Valter Fernando Carvalho Silva

Sistema Multi-Agente de Prestação de Serviços de Seguimento Adaptados aos Requisitos do Utilizador

Dissertação do Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Telecomunicações

Orientação: Prof. Doutora Benedita Malheiro

Instituto Superior de Engenharia do Porto
Departamento de Engenharia Electrotécnica
Rua Dr. António Bernardino de Almeida 431, 4200-072 Porto

Ano Lectivo: 2009-2010

Resumo

A crescente complexidade das aplicações, a contínua evolução tecnológica, o uso cada vez mais disseminado de redes de computadores e a ubiquidade do acesso têm impulsionado o desenvolvimento de sistemas distribuídos e de dispositivos fixos e móveis de características diversas.

Neste contexto, os sistemas distribuídos baseados em agentes que suportam a interacção entre componentes através da tecnologia *Web Services* constituem uma abordagem que assegura não só a distribuição, mas, sobretudo, a interoperabilidade entre componentes heterogéneos.

Em relação aos dispositivos, a popularidade e a evolução contínua das capacidades dos dispositivos móveis permitiram não só a migração para estes dispositivos de tarefas complexas que anteriormente apenas se realizavam em computadores de secretária ou em estações de trabalho, mas também a sua realização em qualquer lugar e a qualquer hora.

O objectivo desta dissertação é a criação de um sistema multi-agente provedor de meta-serviços de intermediação, continuidade de serviço e personalização face a serviços pré-existentes de posicionamento, navegação, seguimento e de informação meteorológica assim como a múltiplos dispositivos cliente. Os agentes de intermediação desenvolvidos, que possuem uma interface do tipo Serviço *Web*, levam em consideração as preferências dos utilizadores, as limitações dos dispositivos cliente e os serviços seleccionados, tornando transparente para os utilizadores o uso de diferentes tipos de dispositivos cliente. Por último, foram ainda desenvolvidas aplicações cliente para diversos tipos de dispositivos.

Para validar e avaliar o sistema foram definidos casos de uso dos principais serviços oferecidos e realizados diversos testes, tendo os resultados demonstrado a capacidade do sistema prestar de forma cabal e bastante satisfatória todos os serviços.

Por último, efectuam-se sugestões de melhoramento e apresentam-se direcções de desenvolvimento futuro.

Abstract

The increasing complexity of applications, the continuous technological evolution, the increasing dissemination of the use of computer networks and ubiquitous access have been driving the development of distributed systems and user devices.

In this context, the development of agent-based distributed systems that rely on the use of Web Services interfaces for interaction not only ensures distribution, but, above all, the interoperability between heterogeneous components.

As for devices, the popularity and the continuous evolution of mobile devices fostered the migration of complex tasks that previously only took place at desktops and workstations to these devices, allowing their execution anywhere at any time.

The goal of this master thesis is the creation of a multi-agent system which acts as a mediation, continuity of service and personalization meta-services provider for end-users of pre-existing positioning, navigation, tracking and weather services. The proxy agents, which have Web Services interfaces, take in consideration the preferences of the users, the limitations of the end-user device and the selected services, making transparent the usage of different types of devices. This project also involved the development of client applications for various types of devices.

To validate and evaluate the system's ability to fully provide all services, use cases were define, tests were conducted and results gathered, which demonstrated that the initial goals were fulfilled in a very satisfactory fashion.

Finally, improvement suggestions and future development directions are proposed.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Excertos de Código

Glossário

Abreviatura	Descrição	Definição
WS	*Web Service*	página 2
PDA	*Personal Digital Assistant*	página 2
ACL	*Agent Communication Language*	página 6
BDI	*Belief-Desire-Intention*	página 9
SMA	*Sistema Multi-agente*	página 13
TAC	*Trading Agent Competition*	página 14
IAD	*Inteligência Artificial Distribuída*	página 14
KQML	*Knowledge Query and Manipulation Language*	página 19
KIF	*Knowledge Interchange Format*	página 19
KSE	*Knowledge Sharing Effort*	página 19
ARPA	*Advanced Research Projects Agency*	página 20
FIFA	*Foundation for Intelligent Physical Agents*	página 21
JADE	*Java Agent Development Framework*	página 21
LPGL	*Library Gnu Public Licence*	página 21
JVM	*Java Virtual Machine*	página 22
JRE	*Java Runtime Environment*	página 22
JDK	*Java Development Kit*	página 22
API	*Application Programming Interface*	página 22
J2EE	*Java Enterprise Edition*	página 22
J2SE	*Java Standard Edition*	página 22
J2ME	*Java Mobile Edition*	página 22
P2P	*Peer-to-Peer*	página 23
JRMI	*Java Remote Method Invocation*	página 25
CORBA	*Common Object Request Broker Architecture*	página 25
SACI	*Simple Agent Communication Infrastructure*	página 25
IDE	*Integrated Development Environment*	página 26
W3C	World Wide Web Consortium	página 28
WSDL	*Web Services Description Language*	página 28
HTTP	*HyperText Transfer Protocol*	página 29
XML	*eXtensible Markup Language*	página 29
UDDI	*Universal Description Discovery and Integration*	página 29

Abreviatura	Descrição	Definição
SOAP	*Simple Object Access Protocol*	página 30
SLL	*Secure Socket Layer*	página 30
SGML	*Standard Generalized Markup Language*	página 30
RPC	*Remote Procedure Call*	página 31
FTP	*File Transfer Protocol*	página 32
SMTP	*Simple Mail Transfer Protocol*	página 32
URL	*Uniform Resource Locator*	página 32
MIME	*Multipurpose Internet Mail Extensions*	página 33
WSAI	*Web Service Agent Integration*	página 43
WSAG	*Web Service Agent Gateway*	página 43
WSIG	*Web Service Integration Gateway*	página 44
DF	*Directory Facilitator*	página 44
KVM	*Kilo Virtual Machine*	página 46
GPS	*Global Positioning System*	página 47
VM	*Virtual Machine*	página 49
AMS	*Agent Management System*	página 50
RMA	*Remote Management Agent*	página 51
CDDL	*Common Development and Distribution License*	página 57
CLDC	*Connected Limited Device Configuration*	página 58
MIDP	*Mobile Information Device Profile*	página 58
LWUIT	*Light Weight UI Toolkit*	página 58
GUI	*Graphical User Interface*	página 58
RI	*Implementação de Referência*	página 59
JSP	*Java Server Pages*	página 59

Capítulo 1

Introdução

Esta introdução apresenta ao leitor o contexto, os objectivos e a estrutura desta Dissertação.

1.1 Contextualização

A história da humanidade tem sido marcada por mudanças, de tal modo significativas, que acabaram por traçar decisivamente o rumo da sua evolução. Segundo Alvin Toffler, estamos a viver a terceira vaga de mudança também designada por sociedade da informação e que tem na Internet a sua fonte impulsionadora.

