos.

Na fase de comunicação vídeo são pedidas as informações do serviço de vídeo ao LULMS. Após o processo de obtenção dessa informação é estabelecida a sessão de vídeo. Sempre que é necessário enviar um comando de controlo para a câmara de filmar, a aplicação HPS comunica diretamente com o serviço de vídeo do HG.

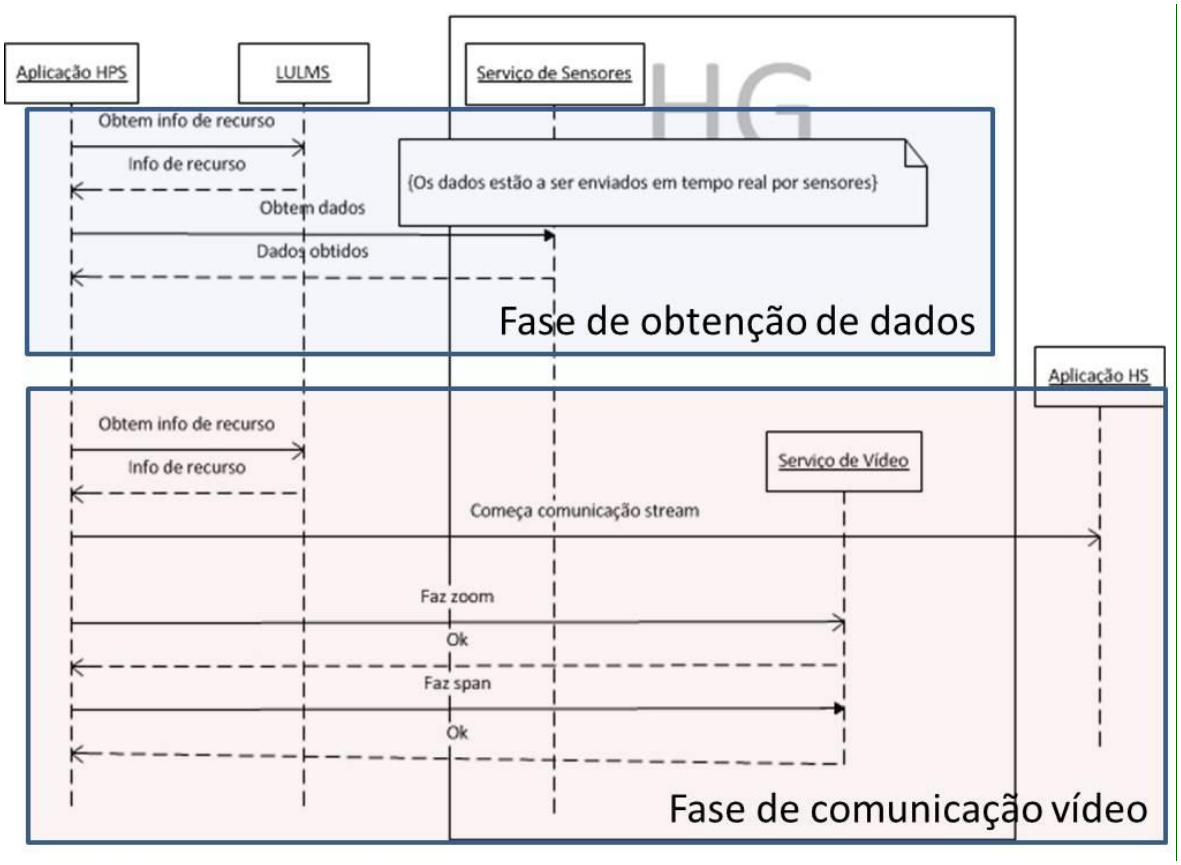


Figura 39 – O HG e a aplicação - fase de obtenção de dados e de vídeo

A Figura 39 representa a azul a fase de obtenção de dados e a rosa a fase de comunicação vídeo.

Durante todas as fases, os serviços do HG registam informação no serviço de *logger* sempre que são invocados. Assim, é criado um registo completo com toda a informação de acessos aos serviços do HG.

Nesta aplicação pode ser utilizado o serviço de exercícios. A obtenção da informação deste recurso é feito da mesma forma que os explicados na Figura 39.

A aplicação HS tem que, como pré-requisito, aceder ao serviço de descoberta de serviços para obter a informação associada aos serviços dos recursos que necessitará. Esta informação é utilizada na fase de conexão e de alocação de recursos.

Por outro lado, necessita de aceder ao serviço de gestão de utilizadores de forma a autenticar a aplicação que pretende comunicar. Apesar da autenticação da aplicação HPS com o LULMS já ter sido estabelecida, é importante existir autenticação no HS de forma a verificar se a aplicação HPS pode aceder aos recursos oferecidos.

Para além destes serviços, existe o serviço AdaptO do HG que vai adaptar a fonte da letra dependendo das necessidades do utilizador.

4.4. Teste integrado do LUL

Nesta secção pretende-se explicar como é que o HG e os seus serviços podem facilitar o processo de testes e o que é que estes permitirão.

O teste e avaliação dos conceitos/soluções criadas durante o mesmo são nucleares ao projeto LUL. Os testes associados ao projeto não podem ser realizados apenas em laboratório, devendo ser efetuados em ambientes próximos dos reais, pelo que fatores como a definição dos locais a testar, a definição do tipo de ligação necessária e a definição de equipamentos é crucial.

Os testes integrados com o projeto LUL ainda não foram realizados por questões de calendarização do projeto e, por isso, não é possível mostrar resultados dos mesmos. No entanto, todas as condições estão criadas para que estes testes sejam efetuados com sucesso.

A Figura 40 representa de forma sistemática quais os locais planeados para realizar os testes. O sistema é constituído por dois tipos de locais: o HS e HPS. Em alguns casos, os edifícios compreendem as duas vertentes, porque têm uma zona que corresponde à HS e outra que corresponde à HPS.

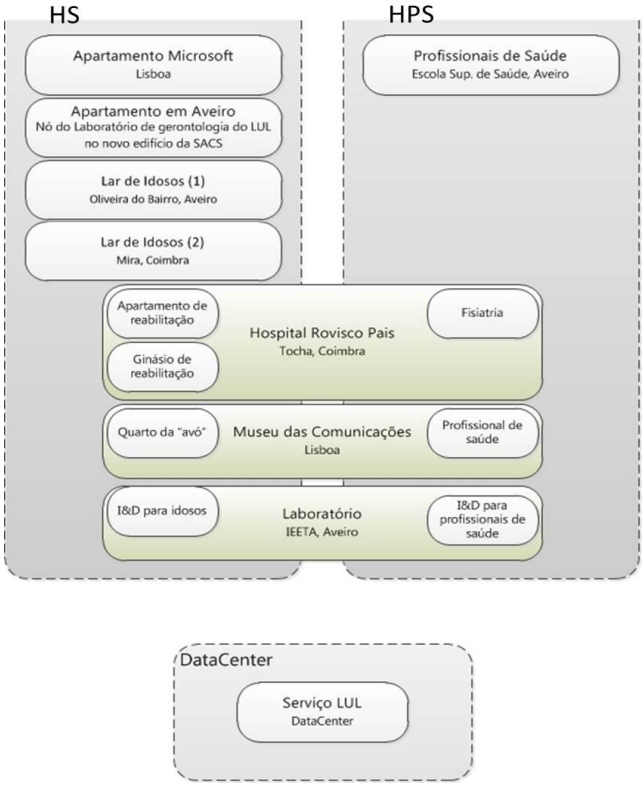


Figura 40 – Locais de teste

A utilização de vários locais é bastante importante uma vez que permite testar a capacidade do sistema funcionar num paradigma de cliente-servidor (entre o HS e o LULMS).

Vão existir muitos nós correspondentes ao HG e este está preparado para alterar os seus serviços e módulos de forma dinâmica e automática, recorrendo à informação disponibilizada para o efeito pelo LULMS. Assim, o HG facilitará a instalação dos serviços nos vários nós e facilitará ainda a realização de atualizações para corrigir problemas com os novos serviços.

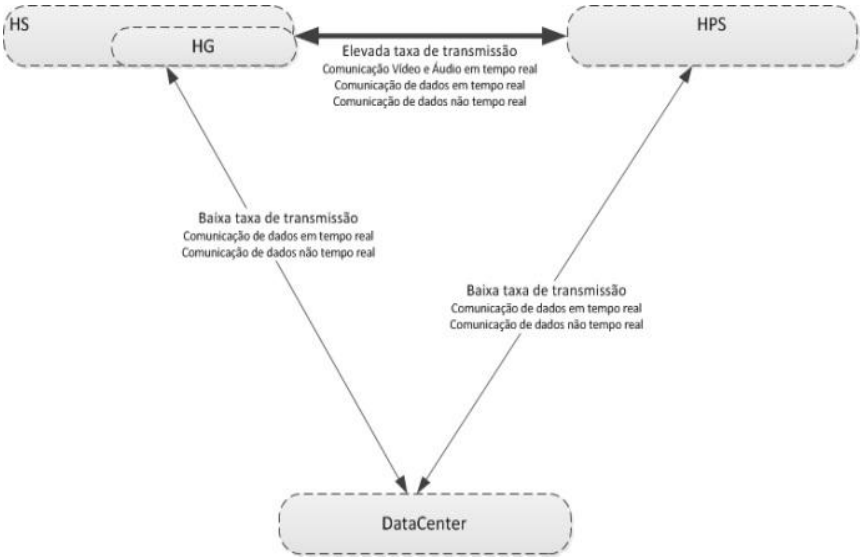


Figura 41 – Comunicação necessária para teste

É muito importante analisar capacidade necessária para efetuar uma boa comunicação, para que se possam realizar os testes com sucesso (Figura 41).

Entre o HS, mais concretamente o HG, e o HPS é importante a existência de uma estrutura que permita uma elevada taxa de transmissão, já que se realizarão sessões de comunicação de elevado débito (comunicação vídeo e áudio em tempo real) e, em alguns casos, sensível, sendo importante garantir a qualidade desta. O HG e os seus serviços garantem a qualidade e a segurança de transmissão de dados, recorrendo, por exemplo, ao protocolo RTCP para aumentar a qualidade de *streaming* de vídeo e áudio e ao serviço de monitorização que pode terminar serviços de modo a reduzir o fluxo de dados na rede. Por outro lado, de forma a diminuir o fluxo de dados da rede, o HG faz as suas atualizações durante a noite ou quando nenhum recurso está a comunicar com o exterior.

Para a comunicação entre a HG/HPS e o LULMS os requisitos são menos exigentes, pois a taxa de transmissão de dados é bastante reduzida. A comunicação dos dados é de baixo débito mas em situações particulares poderá ser necessário efetuar comunicações com um débito mais elevado. No entanto, não são requeridos acessos rápidos, pelo que se a comunicação for mais lenta não haverá qualquer impacto.