A expansão e a evolução da Internet desencadearam uma nova era no desenvolvimento de sistemas distribuídos. Desde então a comunidade de engenharia de *software* tem-se preocupado em oferecer técnicas e ferramentas adequadas ao desenvolvimento de *software* de acordo com diferentes paradigmas. Este trajecto cobriu diversos paradigmas como a programação estruturada, orientada a objectos, baseada em componentes e, mais recentemente, a abordagem baseada em agentes.

A computação baseada em agentes tem sido vista como a nova revolução no desenvolvimento de sistemas distribuídos. Estas entidades ou agentes têm vindo a ser utilizadas em diversas aplicações que vão desde sistemas pequenos e simples como filtros de *email*, a grandes e complexos sistemas como o controlo do tráfego aéreo.

Por outro lado, devido à necessidade de integração dos sistemas, surgiu o conceito de operação conjunta, onde sistemas heterogéneos devem ser capazes de estabelecer comunicação e partilhar dados, sem estarem ligados fisicamente

entre si, através de uma comunicação transparente. Assim, no início do século XXI, apareceu uma tecnologia que pretende derrubar definitivamente a barreira de comunicação entre plataformas heterogéneas, designada *Web Services* (WS) ou, em português, Serviços *Web*. Esta tecnologia revolucionou a programação de aplicações orientadas para a Internet, transformando-a numa arquitectura de *software* orientada a serviços.

Outro aspecto que deve ser mencionado nesta secção está relacionado com o facto de, hoje em dia, os dispositivos móveis como os telemóveis, *smartphones* e Personal Digital Assistants (PDA) serem bastante populares. Se, antes, estes dispositivos estavam restritos à execução de aplicações simples como, *e.g.*, calculadoras e calendários, actualmente isso já não acontece. Hoje em dia é possível aceder à Internet a partir destes dispositivos e realizar em qualquer lugar e a qualquer hora muitas das tarefas que até a alguns anos atrás apenas se poderiam realizar em computadores de secretária ou em estações de trabalho dedicadas.

1.2 Objectivos

Os objectivos que guiaram a realização da presente dissertação são os seguintes:

- Desenvolver um sistema multi-agente provedor de meta-serviços de intermediação, continuidade de serviço e personalização face a serviços pré-existentes de posicionamento, navegação, seguimento e de informação meteorológica assim como a múltiplos dispositivos cliente;
- Desenvolver aplicações cliente para vários tipos de dispositivos.
- Permitir aos utilizadores, de forma transparente, a utilização de diferentes tipos de dispositivos cliente tendo em conta as suas preferências, os serviços seleccionados e as limitações dos dispositivos;
- Armazenar as características relevantes dos diversos dispositivos e as preferências dos utilizadores e dos serviços activos numa base de dados;
- Aceder aos diferentes serviços pré-existentes de informação meteorológica, posicionamento, navegação, seguimento assim como providenciar serviços de valor acrescentado como, *e.g.*, o seguimento e gestão de frotas;
- Garantir um elevado nível de interoperabilidade através da adopção de interfaces do tipo serviço *Web*;

1.3 Estrutura da Dissertação

Este documento está dividido em seis capítulos e um anexo. Os capítulos apresentados são dirigidos para diferentes níveis de conhecimento. Esta divisão destina-se a que o documento seja perceptível para qualquer leitor, desde que tenha um nível de conhecimento tecnológico básico. O grau de conhecimento necessário para compreender os diversos capítulos aumenta de forma gradual ao longo do documento.

O Capítulo 2 do "Estado da Arte" é indicado para quem não esteja familiarizado com o contexto deste trabalho. É fornecida uma visão sobre os conceitos relacionados com Agentes e Sistemas Multi-agente assim como sobre algumas tecnologias que podem ser adoptadas.

O Capítulo 3, intitulado "Arquitectura", fornece uma visão de alto nível de todo o sistema, apresentando a arquitectura e justificando todas as opções de implementação efectuadas.

O Capítulo 4 de "Desenvolvimento" consiste numa abordagem de nível tecnológico onde se descreve a realização do sistema.

No Capítulo 5 de "Avaliação do Sistema" são descritos os testes efectuados, é explicada a adequação dos testes e são analisados e explicados os resultados obtidos.

No Capítulo 6 são sumariadas as principais conclusões e apresentadas algumas sugestões de melhoria e desenvolvimento do trabalho realizado.

No Anexo A são apresentados os diferentes tipos de mensagens do protocolo desenvolvido para assegurar a comunicação entre os módulos do sistema.

Capítulo 2

Estado da Arte

Neste capítulo apresentam-se os conceitos de agente, sistema multi-agente e as principais plataformas de desenvolvimento de sistemas multi-agente. Por último, introduz o conceito de Serviço Web.

2.1 Agente

A definição de agente pauta-se pela falta de consenso, uma vez que existem várias definições distintas. Uma possível razão para esta situação deve-se ao facto do termo agente ser uma designação de uso comum com inúmeros significados, levando assim a diversas definições quanto à sua utilização. A palavra "agente" é definida no dicionário com os seguintes significados: "pessoa que actua por ou que gere os negócios de alguém" ou "pessoa encarregada de praticar certas operações com um determinado objectivo por outrem" [1] definições estas que não estão muito distantes do que se entende por agente inteligente no meio da comunidade científica dos agentes computacionais.

No entanto, as definições de agente descobertas nalguma da literatura da especialidade não são tão coerentes entre si. Podemos constatar tal facto através dos seguintes exemplos:

- Segundo Stuart Russel e Peter Norvig [2], um agente é "algo que sente o mundo que o rodeia através de sensores, actuando nesse ambiente através de actuadores".
- Para Pattie Maes [3], um agente "é um sistema computacional que habita um dado ambiente, sente e age nesse ambiente, realizando um conjunto de objectivos ou tarefas para o qual foi projectado".

- De acordo agora com Michael Genesereth, "uma entidade apenas pode ser considerada um agente se comunicar correctamente através de uma linguagem de agentes como, *e.g.*, a Agent Communication Language (ACL)".

- Para a IBM[4], um agente "é uma entidade de *software* que cumpre um conjunto de instruções em representação de um utilizador ou um programa com um certo grau de independência ou autonomia que possui algum conhecimento ou representação dos desejos e objectivos desse utilizador".

- Já Michael Wooldrige e Nicholas Jennings, noutro artigo, [5] apresentam o termo agente dividido em dois níveis de definição: um mais "fraco" e mais abrangente e outro mais "forte" e mais específico.