A utilização de redes de nova geração é essencial no projeto, portanto é importante que os testes sejam efetuados sobre fibra, 4G, ADSL e 3G (Figura 42). Nos locais com funções duais (HS e

HPS), e fundamentalmente dentro campus da Universidade de Aveiro, a presença simultânea de todos estes tipos de comunicação é essencial. A presença de ligações via Cabo é interessante no sentido de provar a expansibilidade dos serviços para outros fornecedores. O HG e os seus serviços são capazes de se adaptar ao tipo de ligação, principalmente recorrendo ao serviço de monitorização. No entanto, o tipo de ligação pode pôr em causa a qualidade de alguns serviços, como por exemplo, o serviço de vídeo que exige uma grande capacidade de transmissão de dados.

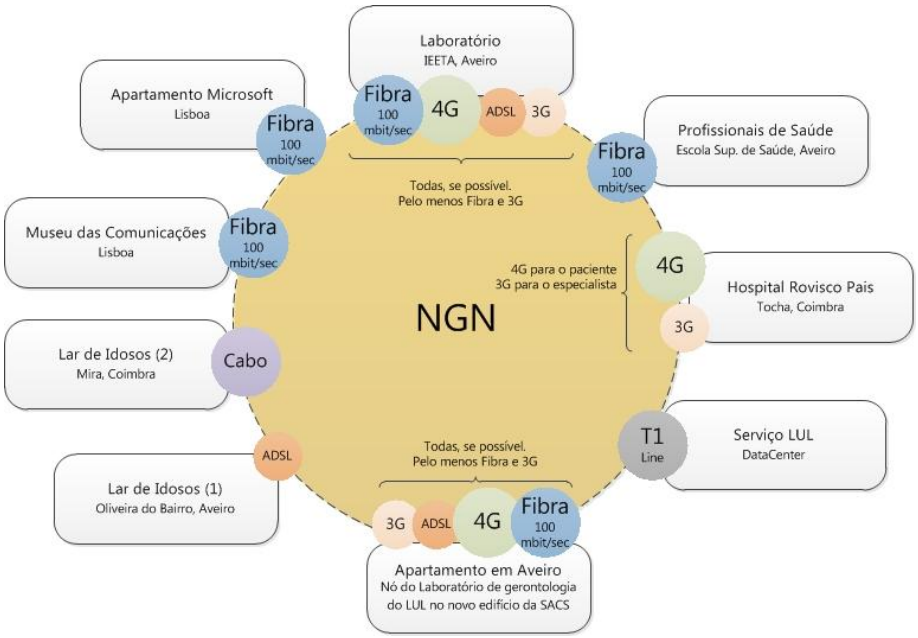


Figura 42 – Tipos de ligação de teste

O HS é constituído por vários equipamentos como, por exemplo: os sensores (responsáveis por obter dados como sinais vitais), os atuadores (responsáveis por atuar dependendo dos dados obtidos pelos sensores), os dispositivos de visualização (responsáveis por mostrar informação aos seniores), os dispositivos de captura de imagem/vídeo (responsáveis pela transmissão de imagens/vídeo para os especialistas de saúde) e o HG (responsável por gerir toda a comunicação dentro e fora da casa de forma rápida, segura e eficiente). O HPS é constituído essencialmente por dispositivos de visualização (responsáveis pela apresentação de informação aos especialistas) e por dispositivos de captura de imagem/vídeo (responsáveis pela transmissão de imagens/vídeo para os seniores).

De forma a compreender melhor o papel do HG, a estrutura física dentro do campus da Universidade de Aveiro (UA), que inclui dois nós correspondentes ao HS e dois nós correspondentes ao HPS, será explicitada de seguida.

Dentro do campus da UA existem três locais principais de teste: (1) o laboratório do IEETA, (2) o apartamento da Secção Autónoma de Ciências da Saúde (SACS) e (3) o local onde estão os profissionais de saúde. O laboratório do IEETA tem funções duais, incorporando o HS e o HPS,

sendo este utilizado para propósitos de testes de desenvolvimento. O apartamento da SACS representa o HS. E o último local representa um nó HPS.

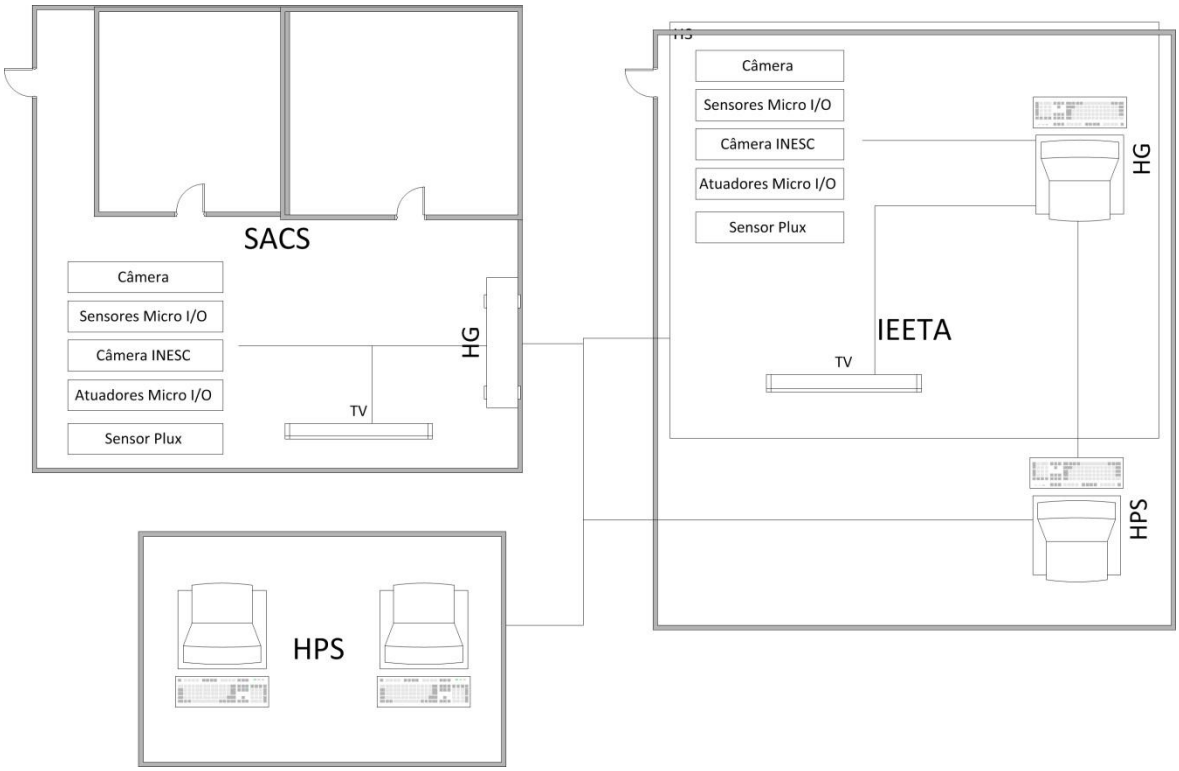


Figura 43 – Nós da Universidade de Aveiro

A Figura 43 representa os nós criados no campus da Universidade de Aveiro e respetivos dispositivos. Atualmente existe, no laboratório do IEETA, uma televisão, uma câmara de vídeo IP, alguns sensores da Micro I/O e o sensor da Plux. Os testes já realizados foram feitos dentro do laboratório do IEETA.

5. Conclusões

5.1. Trabalho desenvolvido

O desenvolvimento deste trabalho dividiu-se em várias fases distintas. Numa primeira fase houve a necessidade efetuar revisão bibliográfica, para melhor compreensão e envolvimento no projeto. Posteriormente foi feita a análise de várias soluções de AAL e a conceção da arquitetura e dos serviços do HG, seguida do desenvolvimento dos mesmos. Por fim, elaboraram-se os testes e a respetiva análise dos resultados dos vários serviços.

O projeto começou, assim, com os procedimentos e leituras indispensáveis, de forma a compreender os principais objetivos do mesmo e de ter bases para a tomada de decisão acerca da arquitetura a utilizar.

Procedeu-se a uma leitura e análise de várias soluções AAL com o intuito de obter um conjunto de requisitos essenciais e de soluções de arquitetura e tecnologias para cada um desses requisitos.

Durante a fase de conceção, tendo em conta os requisitos obtidos e a análise feita anteriormente, decidi, em conjunto com os restantes membros do projeto, elaborar uma arquitetura orientada a serviços que, quando bem implementada, proporciona re-utilização, granularidade, modularidade, composabilidade e componetização, auditoria de serviços, e interoperabilidade.

Passou-se à fase de desenvolvimento de um sistema que respondesse eficientemente à necessidade de este ser autónomo, de interoperabilidade, de segurança, de escalabilidade, de integração de um número arbitrário de dispositivos, de suportar processamento paralelo, de conter monitorização, de ter o efeito de *firewall*, de recolha e processamento de dados, de gestão remota, de permitir parar/atualizar/reinicializar/desinstalar serviços em tempo de execução, de existência de multiplataforma, de personalização, de adaptação, de confirmação/verificação das decisões autónomas do sistema, de troca de dados apenas se estes forem relevantes, de ser eficaz, de ser embebido e distribuído e de providenciar interação explícita. Foi ainda realizada, na fase de desenvolvimento, uma análise para que o HG se pudesse fundir nas equipamentos de operadores de rede de nova geração.

Foram, por fim, efetuados testes a todos os serviços e fez-se uma análise dos resultados de forma a melhorar e corrigir os serviços que não respondessem de acordo com o esperado. Esta fase foi feita de forma iterativa, pelo que, todos os serviços estão a responder como esperado. Apesar de alguns serviços providenciarem tempo de resposta demasiado alto (serviço de exercícios), esse problema não acarreta consequências críticas no sistema. A máquina escolhida responde eficientemente aos requisitos do HG, apesar de conter os requisitos mínimos. Os testes realizados demonstram que HG facilita o desenvolvimento e *deploy* de novos serviços e de aplicações.

Em suma, os objetivos do trabalho foram cumpridos, uma vez que foi desenvolvido um HG para suporte a serviços AAL, tendo sido criadas as condições essenciais para o desenvolvimento de novos serviços.

5.2. Limitações

Apesar de terem sido feitos alguns testes aos módulos desenvolvidos é crucial que esses testes sejam feitos num ambiente real de forma a obter um maior *feedback*, tanto dos utilizadores do sistema como da capacidade dos serviços e mesmo da rede.

O sistema de obtenção de dispositivos foi apenas testado utilizando uma câmara pelo que são necessários testes mais reais (com mais dispositivos). No entanto, de acordo com o *feedback* obtido é igualmente previsível que o sistema responda da mesma maneira com a existência de novos dispositivos.

A necessidade de testar toda a arquitetura do projeto com vários nós é igualmente crucial para averiguar se o sistema desenvolvido realmente facilita o processo de integração de novos serviços e o processo de atualização dos mesmos. Estes nós foram testados apenas localmente não tendo havido comunicação entre redes diferentes mas sim dentro da mesma rede, pelo que é de esperar que o sistema tenha resultados semelhantes. No entanto, o teste é crucial.