Neste último caso, a definição mais "fraca" de agente é utilizada para descrever *software* com as seguintes características:

- Autonomia — capacidade de desempenhar funções autonomamente sem intervenção de seres humanos ou outros agentes, de controlo sobre as suas próprias acções e sobre o seu estado e que são desenvolvidos de forma a não bloquearem, mesmo perante a ocorrência de falhas;

- Capacidade de socializar — capacidade de interagir com o ambiente que o rodeia através da interpretação de mensagens numa linguagem que seja entendida pelas restantes entidades (outros agentes ou com humanos);

- Reactividade — capacidade de percepção do que ocorre no seu ambiente e de realizar acções de resposta quando requisitado por outras entidades;

- Capacidade Sensorial — capacidade de obter informação sobre o meio envolvente através de sensores;

- Iniciativa — capacidade de pro-actividade, *i.e.*, não se limita apenas a responder às alterações que ocorrem no seu ambiente, mas desencadeia, consoante os seus objectivos, acções oportunas para os atingir.

A definição mais "forte" de agente, além de possuir as características da noção anterior, engloba ainda os seguintes conceitos, que podem ser encontrados nos humanos:

- Aprendizagem — capacidade de adquirir conhecimento e alterar o seu comportamento com base em experiências anteriores;

- Crenças — convicções do agente sobre determinados factos ou valores que podem ser modificadas com o tempo;

- Intenções — acções que o agente planeia realizar no futuro;
- Obrigações — valores que o agente considera que é obrigado a cumprir;
- Emoções — valores que podem condicionar as acções ou o comportamento do agente;

Para além destas características, um agente pode apresentar ainda as seguintes propriedades:

- Mobilidade — capacidade do agente se movimentar entre máquinas ou sistemas computacionais diferentes através de uma rede;
- Veracidade — presunção que o agente não comunica intencionalmente informação falseada;
- Benevolência — presunção que o agente não possui objectivos contraditórios, procurando sempre realizar o que lhe é pedido;
- Racionalidade — capacidade do agente de realizar acções de forma a atingir os seus objectivos;
- Conhecimento — capacidade de raciocínio sobre o conhecimento representado.

Russel e Norvig apresentam em [2] uma perspectiva diferente sobre a eventual falta de coerência dos agentes, afirmando que "a definição de agente deve ser utilizada como uma ferramenta de análise de sistemas e não como uma caracterização rígida que divida o mundo em agentes e não-agentes". Esta nova perspectiva possibilita considerar o agente, não como uma característica de um programa ou sistema, mas com a forma de analisar e desenvolver sistemas computacionais.

2.2 Tipos de Agente

A grande variedade de aplicações baseadas na tecnologia dos agentes, a enorme dinâmica que tem caracterizado esta área de investigação e a dificuldade de definição do que é ou não um agente, levou os diversos investigadores a atribuir um grande número de sinónimos para melhor caracterizar o seu trabalho.

A fim de descrever melhor esta área científica e dividir os agentes em classes, diversas tipologias de agentes podem ser encontradas na literatura da especialidade.

A tipologia proposta Hyacinth Nwana em [6] é uma classificação de agentes de acordo com:

- Mobilidade — os agentes podem ser móveis, caso tenham a capacidade de se deslocar ao longo de uma rede de computadores passando de uma máquina para outra, ou estáticos, caso não possuam essa capacidade;

- Tipo de raciocínio — os agentes podem ser reactivos — não possuem qualquer modelo do mundo que os rodeia e reagem a estímulos segundo determinados comportamentos — ou deliberativos — utilizam um paradigma de pensamento deliberativo e possuem um modelo interno do mundo que os rodeia, permitindo-lhes planear as acções que tomam.

- Autonomia, cooperação e aprendizagem — estes três atributos dão origem a quatro tipos de agentes: cooperativos, cooperativos com aprendizagem, de interface e inteligentes (ver Figura 2.1).

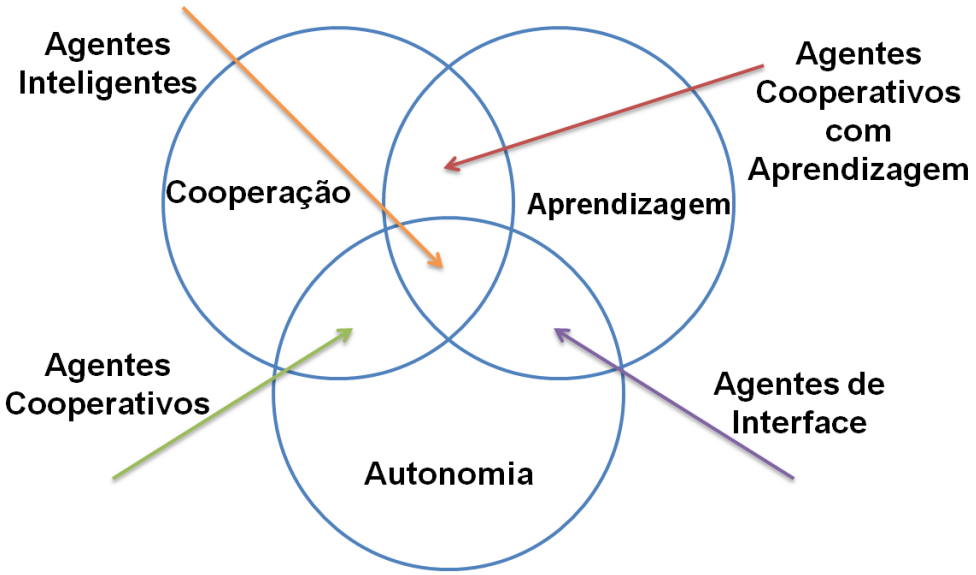

Figura 2.1: Tipos de Agente

No entanto estas fronteiras não são rígidas, *i.e.*, nada impede um agente cooperativo de aprender ou um agente cooperativo com aprendizagem de possuir algum grau de autonomia. Esta classificação apenas indica que, *e.g.*, nos agentes cooperativos a ênfase dada à cooperação e à autonomia é superior à dada à aprendizagem. O objectivo da investigação sobre agentes é a construção de agentes que possuam as três características principais igualmente bem desenvolvidas.

2.3 Arquitecturas de Agentes

Uma arquitectura pode ser descrita como um conjunto de módulos que interagem entre si.

Neste contexto, quando se fala da arquitectura, pode-se não só estar a referir à arquitectura interna de cada agente, mas também à arquitectura do próprio sistema multi-agente, *i.e.*, do modo de organização dos agentes dentro de um sistema e a forma como estes se relacionam e interagem.

De acordo com Wooldridge [5], pode-se dividir as arquitecturas de agentes em três grandes categorias: (*i*) arquitectura cognitiva ou deliberativa; (*ii*) arquitectura reactiva; e (*iii*) arquitectura híbrida. Além destas três grandes categorias, Wooldridge, apresenta ainda o conceito de arquitecturas por camadas (horizontais ou verticais). Neste tipo de arquitecturas, os vários agentes organizam-se em hierarquias e interagem por níveis. Outra arquitectura que merece ser referida é a arquitectura Belief-Desire-Intention (BDI).