O sistema de interoperabilidade funciona à base de serviços, onde os fornecedores são obrigados a implementar serviços específicos para os seus dispositivos, tornando-os, assim, automaticamente acessíveis por todas as aplicações (como, por exemplo, os dispositivos e respetivos serviços da Micro I/O). Apesar de isto providenciar interoperabilidade, isso não acontece no caso de os fornecedores não desenvolverem os seus serviços, sendo, por isso, um aspeto menos positivo do sistema.

5.3. Sugestões de Continuidade

5.3.1. Outras aplicações na vida real

O principal objetivo deste projeto e, consequentemente, do HG, era apoiar a população sénior nas suas habitações. No entanto, tornou-se claro que o HG pode ser utilizado noutras áreas.

Uma das vertentes que poderia lucrar com a introdução do HG seria a dos doentes portadores de disfunções a nível cognitivo (doentes com demência de Alzheimer em estado precoce ou outro tipo de demência incapacitante ligeira, traumatizados de crânio ligeiro e moderado, doentes que tenham sofrido de acidente vascular cerebral), uma vez que estes são bastante penalizados com a falta de autonomia e consequente falta de independência dentro da sua habitação tendo, na maioria dos

casos, que se deslocar permanentemente para a habitação de prestadores de cuidados informais. A população-alvo neste caso teria uma faixa etária compreendida entre o jovem adulto até ao sénior, uma vez que há défices cognitivos que podem ocorrer em qualquer idade, originando a incapacidade e criando obstáculos no quotidiano que anteriormente não existiam.

O HG criado poderia, ainda com a população acima descrita, não só apoiar no quotidiano mas também na reabilitação de pessoas com disfunções cognitivas que tivessem acompanhamento complementar exterior. Desta forma, o sistema poderia utilizar lembretes com as atividades do dia-a-dia, com a intenção de os doentes tomarem os comprimidos às horas corretas, e com a intenção destes praticarem exercícios virtuais (que, para a população com lesão cerebral progressiva seriam exercícios de estimulação cognitiva) propostos pelo neuropsicólogo cujos resultados seriam automaticamente comunicados ao mesmo. Neste panorama, tanto a população-alvo como os prestadores de cuidados informais sentiriam um alívio na sobrecarga diária de tarefas, no sentido em que reduziam a frequência de deslocações (que não são fáceis devido às dificuldades de mobilidade e, por vezes, à demora que se pode fazer sentir na mesma), deixando a agenda mais livre para outros tratamentos essenciais no processo de reabilitação como a terapia ocupacional e da fala, hidroterapia, entre outras.

O projeto realizado com o HG tem, para além da capacidade de aplicação em habitações próprias, o potencial para se alargar a instituições de apoio à sociedade como os lares de idosos ou os centros prestadores de cuidados a idosos onde é frequente encontrar-se um número de funcionários que está aquém das necessidades e da quantidade de seniores a usufruir do serviço. Assim, torna-se inexequível o acompanhamento constante a todos os seniores da instituição por parte dos funcionários. Neste cenário, e ainda tendo em conta que os lares de idosos, de forma geral, apenas costumam contar com um médico e um enfermeiro que não estão presentes a tempo inteiro nem todos os dias da semana, um sistema como o desenvolvido neste projeto, ajudaria a prevenir situações de risco, a monitorizar a população sénior (que, ainda com um desenvolvimento saudável, acaba por carecer de um acompanhamento regular, devido aos problemas de saúde intrínsecos à idade avançada), a alertar para situações de urgência, e a gerir alguns aspetos da instituição, como forma de apoio aos especialistas de saúde da mesma.

Há outro tipo de instituições que poderiam beneficiar com o sistema acima descrito, como as instituições de apoio a pessoas portadores de deficiência (caso do Centro de Medicina de Reabilitação de Alcoitão, em Alcabideche), nos casos de internamento para reabilitação, por incapacidade motora e/ou cognitiva (esclerose múltipla, lesões músculo-esqueléticas ou vertebro-modulares, entre outras), fornecendo aos funcionários (nomeadamente aos fisioterapeutas) apoio através da utilização de serviços especializados ou mesmo através dos serviços já existentes no sistema.

Seria igualmente interessante implementar este sistema em hospitais e centros de saúde, também na secção do internamento, com o objetivo de apoiar os médicos e enfermeiros a gerir as suas obrigações profissionais para com os doentes (e até de ajudar os doentes mais independentes a gerir por si a maioria das tarefas do seu dia-a-dia no hospital), encurtando o tempo que estes precisam de despender com alguns doentes e, consequentemente, criando-lhes a possibilidade de terem um maior número de consultas diárias, aliviando as listas de espera que se fazem sentir neste tipo de instituições públicas.

Aliado a dispositivos móveis, e saindo um pouco do que está inerente à definição de AAL mas indo de encontro ao AmI, este sistema de HG poderia ser útil para pais que precisem de controlar as crianças em casa, em casos em que têm, por exemplo, que sair de casa e deixar as crianças sozinhas durante algum tempo.

5.3.2. Trabalho futuro

Apesar de o sistema já responder aos objetivos principais, seria interessante implementar novas funcionalidades para que este funcione de forma mais eficiente.

A integração e implementação de ontologias tanto no HG como na restante arquitetura do projeto LUL facilitaria a partilha de informação (relativa aos serviços, dispositivos, etc.) entre os vários componentes do sistema, criaria conhecimento, permitiria a reutilização do conhecimento do domínio, tornaria as suposições do sistema explícitas para todos os componentes, dividiria o conhecimento do domínio do conhecimento operacional, e analisaria o conhecimento eficazmente. Para isso, seria interessante criar uma solução baseada nas propostas do projeto OASIS [4].

O sistema de interoperabilidade criado é baseado na criação de serviços que permitam controlar/obter dados dos dispositivos. No entanto, isto pode criar alguns constrangimentos, pelo que seria interessante implementar o sistema de *bus* do projeto SODAPOP [38] que é utilizado em outros projetos e está bastante testado, criando assim interoperabilidade total no sistema.

O teste do sistema em ambientes próximos dos reais e com recursos às redes de nova geração é algo crucial para o desenvolvimento da aplicação e deve ser feito futuramente.

É de esperar que novos serviços sejam criados e colocados no HG, pelo que a integração de novos serviços de apoio ao utilizador sénior é crucial, tanto para testar o sistema como para o fazer crescer.

5.4. Oportunidade de Aprendizagem

A oportunidade de ter trabalhado neste sistema inserido no projeto LUL trouxe benefícios e uma evolução positiva na minha aprendizagem, enriquecendo os meus conhecimentos, não só na área de arquiteturas orientadas a serviços, e na área de desenvolvimento de sistemas autónomos de suporte

ao desenvolvimento de serviços para AAL, mas também na área das ciências sociais que serve de base ao estudo do envelhecimento saudável e disfuncional. Para além disso, ajudou-me a desenvolver competências de análise de requisitos, compreensão dos requisitos pedidos, competências interpessoais de trabalho em equipas de grandes dimensões e de elaboração de documentos explicativos do *Software* e da documentação necessária para utilização do código desenvolvido. Proporcionou-me a oportunidade de trabalhar com diversas instituições com o objetivo de integrar o trabalho desenvolvido por eles com o trabalho desenvolvido por mim, intensificando, assim, as minhas capacidades de comunicação e de trabalho em equipa.

Para além disso, ofereceu-me a oportunidade de cooperar na criação de dois artigos [2, 10] que já foram publicados e dois capítulos de livro [1, 21] que, apesar de terem sido aceites, ainda não foram publicados.

6. Referências

[1] A. Teixeira, N. Rocha, C. Pereira, J. S. Pinto, M. S. Dias, C. Teixeira, M. O. e. Silva, A. Queirós, F. Ferreira, and A. Oliveira, "The Living Usability Lab Architecture: Support for the Development and Evaluation of New AAL Services for the Elderly," 2011.

[2] C. Teixeira, J. S. Pinto, F. Ferreira, A. Oliveira, A. Teixeira, and C. Pereira, "Cloud Computing Enhanced Service Development Architecture for the Living Usability Lab ENTERprise Information Systems." vol. 221, M. M. Cruz-Cunha, J. Varajão, P. Powell, and R. Martinho, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 289-296.

[3] O. Alliance, "OSGi Service Platform Core Specification," 2011.

[4] O. Consortium. (2008). *About OASIS*. Available: http://www.oasis-project.eu/

[5] M. N. K. Boulos and eCAALYX Project Consortium, "An Enhanced Ambient Assisted Living Experiment for Older People with Multiple Chronic Conditions," ed, 2009.

[6] A. Grguric, "ICT towards elderly independent living," Research and Development Center Ericsson Nikola Tesla, Krapinska 45, Zagreb, Croatia2012.

[7] K. Lee, T. Lunney, K. Curran, and J. Santos, "Proactive Context-Awareness in Ambient Assisted Living," University of Ulster2008.

[8] M.-R. Tazari, F. Furfari, J.-P. L. Ramos, and E. Ferro, "The PERSONA Service Platform for AAL Spaces," in *Handbook of Ambient Intelligence and Smart Environments*, ed US: Springer-Verlag, 2010, pp. 1171-1199.

[9] A. Hein, M. Eichelberg, O. Nee, A. Schulz, A. Helmer, and M. Lipprandt, "A Service Oriented Platform for Health Services and Ambient Assisted Living," in *Advanced Information Networking and Applications Workshops, 2009. WAINA '09. International Conference on*, 2009, pp. 531-537.

[10] A. Teixeira, C. Pereira, M. O. e. Silva, N. Almeida, J. S. Pinto, C. Teixeira, F. Ferreira, and A. Mota, "Health@Home Scenario: Creating a New Support System for Home Telerehabilitation," presented at the 2º International Living Usability Lab Workshop on AAL Latest Solutions, Trends and Applications in BIOSTEC 2012, Vilamoura, Portugal, 2012.

[11] S. O'Brien. (2012). *Aging Population: Seniors are Fastest Growing Population Worldwide*. Available: http://seniorliving.about.com/od/lifetransitionsaging/a/seniorpop.htm

[12] D. H. Steg, D. H. Strese, C. Loroff, J. Hull, and S. Schmidt, "Europe Is Facing a Demographic Challenge Ambient Assisted Living Offers Solutions," March 2006.

[13] I. N. d. Estatística, "Censos 2011," 2011.

[14] A. J. Jara, M. A. Zamora, and A. F. G. Skarmeta, "An Architecture Based on Internet of Things to Support Mobility and Security in Medical Environments," in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, 2010, pp. 1-5.

[15] (2008). *AAL Joint Programme*. Available: http://www.aal-europe.eu/

[16] H. Pigot, A. Mayers, and S. Giroux, "The intelligent habitat and everyday life activity support," presented at the 5th Int. Conf. on Simulations in Biomedicine, 2003.