2.3.1 Arquitectura Deliberativa

A arquitectura deliberativa é considerada uma arquitectura clássica de agentes. É caracterizada por possuir um modelo simbólico explícito do ambiente onde se encontra, *i.e.*, os agentes que tenham esta arquitectura possuem uma representação interna do mundo. As decisões dos agentes como, *e.g.*, a escolha da próxima acção a executar são realizadas através de raciocínio lógico, com base em padrões e manipulação simbólica. A Figura 2.2 apresenta um esquema representativo desta arquitectura.

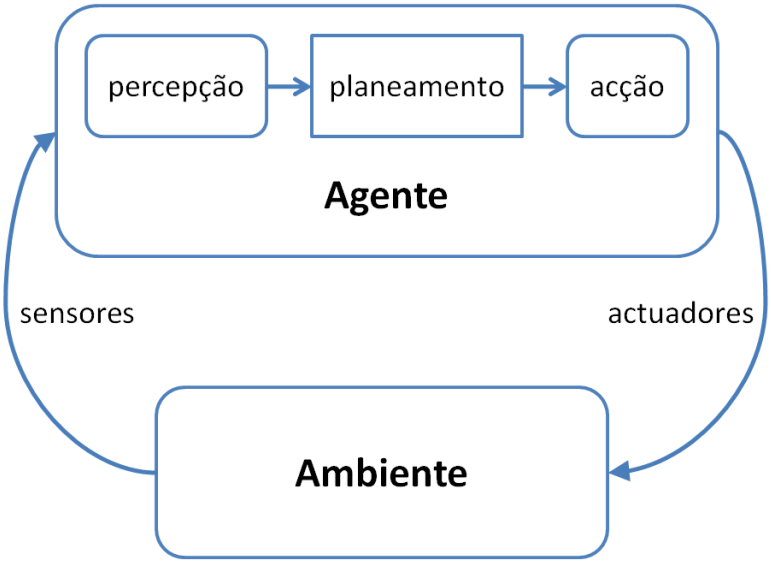

Figura 2.2: Arquitectura deliberativa

Neste tipo de arquitectura são levantados dois problemas importantes: (*i*) como traduzir o mundo real para uma descrição simbólica; e (*ii*) como utilizar a informação simbólica para decidir as acções a serem realizadas.

2.3.2 Arquitectura Reactiva

A arquitectura reactiva não utiliza qualquer tipo de modelo simbólico do ambiente nem nenhum mecanismo de raciocínio lógico complexo. Esta arquitectura baseia-se no conceito de que os agentes podem desenvolver inteligência através das interacções com o ambiente. As decisões tomadas pelos agentes que possuem este tipo de arquitectura são baseadas num conjunto de regras simples de situação/acção, sendo que a informação obtida através dos sensores é utilizadas na tomada das decisões, não sendo criada uma representação simbólica. A Figura 2.3 apresenta o diagrama da arquitectura reactiva.

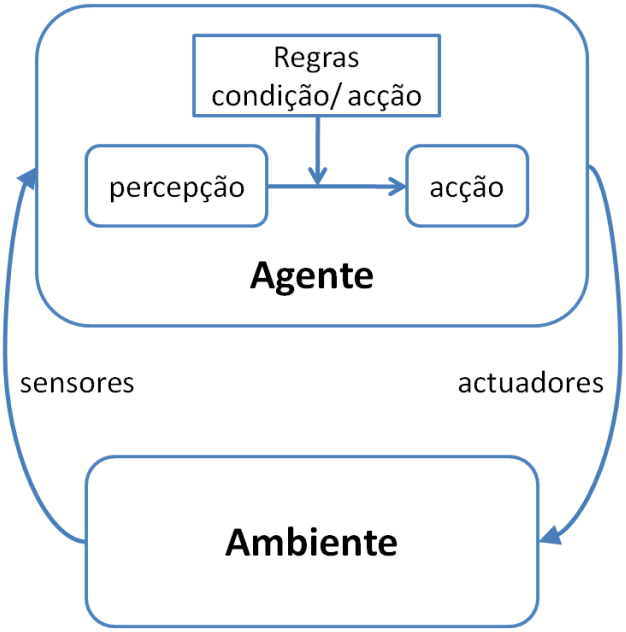

Figura 2.3: Arquitectura reactiva

Esta arquitectura apresenta como vantagens a simplicidade e robustez contra falhas e, como desvantagens, o facto dos agentes decidirem as acções baseados apenas na informação que possuem no momento e a incapacidade de realizarem acções que impliquem a execução de planos de longo prazo.

2.3.3 Arquitectura Híbrida

A arquitectura híbrida combina características da arquitectura deliberativa com características da arquitectura reactiva. Por um lado, os agentes puramente deliberativos, devido ao facto de utilizarem um modelo e raciocínio simbólico complexo, possuem uma dificuldade em reagir rapidamente a estímulos do exterior. Enquanto por outro lado, os agentes puramente reactivos possuem uma dificuldade em implementar comportamentos conduzidos por objectivos. Um agente híbrido combina estas duas componentes. Esta arquitectura é caracterizada por ser composta por níveis ou camadas dispostas de forma hierárquica permitindo assim estruturar as funcionalidades dos agentes. Normalmente, a camada reactiva possui alguma prioridade sobre a camada deliberativa para que seja possível responder rapidamente aos eventos que ocorram no ambiente. Um dos aspectos mais complexos deste tipo de arquitectura é a forma de efectuar a combinação das decisões deliberativas e reactivas e escolher a acção final a tomar. Um esquema da arquitectura híbrida é mostrado na Figura 2.4.

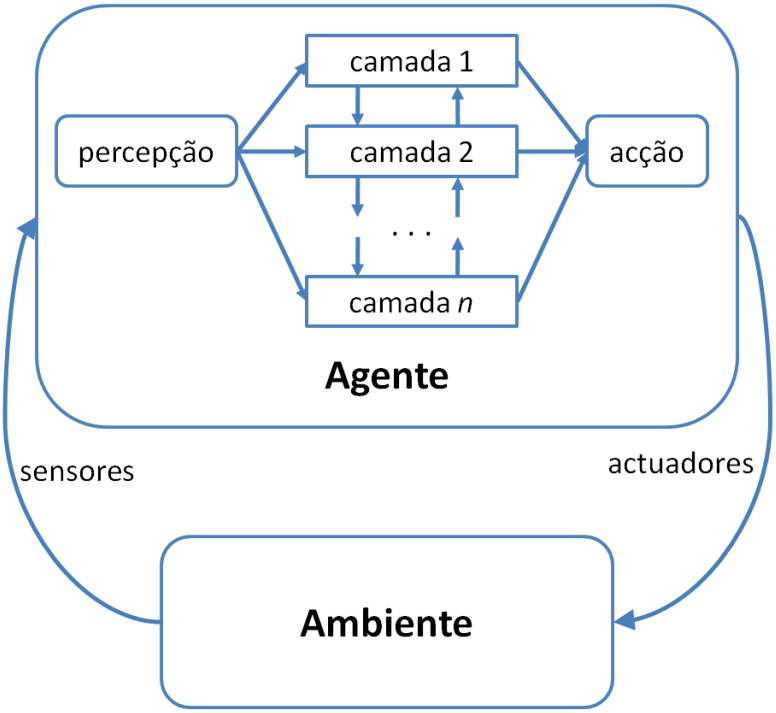

Figura 2.4: Arquitectura híbrida

2.3.4 Arquitectura BDI

A arquitectura BDI é uma arquitectura essencialmente deliberativa em que o estado interno de processamento de um agente é descrito através de um conjunto de "estados mentais": crenças ("beliefs"), desejos ("desires") e intenções ("intentions").

As **crenças** de um agente referem-se ao conjunto de suposições em que o agente acredita num dado momento. Constituem a perspectiva do agente sobre o estado do ambiente, *i.e.*, representa o seu conhecimento sobre o ambiente.

Os **desejos** de um agente correspondem aos seus objectivos. No entanto, o agente pode ainda não conhecer a forma como alcançará esses desejos. Um objectivo pode ser definido como um desejo que é consistente com o estado do agente e que, consequentemente, pode ser atingido.

As **intenções** referem-se a um conjunto de acções ou tarefas que o agente seleccionou e que o podem ajudar na concretização dos objectivos que possui.

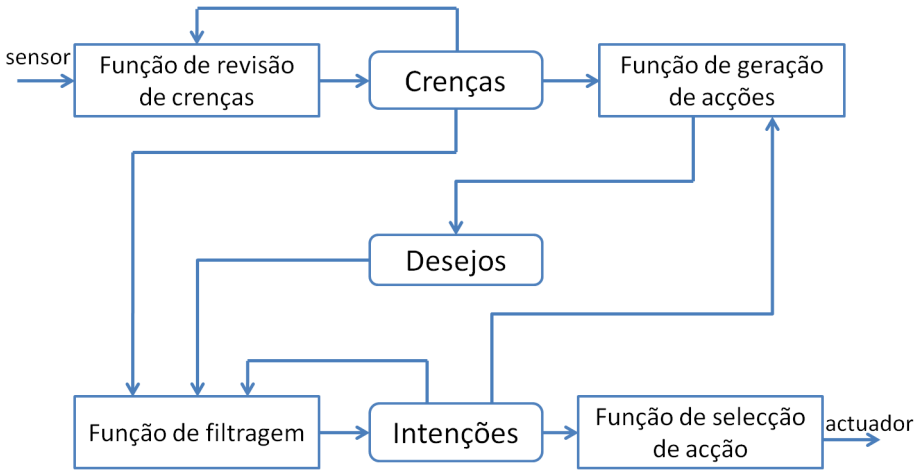

Figura 2.5: Arquitectura BDI

Segundo Wooldrige [5] e como se mostra na Figura 2.5, um agente BDI é constituído por sete componentes principais:

- Conjunto de crenças: que representam a informação que o agente possui sobre o ambiente onde se encontra;
- Função de revisão de crenças: que actualiza o conjunto de crenças do agente com base na informação recebida pelos sensores e nas crenças actuais do agente;
- Função de geração de opções: que, com base nas crenças e nas intenções actuais do agente, determina as suas opções ou desejos;
- Conjunto de opções: que representa os vários planos de acção possíveis que o agente possui ao seu dispor;
- Função de filtragem: que determina as intenções do agente com base nas suas crenças e desejos actuais;
- Conjunto de intenções: conjunto de estados que o agente pretende alcançar;
- Função de selecção de acção: que determina a acção a realizar com base nas intenções actuais do agente.

2.4 Sistema Multi-agente

Os Sistemas Multi-Agente (SMA) constituem uma área relativamente recente das ciências da computação. Apesar da investigação na área ter começado nos anos

80, apenas nos anos 90 ganhou notoriedade. Nos últimos anos, tem apresentado um elevado crescimento que levou ao aparecimento de artigos, revista, livros, conferências internacionais e até de uma rede europeia de computação baseada em agentes — AgentLink — responsável pela promoção e coordenação da investigação realizada no continente europeu nesta área. Outro aspecto que evidencia este pronunciado crescimento é a criação de competições internacionais de investigação como, *e.g.*, a Trading Agent Competition (TAC) que, desde 2000, promove a competição entre agentes individuais ou equipas de agentes, tendo como objectivo adquirir e fornecer os melhores pacotes de viagens aos seus clientes.

Foi no domínio da Inteligência Artificial Distribuída (IAD) que a investigação em sistemas baseados em agentes teve início. Este campo está dividido nas áreas de investigação da Resolução Distribuída de Problemas e dos Sistemas Multi-Agente. Enquanto a área de investigação de Resolução Distribuída de Problemas se preocupa com o modo como decompor e coordenar vários módulos de forma a resolver um problema específico, a área de investigação de Sistemas Multi-Agente (SMA) dedica-se a estudar sociedades de agentes que, em conjunto, conseguem resolver um problema que está para além das suas capacidades individuais [7] [8]. Pode também dizer-se que um Sistemas Multi-Agente é um sistema computacional constituído por dois ou mais agentes onde cada agente interage ou trabalha com os outros agentes de forma cooperativa, competitiva ou autónoma com o objectivo de atingir os seus próprios objectivos.

Daqui resulta que, num SMA, não se pode assumir que os agentes possuem um objectivo comum. Os vários agentes podem representar interesses e objectivos de diferentes entidades ou organizações dispares, tal como acontece na sociedade humana. No entanto, mesmo neste último caso, os agentes terão sempre de interagir para alcançarem os seus objectivos.

Logo, os agentes de um SMA devem ser capazes de agir de forma autónoma, tomando decisões que levam à concretização dos seus objectivos, ser capazes de interagir com outros agentes utilizando protocolos de interacção social e possuir funcionalidades de coordenação.

Um dos aspectos essenciais da construção de sociedades de agentes consiste na gestão das interacções e das dependências entre as actividades dos diferentes agentes no contexto do SMA. Desta forma, dado que se trata de sistemas inerentemente distribuídos, a coordenação desempenha um papel essencial. As principais metodologias de coordenação existentes podem ser divididas em dois grupos:

- Metodologias aplicáveis a agentes competitivos, *i.e.*, agentes que se preocupam apenas em atingir os seus próprios objectivos;
- Metodologias aplicáveis a agentes cooperativos, *i.e.*, agentes que se preocupam em atingir objectivos comuns.

No primeiro caso, a coordenação designa-se **negociação**. No segundo caso, a coordenação denomina-se **cooperação** e dedica-se à construção de equipas de agentes. Neste contexto, assumem particular relevância as metodologias que permitem definir uma organização estrutural da sociedade de agentes, a definição e troca de entidades, a definição de tarefas aos diversos agentes e o planeamento de forma conjunta das acções de cada agente.