[17] J. Nehmer, A. Karshmer, M. Becker, and R. Lamm, "Living assistance systems: an ambient intelligence approach," presented at the 28th Intl. Conf. on ICSE '06, 2006.

[18] L. Consortium. (2010). *Living Usability Lab*. Available: www.livinglab.pt

[19] A. Teixeira, N. Rocha, M. S. Dias, D. Braga, A. Queirós, O. Pacheco, J. A. Fonseca, J. S. Pinto, H. Gamboa, L. Corte-Real, J. Fonseca, J. A. Martins, A. Neves, P. Bartolomeu, C. Oliveira, J. Cunha, and C. Pereira, "A New Living Lab for Usability Evaluation of ICT and Next Generation Networks for Elderly@Home," presented at the 1º International Living Usability Lab Workshop on AAL Latest Solutions, Trends and Applications in BIOSTEC 2011, Rome, Italy, 2011.

[20] A. A. Consortium. (2012). *Ambient Assisted Living for All*. Available: http://www.aal4all.org

[21] C. Teixeira, J. S. Pinto, F. Ferreira, A. Oliveira, A. Teixeira, and C. Pereira, "Cloud Computing Enhanced Service Development Architecture for the Living Usability Lab," ed, 2012.

[22] I. S. T. A. Group, "Recommendations of the IST Advisory Group for Workprogramme 2001 and beyond "implementing the vision"," 2001.

[23] C. I. w. C. Crutzen. (2010). *Council Interview with Cecile Crutzen on Ambient Intelligence and Internet of Things* Available: http://www.theinternetofthings.eu/content/council-interview-cecile-crutzen-ambient-intelligence-and-internet-things

[24] H.-D. Ma, "Internet of Things: Objectives and Scientific Challenges," *Journal of Computer Science and Technology*, vol. 26, pp. 919-924, 2011.

[25] M. N. K. Boulos and eCAALYX Project Consortium, "eCAALYX: Towards a Real-world Ambient Assisted Living Solution that Delivers in Non-technical Environments and Is Sustainable," 2009.

[26] A. Rodrigues, C. Resende, L. Carvalho, P. Saleiro, and F. Abrantes, "Performance analysis of an adaptable home healthcare solution," in *e-Health Networking Applications and Services (Healthcom), 2011 13th IEEE International Conference on*, 2011, pp. 134-141.

[27] A. Pitsillides, E. Themistokleous, G. Samaras, S. Walderhaug, and O. M. Winnem, "Overview of MPOWER: Middleware Platform for the Cognitively Impaired and Elderly," presented at the ISTAfrica, Africa, 2007.

[28] S. Walderhaug, E. Stav, and M. Mikalsen, "Experiences from Model-Driven Development of Homecare Services: UML Profiles and Domain Models," in *Models in Software Engineering*, R. C. Michel, Ed., ed: Springer-Verlag, 2009, pp. 199-212.

[29] W. Ståle, M. Marius, and S. Erlend, "Reusing Models of Actors and Services in Smart Homecare to Improve Sustainability," in *eHealth Beyond the Horizon – Get IT There - Proceedings of MIE2008 – The XXIst International Congress of the European Federation for Medical Informatics*, Gothenburg, 2008, pp. 107-112.

[30] P. Lazarevski, I. Benc, and J. Jurišić, "The role of business processes in creation of adaptive health-care applications by utilization of MPOWER platform."

[31] M. Consortium, "MPOWER Project Deliverable: Overall Architecture," 2008.

[32] M. Marius, S. Hanke, T. Fuxreiter, W. Ståle, and W. Leendert, "Interoperability Services in the MPOWER Ambient Assisted Living Platform," in *The XXIInd International Congress of the European Federation for Medical Informatics*, Amsterdam, 2009, pp. 366-370.

[33] A. Arsanjani, L.-J. Zhang, M. Ellis, A. Allam, and K. Channabasavaiah. (2007). *Design an SOA solution using a reference architecture - Improve your development process using the SOA solution stack*. Available: www.ibm.com/developerworks/library/ar-archtemp/

[34] Amigo, "Short project description," 2004.

[35] M. Janse, P. Vink, and N. Georgantas, "Amigo Architecture: Service Oriented Architecture for Intelligent Future In-Home Networks," in *AmI 2007 Workshops*, 2008, pp. 371–378.

[36] G. Thomson, D. Sacchetti, Y.-D. Bromberg, J. Parra, N. Georgantas, and V. Issarny, "Amigo Interoperability Framework: Dynamically Integrating Heterogeneous Devices and Services Constructing Ambient Intelligence." vol. 11, M. Mühlhäuser, A. Ferscha, and E. Aitenbichler, Eds., ed: Springer Berlin Heidelberg, 2008, pp. 421-425.

[37] J. Kalaoja, J. Kantorovitch, S. Carro, J. M. Miranda, Á. Ramos, and J. Parra, "The Vocabulary Ontology Engineering for the Semantic Modelling of Home Services," in *ICEIS (3)'06*, 2006, pp. 461-466.

[38] T. Heider and T. Kirste, "Architecture Considerations for Interoperable Multi-modal Assistant Systems," presented at the Proceedings of the 9th International Workshop on Interactive Systems. Design, Specification, and Verification, 2002.

[39] W. W. W. C. (W3C). (2012). *Resource Description Framework*. Available: http://www.w3.org/RDF/

[40] Metadados. (2010). *O que são Metadados ?* . Available: http://www.metadados.pt/index.php/oquesaometadados

[41] K. Lee, T. Lunney, K. Curran, and J. Santos, "Ambient Middleware for Context-Awareness (AMiCA)," University of Ulster2008.

[42] C. Toolkit. (2010). *Context Toolkit* Available: http://www.contexttoolkit.org/

[43] Microsoft. (2012). *Windows Azure*. Available: http://www.windowsazure.com/en-us/

[44] O. Corporation. (2012). *Sun SPOT World*. Available: http://www.sunspotworld.com/

[45] W. W. W. C. (W3C), "OWL Web Ontology Language," 2004.

[46] U. Consortium. (2012). *Project Description* Available: http://www.universaal.org/en/about/project-description

[47] U. Consortium. (2012). *Objectives* Available: http://www.universaal.org/en/about/objectives

[48] U. Consortium, "universAAL Reference Architecture," 2011.

[49] U. Consortium, "universAAL Generic Platform Services, AAL platform services and ontology artefacts," 2011.

[50] U. Consortium, "Developers handbook " 2011.

[51] L. Fan, W. Buchanan, C. Thummler, O. Lo, A. Khedim, O. Uthmani, A. Lawson, and D. Bell, "DACAR Platform for eHealth Services Cloud," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, 2011, pp. 219-226.

[52] Microsoft. (2012). *Microsoft HealthVault*. Available: http://www.microsoft.com/en-gb/healthvault/default.aspx

[53] A. M. Khattak, P. T. H. Truc, L. X. Hung, L. T. Vinh, V. Dang, D. Guan, Z. Pervez, M. Han, S. Lee, and Y. Lee, "Towards Smart Homes Using Low Level Sensory Data," *Sensors (Basel)*, vol. 11, pp. 11581-11604, 2011.

[54] E. Farella, M. Falavigna, and B. Ricco, "Aware and smart environments: The Casattenta project," in *Advances in sensors and Interfaces, 2009. IWASI 2009. 3rd International Workshop on*, 2009, pp. 2-6.

[55] M. Klein, A. Schmidt, and R. Lauer, "Ontology-Centred Design of an Ambient Middleware for Assisted Living: The Case of SOPRANO," presented at the 30th Annual German Conference on Artificial Intelligence, German, 2007.

[56] A. Consortium. (2009). *The Project* Available: agnes-aal.eu/

[57] A. Association, "Catalogue of Projects," in *Ambient Assisted Joint Programme*, T. C. M. U. (CMU), Ed., ed. Belgium, 2011.

[58] T. Saito, I. Tomada, Y. Takabatake, J. Ami, and K. Teramoto, "Home gateway architecture and its implementation," in *Consumer Electronics, 2000. ICCE. 2000 Digest of Technical Papers. International Conference on*, 2000, pp. 194-195.

[59] J. E. L. d. Vergara, V. A. Villagrá, C. Fadón, J. M. González, J. A. Lozano, and M. Álvarez-Campana, "An autonomic approach to offer services in OSGi-based home gateways," *Comput. Commun.*, vol. 31, pp. 3049-3058, 2008.

[60] D. Vergados, A. Alevizos, A. Mariolis, and M. Caragiozidis, "Intelligent services for assisting independent living of elderly people at home," presented at the Proceedings of the 1st

international conference on PErvasive Technologies Related to Assistive Environments, Athens, Greece, 2008.

[61] T. Perumal, A. R. Ramli, C. Y. Leong, S. Mansor, and K. Samsudin, "Interoperability for Smart Home Environment Using Web Services," *International Journal of Smart Home,* vol. 2, October, 2008 2008.

[62] V. Miori, D. Russo, and M. Aliberti, "Domotic Technologies Incompatibility Becomes User Transparent," *Communications of the acm,* vol. 53, pp. 153-157, 2010.

[63] A. Bamis, D. Lymberopoulos, T. Teixeira, and A. Savvides, "The BehaviorScope framework for enabling ambient assisted living," *Personal Ubiquitous Comput.*, vol. 14, pp. 473-487, 2010.

[64] Z. Xiao Ming and Z. Ning, "An Open, Secure and Flexible Platform Based on Internet of Things and Cloud Computing for Ambient Aiding Living and Telemedicine," in *Computer and Management (CAMAN), 2011 International Conference on*, 2011, pp. 1-4.

[65] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *SIGSOFT Softw. Eng. Notes,* vol. 17, pp. 40-52, 1992.

[66] C. Software Enginnering Institute. (2012). *Software Architecture*. Available: http://www.sei.cmu.edu/architecture/?location=secondary-nav&source=19118

[67] Microsoft. (2012). *Software Architecture and Design*. Available: http://msdn.microsoft.com/en-us/library/ee658093

[68] R. Schollmeier, "[16] A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications," presented at the Proceedings of the First International Conference on Peer-to-Peer Computing, 2001.

[69] P. Duchessi and I. Chengalur-Smith, "Client/server benefits, problems, best practices," *Commun. ACM,* vol. 41, pp. 87-94, 1998.

[70] G. C. I. Centre. *GRID Infoware*. Available: http://www.gridcomputing.com/

[71] OASIS, "Reference Model for Service Oriented Architecture 1.0," ed, 2006.

[72] OASIS. (2011). *Reference Architecture Foundation for Service Oriented Architecture Version 1.0*.

[73] Y. Balzer. (2004). *Improve your SOA project plans*.

[74] Z. Yu Hua, Z. Jian, and Z. Wei Hua, "Discussion of a Smart House Solution Basing Cloud Computing," in *Communications and Intelligence Information Security (ICCIIS), 2010 International Conference on*, 2010, pp. 244-247.

[75] D. Jovevski, "Impact of Cloud computing on the business worldwide, the level of use in Macedonian companies," 2011.

[76] W3SCHOOLS. *Web Services Platform Elements*. Available: http://www.w3schools.com/webservices/ws_platform.asp

[77] W3SCHOOLS. *SOAP Tutorial*. Available: http://www.w3schools.com/soap/default.asp

[78] OASIS, "UDDI Spec Technical Committee Draft.," 2004.

[79] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," UNIVERSITY OF CALIFORNIA, IRVINE, 2000.

[80] W3SCHOOLS. *Introduction to XML*. Available: http://www.w3schools.com/xml/xml_whatis.asp

[81] W3SCHOOLS. *JSON Tutorial*. Available: http://www.w3schools.com/json/default.asp

[82] IBM. *Java 2 Security*. Available: http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=%2Frzamy%2F50%2Fsec%2Fjava2.htm

[83] O. Alliance. (2012). *Benefits of Using OSGi*. Available: http://www.osgi.org/Technology/WhyOSGi

[84] T. A. S. Foundation. (2012). *Introducing Axis2*. Available: http://axis.apache.org/axis2/java/core/docs/userguide.html#underhood

[85] S. Sahoo, "OSGi Application Development using GlassFish Server."

[86] E. Corporation. (2012). *What is a one-way function?* Available: http://www.rsa.com/rsalabs/node.asp?id=2188

[87] A. Teixeira, C. Pereira, M. O. e. Silva, J. Alvarelhão, A. Neves, and O. Pacheco, "Output Matters! Adaptable Multimodal Output for New Telerehabilitation Services for the Elderly," presented at the 1º International Living Usability Lab Workshop on AAL Latest Solutions, Trends and Applications in BIOSTEC 2011, Rome, Italy, 2011.

[88] J. Cunha, A. J. R. Neves, J. L. Azevedo, B. Cunha, N. Lau, A. Pereira, and A. Teixeira, "A Mobile Robotic Platform for Elderly Care," presented at the 1º International Living Usability Lab Workshop on AAL Latest Solutions, Trends and Applications in BIOSTEC 2011, Rome, Italy, 2011.

[89] C. G. Pires, F. M. Pinto, E. M. Rodrigues, and M. S. Dias, "On the Benefits of Speech and Touch Interaction with Communication Services for Mobility Impaired Users," presented at the 1º International Living Usability Lab Workshop on AAL Latest Solutions, Trends and Applications in BIOSTEC 2011, Rome, Italy, 2011.

[90] T. Lisboa. (2012). *Set-top Boxes*. Available: https://sites.google.com/site/televisaointeractiva/tecnolog/set-top-boxes

[91] J. Andrews and N. Baker, "Xbox 360 System Architecture," *IEEE,* 2006.

[92] S. Software. (2011). *What is soapUI*. Available: http://www.soapui.org/About-SoapUI/what-is-soapui.html

[93] S. Software. (2011). *What is loadUI?* Available: http://www.loadui.org/

7. Anexos

7.1. Anexo A – Serviço de Vídeo

7.1.1.1.1. Dependências

Para perceber este serviço é necessário conhecer o serviço de gestão de recursos do LULMS, já que a utilização deste depende do serviço de gestão de recursos. A câmara de vídeo é vista como um recurso.

7.1.1.1.2. Visão Geral

Este serviço permite que um especialista de saúde controle uma câmara de vídeo remotamente, podendo assim visualizar o paciente de forma mais autónoma.
Para aceder à câmara de vídeo existe a necessidade de comunicar com o LULMS e portanto com o serviço que permite gerir e visualizar recursos.
Depois de obtidas algumas informações, a aplicação do especialista de saúde passa a comunicar diretamente com o serviço da câmara de vídeo que está a visualizar o paciente.

7.1.1.1.3. Métodos do Serviço

ZoomIn

Aproxima a câmara de vídeo se esta permitir esta funcionalidade. Caso não permita, o resultado deste serviço é uma exceção que diz que a funcionalidade não é suportada pela câmara de vídeo. Não contém parâmetros de entrada.

ZoomOut

Afasta a câmara de vídeo se esta permitir esta funcionalidade, tal como no método anterior devolve exceção se esta funcionalidade não for suportada. Não contém parâmetros de entrada.

Zoom

Aproxima ou afasta a câmara de vídeo. Contém um parâmetro de entrada: o valor de *zoom.* Se o parâmetro de entrada for 1, aproxima a câmara de vídeo 1 nível em valor relativo, caso o valor for 2, aproxima a câmara de vídeo 2 valores. Se o valor for zero, volta ao *zoom default* da câmara. Se for negativo, afasta na mesma proporção descrita para a aproximação.

Move

Move a câmara para cima, para baixo, para a direita ou para a esquerda, dependendo da instrução dada.
Contém dois parâmetros de entrada: um que indica qual a direção do movimento da câmara e outro que indica a quantidade de deslocação. O primeiro parâmetro só aceita quatro letras (U – Up, D – Down, L – Left, R – Right) e o segundo parâmetro é um número.

GetImplementedServices

Devolve todas as operações que são implementadas pela câmara de vídeo. Assim o utilizador do serviço sabe com quais operações poderá contar antes de fazer os pedidos aos serviços.

Exceções

Para o caso de a câmara de vídeo não suportar alguma das operações o serviço deve devolver a informação de que a operação não é suportada.

Idealmente, todas as funcionalidades são suportadas pela câmara de vídeo. No entanto, uma câmara de vídeo pode não suportar uma operação específica.

Resposta do Serviço

Tal como em todos os outros serviços, a resposta é um objeto do tipo Operation Status. Caso a operação não seja suportada, a resposta terá o valor de *false* na componente *Sucess* e a respetiva exceção. Se for suportada, a resposta terá o campo *Sucess* a *true*.

7.1.1.1.4. Fluxo de trabalho do serviço

Para utilizar este serviço é necessário obter alguma informação relativa à câmara de vídeo.

Inicialmente, o especialista de saúde deve obter a informação necessária para poder estabelecer ligação com a câmara de vídeo. Depois de devidamente estabelecida, a câmara de vídeo deve ser adicionada à lista de recursos em uso na sessão.

Depois de estabelecida a ligação com a câmara de vídeo, o especialista pode controlá-la, utilizando o serviço descrito neste documento.

O diagrama seguinte ilustra o que foi explicado acima. Note-se que a câmara de vídeo é um recurso como qualquer outro para o LULMS.

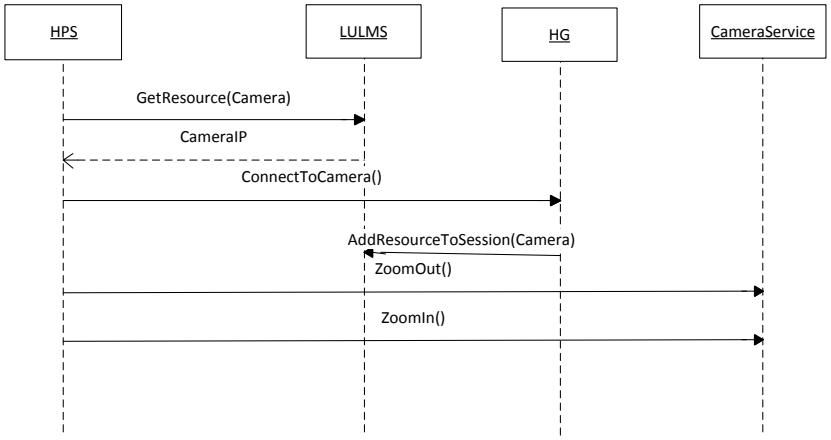
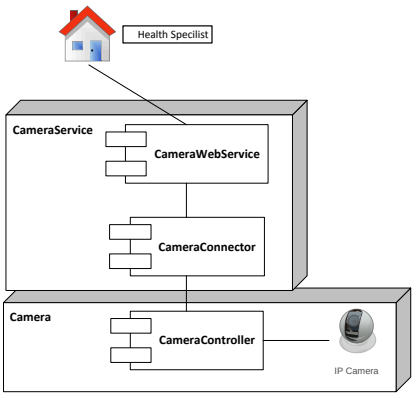


Diagrama de Sequencia

7.1.1.1.5. Arquitetura Interna

Como se pode constatar na figura abaixo, o serviço está dividido por três componentes. O primeiro componente será o ponto de comunicação entre os utilizadores do serviço e a aplicação (CameraWebService). O componente CameraConnector gere a ligação entre o controlador e o componente CameraWebService. E o último componente controla a câmara de vídeo e atua diretamente sobre as operações da mesma (CameraController). Este último componente apenas precisa de respeitar um determinado interface para que o sistema funcione. Assim é fácil trocar a câmara (alterando apenas o seu controlador) sem que a restante estrutura do serviço se altere, tornando assim o serviço extremamente flexível e desacoplado do *hardware.* O serviço é constituído pelos seguintes métodos (que coincidem com os métodos do interface que deve ser respeitado pelos controladores): (1) ZoomIn – aumenta o zoom da câmara; (2) ZoomOut – diminui o zoom da câmara; (3) Zoom – muda o zoom da câmara, dependendo de um parâmetro de entrada; (4) Move – move a câmara para cima, para baixo, para a direita ou para a esquerda, dependendo da instrução dada, ou seja, permite efetuar operações *tilt* e *span*; (5) GetImplementedServices – obtém quais os serviços que a câmara suporta dentro dos que foram listados acima (já que nem todas as câmaras são obrigadas a suportar todas as funcionalidades disponibilizadas pelo serviço).



Arquitetura Geral

7.1.1.1.6. Controlador da Câmara

Como exemplo foi criado um controlador para uma câmara IP. As únicas funcionalidades suportadas pela câmara são o *Tilt* e o *Span*. Pelo que apenas dois métodos dentro do serviço irão funcionar (o Move e o GetImplementedServices).

Substituição do Controlador

Para colocar um novo controlador é necessário substituir a classe *cameraDriver*, por uma classe que controle uma nova câmara de vídeo.

7.2. Anexo B – Serviço de Sensores

7.2.1.1.1. Dependências

Para perceber este serviço é necessário conhecer o serviço de gestão de recursos do LULMS, já que a utilização deste depende do serviço de gestão de recursos.

7.2.1.1.2. Visão Geral

O serviço de sensores é um serviço genérico que pretende abstrair o acesso a todos os sensores. Para isso, é necessário que cada sensor vá atualizando os seus dados, inserindo-os no serviço.