O objecto da investigação em SMA é o desenvolvimento de abordagens e modelos computacionais que permitam construir, descrever, implementar e analisar as formas de interacção e coordenação de agentes em sociedades de reduzida ou elevada dimensão [9].

2.4.1 Motivação dos Sistemas Multi-agente

O principal motivo para a criação, desenvolvimento e utilização de SMA advém da necessidade de resolver problemas intrinsecamente distribuídos. Factores como a dimensão do problema ser demasiado elevada para ser resolvido por um único agente, a necessidade de interligar e garantir a interoperabilidade com sistemas pré-existentes e situações em que o conhecimento está disperso constituem motivos adicionais para a adopção deste paradigma. Segundo [10], as principais razões que levam ao desenvolvimento de SMA são:

- As características do próprio domínio do problema, *e.g.*, devido à distribuição espacial dos vários intervenientes;
- A exploração de paralelismo como mecanismo de aceleração da execução das tarefas necessárias, atribuindo as diferentes tarefas a diversos agentes em paralelo;
- A redução da comunicação através da transmissão de soluções parciais de alto nível em vez da comunicação amiúde de dados brutos;
- A robustez, já que quando se utilizam vários agentes não existe um ponto único de falha no sistema, *i.e.*, se um agente falhar os outros continuam a executar as suas tarefas e a tentar atingir os seus objectivos;
- A flexibilidade que advém de um sistema possuir agentes com diferentes especialidades que podem ser dinamicamente agrupados para resolver problemas;
- A escalabilidade que permite o aumento do número de agentes que compõem um determinado sistema;
- A simplificação de tarefas individuais de programação decorrente da subdivisão do problema global em vários subproblemas de menor complexidade;

- O estudo da inteligência individual e do comportamento social, já que os sistemas multi-agente permitem a interacção entre os agentes;
- A manutenção da privacidade da informação e do conhecimentos individuais que cada agente.

Embora os sistemas multi-agente apresentem múltiplas vantagens, também enfrentam muitos desafios. De acordo com [11], as principais questões que se colocam aquando da concepção e implementação de um SMA são as seguintes:

- Como formular, descrever, decompor e atribuir problemas e sintetizar os resultados a um grupo de agentes?
- Como permitir a comunicação e interacção entre agentes? Quais as linguagens e protocolos de comunicação? Como, o quê e quando comunicar?
- Como assegurar que os agentes agem de forma coerente na tomada de decisões?
- Como permitir que os agentes representem e raciocinem sobre acções, planos e conhecimento de outros agentes a fim de se coordenarem?
- Como reconhecer e resolver pontos de vista díspares e situações de conflito entre agentes que tentam coordenar as suas acções?
- Como, de forma efectiva, equilibrar a computação local e a comunicação, ou de forma mais genérica, como gerir a alocação de recursos limitados?
- Como evitar ou atenuar a possibilidade da ocorrência de comportamentos caóticos ou oscilatórios do sistema global?

2.4.2 Comunicação entre Agentes

Para que exista interacção entre agentes é obrigatório o estabelecimento de comunicação. A comunicação entre entidades computacionais foi desde sempre considerada uma das suas principais características, mas, também, um dos seus maiores problemas. No entanto, na área dos SMA, a comunicação é tratada a um nível muito mais elevado do que noutras áreas das ciências de computação, *i.e.*, a comunicação é realizada utilizando linguagens de comunicação de alto nível.

Uma forma de dotar um agente da capacidade de comunicação é incluir na sua arquitectura um módulo dedicado à comunicação que se pode subdividir nos componentes de percepção ou recepção de mensagens e de acção ou envio de mensagens. Segundo Huhns e Stephens [12], a comunicação entre agentes pode adoptar uma de duas arquitecturas básicas representadas nas Figuras 2.6 e 2.7,

respectivamente: (*i*) a comunicação directa; e (*ii*) a comunicação através de um agente "facilitador".

No método da comunicação directa, cada agente comunica directamente com qualquer outro agente sem utilizar intermediários. Para que a comunicação seja possível, existe a necessidade de partilha de especificações, *i.e.*, os agentes partilham as suas capacidades e/ou necessidades de forma a que cada gente possa individualmente comunicar com qualquer outro agente. Este método possui como desvantagem o facto de não possuir um elemento coordenador da comunicação, o que pode originar o bloqueio do sistema se, *e.g.*, todos os agentes decidirem enviar mensagens ao mesmo tempo.

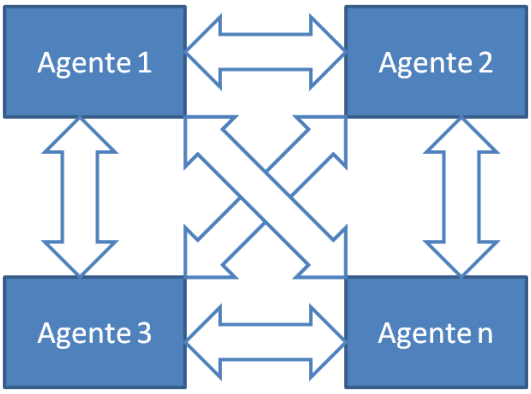

Figura 2.6: Arquitectura da comunicação directa

No método de comunicação através de agentes "facilitadores", os agentes recorrem a agentes facilitadores que são responsáveis pelo estabelecimento da comunicação entre agentes. Neste caso, se um dado agente n desejar enviar uma mensagem a um outro agente x, terá primeiro de a enviar ao "agente facilitador" e este, por sua vez, encaminhará a mensagem ao agente destinatário. Esta arquitectura de comunicação permite resolver de forma parcial o problema da coordenação da comunicação assim como diminuir consideravelmente a complexidade necessária aos agentes individuais para a realização da comunicação, já que os agentes não necessitam armazenar informação sobre todos os outros agentes presentes no seu ambiente. Uma das desvantagens da utilização deste método advém da centralização e estrangulamento provocado pelo agente facilitador no sistema de comunicações.

Figura 2.7: Arquitectura da comunicação através de um agente "facilitador"

A comunicação entre agentes está também relacionada com o nível de implementação das comunicações. Para além de uma arquitectura de comunicação é também necessário escolher o modo de comunicação entre os agentes. Existem duas formas principais de implementação: (*i*) a troca de mensagens entre agentes; e (*ii*) a memória partilhada por todos os agentes de um ambiente que também é designada *blackboard*.

A troca de mensagens entre agentes é a metodologia mais utilizada. No entanto, é sempre necessário considerar qual a arquitectura de comunicação que será implementada. Caso seja implementada uma arquitectura de comunicação directa, é necessário assegurar que os agentes podem receber mensagens constantemente assim como prever a possibilidade de uma mensagem enviada não ser recebida pelo destinatário. Se, por outro lado, for implementada a comunicação através de agentes facilitadores, os problemas referidos podem ser facilmente resolvidos se os agentes facilitadores possuírem a capacidade de armazenar as mensagens ainda não recebidas pelos agentes destinatários e, periodicamente, reenviá-las até que os agentes em questão as recebam. Esta última implementação implica uma menor eficiência na rapidez de transmissão das mensagens assim como da sua latência.

Na modalidade da memória partilhada os agentes nunca comunicam directamente. É criada uma estrutura de dados comum e através da escrita e leitura dessa estrutura todos os agentes podem comunicar entre si, efectuando, *e.g.*, anúncios ou divulgando resultados. Quando um agente pretende transmitir uma mensagem, escreve a mensagem na estrutura de dados e, uma vez escrita, a informação da mensagem fica automaticamente disponível para todos os restantes agentes. Apesar de já ter sido bastante utilizado, este modo de comunicação caiu um pouco em desuso visto que a centralização criada contraria a descentralização de controlo que se pretende nos sistemas multi-agente. Por um lado, a estrutura centralizada da memória partilhada, em caso de falha, compromete o funcionamento do sistema distribuído. Por outro lado, a concentração de acesso

a este tipo de estruturas implica um volume de tráfego que pode prejudicar o desempenho do sistema.

2.4.3 Linguagens de Comunicação entre Agentes

Segundo Russel, de uma forma geral, a comunicação é a troca intencional de informação através da produção e percepção de sinais de acordo com um sistema convencionado de símbolos [2]. Este sistema convencionado de símbolos designa-se linguagem e é utilizado pelos seres humanos para transmitir o conhecimento sobre o ambiente que os rodeia.

Da mesma forma que os seres humanos, também os agentes necessitam de uma linguagem comum. A comunicação entre agentes permite:

- Partilhar o conhecimento adquirido sobre o ambiente, evitando que os restantes agentes tenham de efectuar explorações já realizadas;
- Questionar os restantes agentes para obter conhecimento;
- Responder a questões que lhe sejam endereçadas por outros agentes;
- Solicitar a realização de acções a outros agentes;
- Efectuar propostas de execução de tarefas;
- Partilhar informação e experiências.

Como se pode constatar, a linguagem de comunicação entre agentes constitui um tema de especial importância dentro do estudo desta área científica.

De forma a dotar os agentes da capacidade de comunicação, foram desenvolvidas as linguagens de comunicação de agentes que possuem uma sintaxe e semântica limitadas e bem definidas. Esta estrutura rígida, embora limite a liberdade do discurso, permite uma comunicação mais eficiente e livre de ambiguidades. Entre as linguagens de comunicação entre agentes destacam-se as seguintes:

- Knowledge Query and Manipulation Language (KQML)
- Knowledge Interchange Format (KIF)
- Agent Communication Language (ACL)

KQML

A KQML é uma linguagem desenvolvida para suportar a interacção entre agentes. No início dos anos 90 foi criado o programa Knowledge Sharing Effort (KSE)

financiado pelo Advanced Research Projects Agency (ARPA) que tinha como objectivo o desenvolvimento de tecnologias para a partilha, troca e representação de informação entre sistemas de informação autónomos. A linguagem KQML foi um dos produtos desenvolvidos no âmbito deste programa. Esta linguagem permite transmitir informação descrita numa qualquer linguagem encapsulada em mensagens KQML. Outro aspecto desta linguagem é que esta ignora o conteúdo da mensagem, excepto no que diz respeito ao reconhecimento do seu início e fim, o que permite que uma mensagem pode comportar qualquer tipo de dados. Esta linguagem está dividida em três níveis:

Mensagem: comporta a identidade dos agentes emissor e receptor, a intenção do agente emissor (tipo de interacção entre agentes), o conteúdo da mensagem e o formato da informação;

Conteúdo: a informação que se pretende transmitir na linguagem de representação própria dos agentes envolvidos. A KQML aceita qualquer tipo de linguagem de representação de informação porque o conteúdo da mensagem é ignorado durante o processo de transmissão, ficando a interpretação do conteúdo a cargo do agente receptor;

Comunicação: adiciona à mensagem parâmetros de baixo nível como o nome e endereço dos agentes emissor e receptor, um identificador único de cada mensagem, *etc.*

Normalmente a linguagem de conteúdo utilizada em conjunto com a KQML é a linguagem KIF.

KIF

Esta linguagem, tal como a linguagem referida anteriormente, foi criada pelo programa KSE e desenvolvida como forma linguagem de definição do conteúdo de mensagens KQML, *i.e.*, a KIF é uma linguagem que se destina a representar o conhecimento de um domínio. Nesta qualidade, a KIF tem sido utilizada para a descrição de entidades em bases de dados ou agentes.

A linguagem KIF permite descrever os objectos, as propriedades e relações entre objectos de um domínio. Para representar estes conceitos, a KIF faz uso de operadores lógicos comuns como, *e.g.*, operadores booleanos e lógicos assim como estruturas típicas de dados como, *e.g.*, números e conjuntos de caracteres.

Esta linguagem permite ainda a descrição de procedimentos, ou seja, a descrição de um conjunto sequencial de acções a serem executadas pelos agentes.

ACL

A linguagem ACL foi desenvolvida pela Foundation for Intelligent Physical Agents (FIPA). Trata-se de uma instituição sem fins lucrativos que tem por objectivo produzir especificações para promover a interoperabilidade entre aplicações, serviços e equipamentos baseados nas tecnologias de agentes.

A ACL é bastante semelhante à linguagem KQML. No entanto, segundo a FIPA, a ACL foi simplificada para assegurar uma melhor organização e maior facilidade de utilização. Assim, alguns dos tipos de mensagens presentes na linguagem KQML não foram definidos na linguagem ACL. Tal como a linguagem KQML, a ACL define apenas a estrutura exterior da mensagem e não obriga à utilização de qualquer linguagem específica para o conteúdo.

Apesar das diferenças enunciadas entre as duas linguagens, elas são bastante idênticas, sendo relativamente fácil realizar uma transcrição quase directa entre ambas.

Por fim, para que haja uma interacção efectiva entre os agentes de um sistema, é necessário garantir que, não só, falam a mesma linguagem, mas, sobretudo, atribuem significados idênticos aos conceitos utilizados. Para isso, é necessário definir e adoptar uma Ontologia comum que especifique o significado dos conceitos utilizados.

2.5 Plataformas de Desenvolvimento de SMA

De seguida, apresentam-se algumas das plataformas de desenvolvimento de sistemas multi-agente desenvolvidas em linguagem Java mais utilizadas: JADE, Jini, JASON e JACK.

2.5.1 JADE

O Java Agent Development Framework (JADE) é uma plataforma de desenvolvimento de SMA, totalmente desenvolvida em linguagem Java e conforme com as especificações FIPA [13].