7.2.1.1.3. Métodos do Serviço

newStore

Este método permite criar um repositório para guardar dados enviados pelos sensores. É constituído por quatro parâmetros: *id*, *nChannels*, *channelNames*, e *maxBufSize*. O primeiro representa o identificador, o segundo representa o número de canais que serão criados para essa *store*, o terceiro é o nome de cada um desses canais. O ultimo parâmetro representa o número máximo de dados que podem estar guardados em simultâneo.
Finalmente, o método retorna um resultado igual a verdadeiro se a operação tiver sucesso.

addToStore

Este método é utilizado para guardar dados enviados por vários sensores. É constituído por três parâmetros: *id*, *samples* e *data*. O primeiro representa o identificador do cliente (o numero inteiro que identifica o cliente), o segundo representa o número de amostras recebidas e o ultimo os dados que são enviados pelos sensores.
Finalmente, o método retorna um resultado igual a verdadeiro se a operação tiver sucesso.

getFromStore

Este método permite obter os dados guardados. Depois de obter a informação os dados serão eliminados. Este método recebe como parâmetros de entrada o *id*, o *channel* e as *nSamples*. O *id* representa o identificador do cliente, o *channel* é o canal de onde se quer obter os dados e o ultimo parâmetro representa o número de amostras que se quer retirar (se este valor for zero todas as amostras são obtidas).
Finalmente, o resultado deste método é a lista com a informação pedida.

getSensorsName

Este método permite obter o nome dos canais para um determinado cliente. Tem como parâmetro de entrada o identificador do cliente.

getClientId

Este método permite obter o número inteiro que identifica um cliente. Tem como parâmetro de entrada o nome do cliente.

peekFromStore

Este método é similar ao *getFromStore*. A diferença é que este mantém os dados no repositório.

deleteById

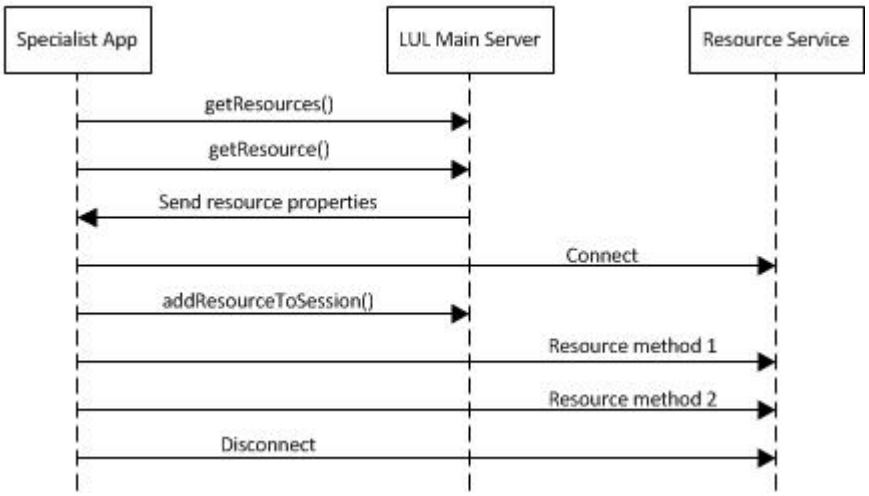
Este método permite eliminar um cliente do repositório. O método retorna um resultado igual a verdadeiro se a operação tiver sucesso.

Resposta do Serviço

Tal como em todos os outros serviços, a resposta é um objeto do tipo *Operation Status*. Caso a operação não seja suportada, a resposta terá o valor de *false* na componente *Sucess* e a respetiva exceção. Se for suportada, a resposta terá o campo *Sucess* a *true*.

7.2.1.1.4. Fluxo de trabalho do serviço

Para o registo de sensores ou qualquer outro tipo de dispositivo ao HS, é tratado de uma forma genérica, ou seja, como um recurso. A esse recurso são adicionados propriedades (método *addResourceProperty*), definidas por um nome e um valor. Deste modo, a cada recurso poderá ser associado tantas propriedades quantas forem necessárias. Um exemplo de uma propriedade seria o endereço IP ou a descrição do recurso.



Após o registo, as aplicações que necessitem de aceder à informação dos sensores terão que efetuar um pedido ao servidor de modo a obter uma listagem de sensores disponíveis (método getResources) para obter a informação do endereço IP para acesso aos mesmos, dentro do HG.

Após estabelecido o acesso com o HG e os sensores, será possível obter a informação destes, para posteriormente ser apresentada das mais diversas formas.

7.3. Anexo C – Serviço de Conexão

7.3.1.1.1. Visão Geral

O serviço de conexão é um serviço que pretende abstrair o acesso ao LULMS. Este serviço tem como objetivo ser utilizado por outros serviços/aplicações que pretendam de alguma forma aceder ao LULMS. Permite realizar conexões, adicionar recursos, recuperar sessões e alterar as configurações de acesso ao LULMS.

7.3.1.1.2. Métodos do Serviço

connect

Este método permite estabelecer conexão com o LULMS. Assim, o LULMS fica com a informação de que o HS está *online* e permite o acesso a recursos da HS por parte de utilizadores externos. Não contém parâmetros de entrada. O método retorna um resultado igual a verdadeiro se a operação tiver sucesso.

disconnect

Este método permite terminar conexão com o LULMS. Assim, o LULMS fica com a informação de que o HS está *offline* e não permite o acesso a recursos da HS por parte de utilizadores externos. Não contém parâmetros de entrada. O método retorna um resultado igual a verdadeiro se a operação tiver sucesso.

.

endSession

Este método permite terminar uma sessão que esteja a decorrer. Ao terminar a sessão o LULMS ficará com a informação de que o utilizador já não devia estar em sessão. O principal objetivo deste método é fazer com que caso uma sessão termine sem que este método seja utilizado tanto o HG como o LULMS percebam que algo inesperado ocorreu. Não contém parâmetros de entrada.
Finalmente, método retorna um resultado igual a verdadeiro se a operação tiver sucesso.

restoreSession

Este método permite restaurar uma sessão. Quando este método é chamado caso uma sessão tenha sido terminada inconvenientemente o LULMS irá retornar os recursos que estavam em utilização, permitindo assim que o HG restaure esses recursos e os disponibilize para utilização. Como parâmetros de entrada este método tem apenas o outro utilizador que estava em sessão. O método retorna como resultado a lista de recursos que estavam em utilização, e a informação de se foi possível restaurar a sessão ou não.

registerResource

Este método permite registar um recurso no LULMS. Os recursos do HS devem estar registados no LULMS para que um utilizador que pretenda aceder à HS consiga obter uma lista dos recursos que tem disponíveis rapidamente. O HG é responsável pelo registo destes recursos sempre que novos recursos aparecem. O método retorna um resultado igual a verdadeiro se a operação tiver sucesso.

addResourceProperties

Este método permite adicionar propriedades de um recurso no LULMS (por exemplo o IP). O método retorna um resultado igual a verdadeiro se a operação tiver sucesso.

addIP

Este método permite adicionar o IP do HG ao ficheiro de configurações. Este valor é utilizado pelos restantes métodos.

addUserName

Este método permite adicionar o nome de utilizador do HG ao ficheiro de configurações. Este valor é utilizado pelos restantes métodos.

addPassword

Este método permite adicionar a *password* do HG ao ficheiro de configurações. Este valor é utilizado pelos restantes métodos.

Resposta do Serviço

Tal como em todos os outros serviços, a resposta é um objeto do tipo Operation Status. Caso a operação não seja suportada, a resposta terá o valor de *false* na componente *Sucess* e a respetiva exceção. Se for suportada, a resposta terá o campo *Sucess* a *true*.

7.3.1.1.3. Fluxo de trabalho do serviço

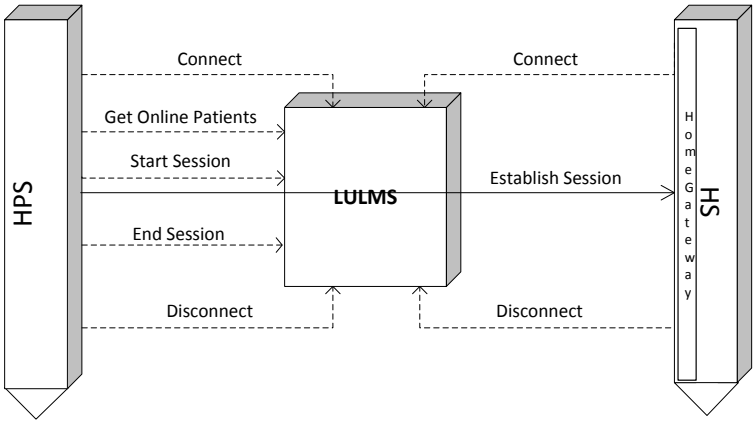
Registo de recursos

Todos os recursos existentes na HS deverão ser registados no LULMS para que, as entidades que exijam o acesso a informação dos mesmos, possam obter do servidor informação sobre os recursos, tais como o endereço IP, o seu estado ou algum tipo de identificação do mesmo. Este registo deve ser feito pelo HG.

Fluxo de estabelecimento de sessão

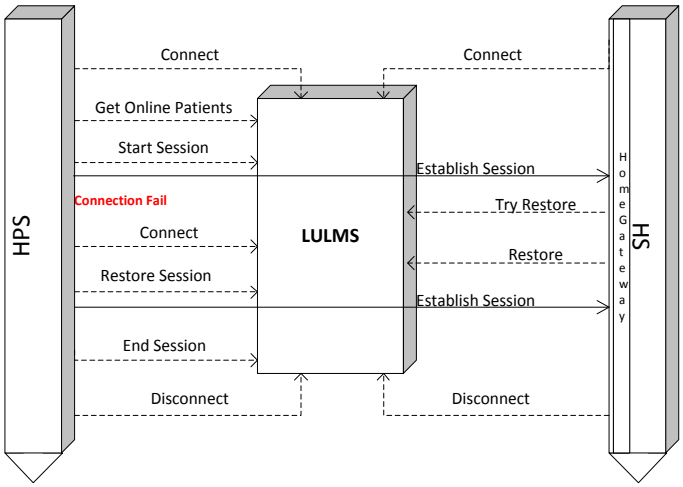
Fluxo normal

A figura abaixo representa o procedimento esperado de estabelecimento de sessão entre o HPS e o HS. Inicialmente ambos têm de se conectar com o LULMS. Quando ambos estão conectados é responsabilidade do HPS dar inicio a uma sessão.



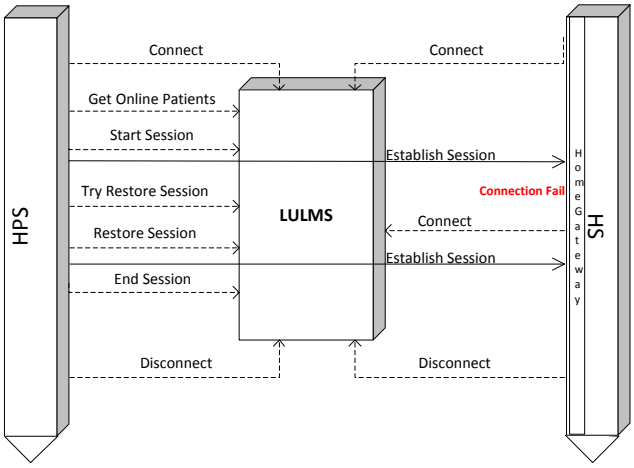
Fluxo com falha no HPS

Neste caso, está representado o fluxo de acontecimentos no caso de ocorrer uma falha no HPS. Neste caso, o HG deve restaurar a sessão e disponibilizar os recursos que estavam em utilização antes da falha.



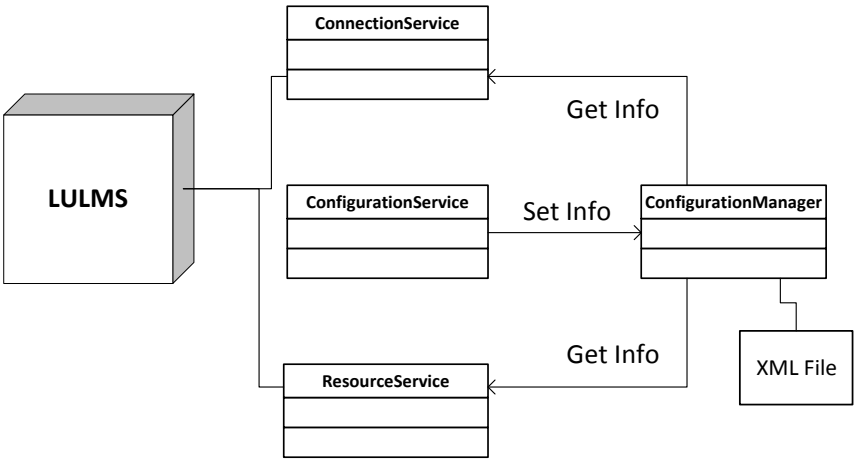
Fluxo com falha no HS

Neste caso, está representado o fluxo de acontecimentos no caso de ocorrer uma falha no HP. Neste caso, o HG deve conectar-se o mais rapidamente possível.



7.3.1.1.4. Arquitetura Interna

A figura abaixo representa as relações entre os vários elementos da arquitetura do serviço de conexões. Este serviço é essencialmente constituído por quatro componentes. O *ConnectionService* que é constituído por todos os métodos de conexão (*connect*, *disconnect*, *restore session*, *end session*), o *ConfigurationService* que é constituído pelos métodos de configuração de conexão (*add ip*, *add user name*, *add password*) e o *ResourceService* que é constituído por métodos de gestão de recursos (*register resource*, *add properties*). É ainda constituído pelo *ConfigurationManager* que atua diretamente sobre o ficheiro XML onde são guardadas as informações de conexão. Este ultimo é utilizado pelo *ConnectionService* e pelo *ResourceService* de forma a obter os dados de conexão. É ainda utilizado pelo *ConfigurationService* para escrever os dados.



7.4. Anexo D – Serviço de Exercícios

7.4.1.1.1. Dependências

Para perceber este serviço é necessário conhecer o serviço de gestão de recursos do LULMS, já que a utilização deste depende do serviço de gestão de recursos.

7.4.1.1.2. Visão Geral

O serviço de exercícios permite a um especialista de saúde localizado no HPS controlar a execução de exercícios. Este serviço permite que outros serviços se registem. Sempre que é enviado um comando para o serviço ele propaga esse comando para todos os outros serviços que estão registados.

7.4.1.1.3. Métodos do Serviço

getExerciseList

Este método permite obter a lista de exercícios que vão ser executados. Não contém parâmetros de entrada. Finalmente, o método retorna como resultado a lista de exercícios.

getPostUrl

Este método é utilizado para obter todos os *URL* registados. Não tem parâmetros de entrada. Como resultado, devolve a lista com todos os *URL* registados.

goToAndPlay

Este método permite ir para um determinado exercícios e coloca-lo em execução. Ao receber as instruções de execução vai enviá-las para todos os *URL* registados através de uma mensagem *HTTP POST*. Como parâmetros de entrada recebe o número do exercício que quer colocar em execução. Finalmente, o resultado deste método é a lista com a informação pedida. Finalmente, o resultado é igual a verdadeiro se a operação tiver sucesso.

play

Este método envia um comando para todos os *URL* registados através de uma mensagem *HTTP POST* informando que devem colocar o exercício atual (o próximo exercício, ou o primeiro se ainda nenhum tiver sido executado) a correr. Como resultado devolve verdadeiro se a operação tiver sucesso.

setExerciseList

Este método permite adicionar a lista de exercícios que vão ser executados. Ao receber a lista de exercícios vai enviá-la para todos os *URL* registados através de uma mensagem *HTTP POST*. Como parâmetros de entrada aceita uma lista de exercícios. Finalmente, o método retorna um resultado igual a verdadeiro se a operação tiver sucesso.

setPostUrl

Este método regista um determinado *URL* passando este a estar na lista de *URL* para onde o serviço enviará a mensagem.

stop

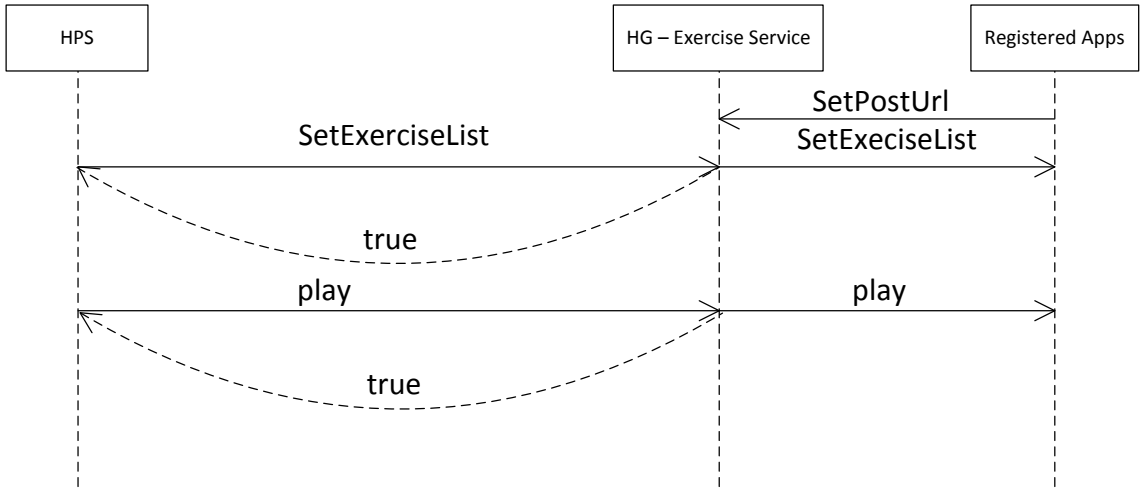
Este método envia um comando para todos os *URL* registados através de uma mensagem *HTTP POST* informando que devem para o exercício que está a correr. Como resultado devolve verdadeiro se a operação tiver sucesso.

Resposta do Serviço

Tal como em todos os outros serviços, a resposta é um objeto do tipo Operation Status. Caso a operação não seja suportada, a resposta terá o valor de *false* na componente *Sucess* e a respetiva exceção. Se for suportada, a resposta terá o campo *Sucess* a *true*.

7.4.1.1.4. Fluxo de trabalho do serviço

A figura abaixo representa um fluxo de tarefas do serviço. Inicialmente uma ou várias aplicações/serviços do HS têm de inserir o seu *URL*. Seguidamente, do HPS é enviado um pedido que insere os exercícios. Esse pedido é depois propagado para todos os *URL* registados. De seguida é enviado um pedido para começar a execução. Neste caso o primeiro exercício da lista é o que é executado. Caso não ocorra uma operação de pausa, os exercícios correm até que a lista termine.



7.5. Anexo E – Serviços do HG

7.5.1.1.1. Interações entre serviços

O HG é constituído por vários componentes/serviços. Estes serviços comunicam entre si de forma a manter a informação atualizada.

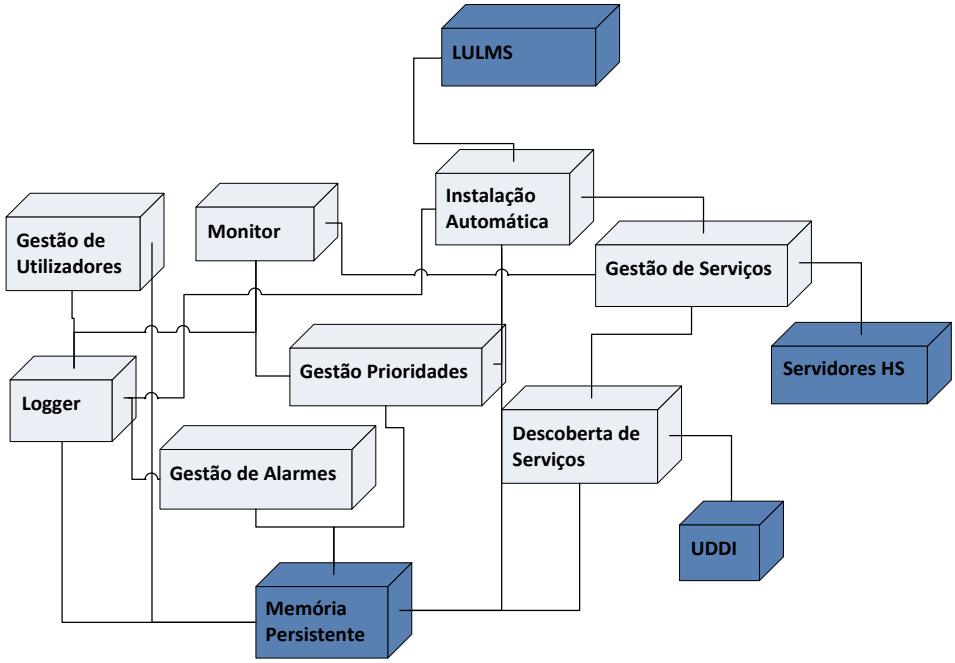
O serviço de instalação automática obtém os dispositivos da habitação (podendo estes ser inseridos manualmente). Depois disso, comunica com o LULMS de forma a obter os novos serviços. Insere os serviços no sistema recorrendo ao serviço de gestão de serviços e associa o nível de prioridades no serviço de gestão de prioridades. Por fim faz o registo no serviço no *logger*.

O monitor monitoriza todos os serviços do sistema e em alguns casos utiliza o serviço de prioridades para tomar decisões. Utiliza também o serviço de gestão de serviços quando pretende desligar ou inicializar um serviço. Regista no *logger* todas as decisões que toma.

A gestão de serviços comunica com os serviços de forma a colocar lá os novos serviços e informa o serviço de descoberta de serviços que existe um novo serviço. Este serviço (descoberta de serviços) regista esse serviço na UDDI ou na memória persistente do sistema.

O serviço de gestão de prioridades, gestão de alarmes e gestão de utilizadores registam toda a informação obtida na memória persistente. Os dois últimos registam ainda no *logger* as operações que efetuam.

Por fim, o serviço de gestão remota, que não está representado na figura abaixo, comunica com todos os outros serviços possibilitando a utilizadores externos controlar todo o sistema.



7.5.1.2. Serviço de Gestão de Utilizadores

7.5.1.2.1. Alguns detalhes

- De momento é utilizado um controlador que guarda os dados em XML, no entanto, é fácil trocar esse controlador para que passe utilizar qualquer tipo de armazenamento (MySql, Sql, entre outros).
- Utilização de uma *one-way-function* para encriptar dados confidências do utilizador.
- Para autenticar tem de se utilizar a mesma *one-way-function* e comparar com o resultado encriptado guardado.
- Cada *role* deve ter um conjunto regras associadas. Os restantes serviços devem gerir a autenticidade dos utilizadores recorrendo aos métodos disponibilizados.

7.5.1.2.2. Métodos do Serviço

addUser

Este método adiciona um novo utilizador ao sistema. Como parâmetros de entrada aceita o nome de utilizador, a *password*, o papel, e o e-mail. Devolve o resultado verdadeiro se o utilizador for criado com sucesso. Este serviço requer que um utilizador esteja autenticado como administrador.

removeUser

Este método remove um utilizador do sistema. Como parâmetros de entrada aceita o nome de utilizador a remover. Devolve o resultado verdadeiro se o utilizador for removido com sucesso. Este serviço requer que um utilizador esteja autenticado como administrador.

7.5.1.2.3. addRole

Este método permite adicionar um novo papel e respetivas autorizações. Devolve o resultado verdadeiro se o papel for adicionado com sucesso. Este serviço requer que um utilizador esteja autenticado como administrador.

7.5.1.2.4. removeRole

Este método permite remover um papel e respetivas autorizações. Devolve o resultado verdadeiro se o papel for removido com sucesso. Este serviço requer que um utilizador esteja autenticado como administrador.

7.5.1.2.5. getUserRole

Este método permite obter o papel de um utilizador. Como parâmetro de entrada recebe o nome de utilizador.

authenticate

Este método autentica um utilizador. Como parâmetros de entrada aceita o nome de utilizador e respetiva *password*. Como resultado devolve o valor verdadeiro se a autenticação for efetuada com sucesso.

7.5.1.3. Serviço de *Logger*

7.5.1.3.1. Alguns detalhes

- De momento é utilizado um controlador que guarda os dados em XML, no entanto, é fácil trocar esse controlador para que passe utilizar qualquer tipo de armazenamento (MySql, Sql, entre outros).

7.5.1.3.2. Métodos do Serviço

addRegist

Este método adiciona um novo registo ao *logger*. Como parâmetros de entrada aceita o assunto e o conteúdo. Devolve o resultado verdadeiro se o registo for criado com sucesso.

getRegists

Este método obtém todos os registos do *logger*.

getRegistsByDate

Este método obtém todos os registos do *logger* dentro de uma determinada data. Como parâmetro de entrada aceita a data inicial e a data final.

7.5.1.4. Serviço de Gestão de Alarmes

7.5.1.4.1. Alguns detalhes

- De momento é utilizado um controlador que guarda os dados em XML, no entanto, é fácil trocar esse controlador para que passe utilizar qualquer tipo de armazenamento (MySql, Sql, entre outros).
- Este serviço é constituído por uma *thread* que tem de ser colocada em execução.
- Em alguns casos os alarmes são executados no momento em que são inseridos mas noutros casos podem ser guardados para executar mais tarde.
- Sempre que "é tempo" de enviar um alarme o serviço envia-o através de *HTTP POST* para todos os serviços/aplicações que estão registados para o receber.

7.5.1.4.2. Métodos do Serviço

activate

Este método permite colocar o serviço de gestão de alarmes em execução. Como resultado devolve o valor verdadeiro se a operação for efetuada com sucesso.

setAlarm

Este método permite inserir um novo alarme. Como parâmetros de entrada aceita o nome do alarme, o conteúdo do alarme, a data de execução e o tipo de alarme. Como resultado devolve o valor verdadeiro se a operação for efetuada com sucesso.

registerToReceive

Este método permite a uma aplicação/serviço registar-se para receber os alarmes. Como parâmetro de entrada aceita o *URL* para onde se pretende enviar a mensagem e o tipo de alarme que pretende receber (se este valor estiver vazio é considerado que pretende receber todos os tipos de alarmes). Como resultado devolve o valor verdadeiro se a operação for efetuada com sucesso.

createAlarmType

Este método permite criar um tipo de alarme. Como parâmetro de entrada aceita o nome do novo tipo. Este serviço requer que um utilizador esteja autenticado como administrador.

getAlarmTypes

Este método obtém todos os tipos de alarme registados.

removeAlarmType

Este método permite remover um tipo de alarme. Como parâmetro de entrada aceita o nome do tipo de alarme. Este serviço requer que um utilizador esteja autenticado como administrador.

7.5.1.5. Serviço de *Monitor*

7.5.1.5.1. Alguns detalhes

- Este serviço é constituído por uma *thread* que tem de ser colocada em execução.
- Caso não seja possível escalar para a *Cloud* (que neste momento ainda não é) o serviço é responsável por desligar algum dos recursos com grau de prioridade mais inferior. Pode também, fazer o oposto e ligar um recurso caso seja possível (e caso tenha sido ele a desliga-lo).

7.5.1.5.2. Métodos do Serviço

activate

Este método permite colocar o serviço de gestão de alarmes em execução. Como resultado devolve o valor verdadeiro se a operação for efetuada com sucesso.

setAlarm

Este método permite inserir um novo alarme. Como parâmetros de entrada aceita o nome do alarme, o conteúdo do alarme, a data de execução e o tipo de alarme. Como resultado devolve o valor verdadeiro se a operação for efetuada com sucesso.

registerToReceive

Este método permite a uma aplicação/serviço registar-se para receber os alarmes. Como parâmetro de entrada aceita o *URL* para onde se pretende enviar a mensagem e o tipo de alarme que pretende receber (se este valor estiver vazio é considerado que pretende receber todos os tipos de alarmes). Como resultado devolve o valor verdadeiro se a operação for efetuada com sucesso.

createAlarmType

Este método permite criar um tipo de alarme. Como parâmetro de entrada aceita o nome do novo tipo. Este serviço requer que um utilizador esteja autenticado como administrador.

getAlarmTypes

Este método obtém todos os tipos de alarme registados.

removeAlarmType

Este método permite remover um tipo de alarme. Como parâmetro de entrada aceita o nome do tipo de alarme. Este serviço requer que um utilizador esteja autenticado como administrador.

7.5.1.6. Serviço de Prioridades

7.5.1.6.1. Alguns detalhes

- De momento é utilizado um controlador que guarda os dados em XML, no entanto, é fácil trocar esse controlador para que passe utilizar qualquer tipo de armazenamento (MySql, Sql, entre outros).
- Este serviço apenas armazena e calcula a prioridade do serviço. O cálculo é feito a partir dos valores de integridade, disponibilidade e confidencialidade.

7.5.1.6.2. Métodos do Serviço

setServicePriority

Este método permite inserir as configurações de prioridade de um serviço. Como parâmetros de entrada aceita o identificador do serviço, o nível de disponibilidade (entre 0 e 2), o nível de integridade (entre 0 e 2) e o nível de confidencialidade (entre 0 e 2). Como resultado devolve o valor verdadeiro se a operação for efetuada com sucesso.

getServicePriority

Este método permite obter a prioridade de um serviço. Como parâmetros de entrada aceita o identificador do serviço. Como resultado devolve o valor inteiro correspondente à prioridade do serviço.

7.5.1.7. Serviço de Instalação Automática

7.5.1.7.1. Alguns detalhes

- Este serviço é constituído por três *thread* que têm de ser colocadas em execução.
    - 1 – Verifica se existem novos dispositivos.
    - 2 – Verifica se existem novas aplicações.
    - 3 – Verifica se existem atualizações.
- De momento é utilizado um controlador que guarda os dados em XML, no entanto, é fácil trocar esse controlador para que passe utilizar qualquer tipo de armazenamento (MySql, Sql, entre outros).

7.5.1.7.2. Métodos do Serviço

activate

Este método permite colocar o serviço em execução, ativando todas as *thread*. Como resultado devolve o valor verdadeiro se a operação for efetuada com sucesso.

manualAddDevices

Este método permite inserir dispositivos manualmente. Caso um dispositivo não seja encontrado de forma automática. Como resultado devolve o valor verdadeiro se a operação for efetuada com sucesso. Esta operação só pode ser efetuada por utilizadores autorizados.

getDevices

Este método permite obter todos os dispositivos encontrados. Assim é fácil verificar se algum dispositivo não está a ser encontrado.

7.5.1.8. Serviço de Gestão de Serviços

7.5.1.8.1. Alguns detalhes

- Este serviço consegue obter os serviços que estão a correr no servidor e ativa-lo ou desativa-los. Tem ainda a capacidade de instalar novos serviços ou de desinstala-los.

7.5.1.8.2. Métodos do Serviço

desactivateService

Este método permite desativar um determinado serviço. Recebe o identificador do serviço como parâmetro de entrada. Como resultado devolve o valor verdadeiro se a operação for efetuada com sucesso. Este método requer que um utilizador esteja autenticado como administrador.

activateService

Este método permite ativar um determinado serviço. Recebe o identificador do serviço como parâmetro de entrada. Como resultado devolve o valor verdadeiro se a operação for efetuada com sucesso. Este método requer que um utilizador esteja autenticado como administrador.

addService

Este método permite instalar um novo serviço. Recebe o nome, o ficheiro executável do serviço,as propriedades de prioridade e o tipo de serviço como parâmetros de entrada. Como resultado devolve o valor verdadeiro se a operação for efetuada com sucesso, devolve ainda o identificador do serviço. Este método requer que um utilizador esteja autenticado como administrador.

getServices

Este método permite obter todos os serviços instalados. Este serviço requer que um utilizador esteja autenticado como administrador.

removeService

Este método remove um determinado serviço. Recebe como parâmetro de entrada o identificador do serviço. Este método requer que um utilizador esteja autenticado como administrador.

editService

Este método permite editar um determinado serviço. Recebe como parâmetro de entrada o identificador do serviço. Este método requer que um utilizador esteja autenticado como administrador

7.5.1.9. Serviço de Descoberta de Serviços

7.5.1.9.1. Alguns detalhes

- Este serviço abstrai o acesso a um repositório de serviços.
- Em alguns casos pode ser utilizado UDDI e noutros pode ser utilizado um repositório de dados persistentes.

7.5.1.9.2. Métodos do Serviço

getServicesInfo

Este método permite obter a informação completa de todos os serviços a correr no sistema. Como resultado devolve uma lista com descrição do serviço, tipo do serviço, identificador do serviço, endereço do serviço, prioridade do serviço, entre outros.

getUDDIUrl

Este método permite obter o URL da UDDI caso esta exista.

7.5.1.10. Serviço de Gestão Remota

- Este serviço abstrai o acesso a todos os serviços de gestão do HG.
- Permite obter os dados de acesso remoto (direto) aos vários servidores).
- Permite utilizar todos os serviços utilizados em cima.