Esta plataforma foi criada pela Telecom Italia em 1998. No ano 2000, o JADE tornou-se uma plataforma *open source* sob a licença Library Gnu Public Licence (LPGL). Esta licença concede o direito de acesso ao código fonte, de desenvolvimento de novas funcionalidades e de incorporação do JADE em programas proprietários.

O objectivo é simplificar o desenvolvimento de SMA de acordo com os padrões da FIPA. Algumas das especificações impostas pela FIPA incluem os serviços de nomes (*naming service*), de páginas amarelas (*yellow pages*), de transporte de mensagens, de codificação e descodificação de mensagens e uma biblioteca de protocolos de interacção pronta a ser utilizada. O JADE é, na verdade, um *middleware* que facilita a criação, depuração e gestão de agentes assim como a comunicação entre agentes. A plataforma de agentes pode ser distribuída através de várias máquinas (que não necessitam de terem instalado o mesmo sistema operativo) e a sua configuração pode ser controlada através de uma única interface gráfica (GUI) remota. A configuração pode ser alterada, mesmo em tempo de execução, por agentes que se deslocam entre máquinas sempre que necessário. Dado que o JADE é completamente implementado em linguagem Java, requer apenas a pré-instalação da Java Virtual Machine (JVM) que é parte integrante do Java Runtime Environment (JRE) — ambiente de execução de Java — ou, alternativamente, do Java Development Kit (JDK) — o ambiente de desenvolvimento de Java. O JADE baseia-se em quatro princípios fundamentais:

Interoperabilidade — é compatível com as especificações FIPA, *i.e.*, os agentes desenvolvidos em JADE podem interagir com quaisquer outros agentes FIPA;

Uniformidade e portabilidade — fornece um conjunto de Application Programmable Interfaces (API) independentes da rede subjacente e da versão do Java, *i.e.*, durante o tempo de execução o JADE fornece sempre as mesmas API, tanto para o Java Enterprise Edition (J2EE), Java Standard Edition (J2SE) como para o ambiente Java Mobile Edition (J2ME);

Facilidade de utilização — a complexidade do *middleware* está escondida através de uma API simples e intuitiva;

Filosofia *Pay-as-you-go* — os programadores não precisam de utilizar todos os recursos que o *middleware* JADE disponibiliza, nem de configurar os recursos fornecido pelo JADE que não são utilizados.

Modelo de Arquitectura

O JADE inclui as bibliotecas (pacotes de classes Java) de desenvolvimento de agentes e de criação do ambiente de execução, *i.e.*, disponibiliza os serviços básicos necessários à execução dos agentes. Cada instância do tempo de execução JADE é designada contentor (uma vez que "contêm" agentes). O conjunto de todos os contentores é chamado plataforma e fornece uma camada homogénea que esconde aos agentes (e também aos programadores da aplicação) a complexidade e diversidade subjacentes, *e.g.*, dos vários tipos de rede ou sistemas operativos.

Modelo Funcional

Do ponto de vista funcional, o JADE fornece os serviços básicos necessários para a execução de aplicações distribuídas do tipo *peer-to-peer* (P2P) tanto em ambiente fixo como móvel. Alguns dos serviços básicos disponibilizados aos agentes incluem o seu registo, a alteração dos serviços que fornecem, a pesquisa de agentes que prestem um determinado serviço, o controlo do seu ciclo de vida e a comunicação com outros agentes. A descoberta, de forma dinâmica, dos agentes que prestam os serviços desejados e a possibilidade de, automaticamente, estabelecer a comunicação directa segundo o paradigma P2P é uma característica essencial da plataforma. Do ponto de vista da aplicação, cada agente é identificado por um nome único e fornece um determinado conjunto de serviços. Assim, o JADE oferece:

- Conformidade total com as especificações FIPA através da utilização da linguagem Java e das API disponibilizadas;
- Conjunto de ferramentas gráficas que facilitam a depuração e a monitorização dos agentes;
- API de localização que constitui a infra-estrutura de comunicação entre agentes e permite a comunicação de forma transparente entre agentes;
- Transporte eficiente de mensagens assíncronas, cabendo à plataforma seleccionar os melhores meios de comunicação disponíveis e, sempre que as mensagens atravessam as fronteiras da plataforma, converter automaticamente a sua representação interna do JADE para a sintaxe compatível FIPA;
- Serviços de *white pages* e *yellow pages* responsáveis pela gestão e localização de agentes, respectivamente;
- Gestor do ciclo de vida dos agentes simples e eficaz que, quando um agente é criado, lhe atribui automaticamente um identificador global único e um endereço de transporte que são utilizados para registar o agente nos serviços de *white pages*. Fornece, também, ferramentas gráficas e API para gerir quer local, quer remotamente os ciclos de vida dos agentes;
- Suporte à mobilidade do agente, *i.e.*, o código do agente e o seu estado podem migrar entre processos e máquinas;
- Suporte para a adopção de ontologias e linguagens de conteúdo, o que possibilita implementar novas linguagens de conteúdos que satisfaçam requisitos de aplicações específicas.

De referir ainda que esta plataforma possui um *site* oficial onde se encontram disponíveis o *software*, a documentação, exemplos de código e outras informações sobre a utilização do JADE.

2.5.2 Jini

O Jini é uma arquitectura orientada a serviços que define um modelo de desenvolvimento de sistemas distribuídos suportado pela tecnologia Java. Esta tecnologia foi criada pela Sun Microsystems em 1999 e é do tipo *open source* [14].

Tal como o JADE, o Jini é um *middleware*. Adopta o modo *"plug and play"* em relação a uma rede onde novos serviços, utilizadores e/ ou equipamentos se podem ligar e ficar imediatamente prontos a ser utilizados sem ser necessário recorrer a instalações ou configurações especificas tanto de *software* como de *hardware*.

O Jini foi também desenvolvido para ajudar os programadores a minimizar os problemas habituais da computação distribuída. Normalmente, quando se desenvolve pela primeira vez uma aplicação distribuída parte-se de um conjunto de pressupostos, que a longo prazo, se provam incorrectos e que causam bastantes problemas. Em 1994, Peter Deutsch [7] designou estas suposições como as oito falácias das aplicações distribuídas:

- A rede é "confiável";
- A latência é zero;
- A largura de banda é infinita;
- A rede é segura;
- A topologia duma rede nunca é alterada;
- Existe apenas um administrador;
- O custo de transporte é zero;
- A rede é homogénea.

O Jini oferece uma série de recursos poderosos como a descoberta de serviços ou o suporte à mobilidade de código para ultrapassar estes problemas incontornáveis.

Arquitectura

A arquitectura do Jini oferece uma infra-estrutura flexível e adaptável às mudanças que permite a criação de sistemas distribuídos dinâmicos, escaláveis e

evolutivos. Uma aplicação, rede, serviço, utilizador ou periférico pode anunciar a sua presença na rede a qualquer outro cliente ligado à rede. A partir desse momento, qualquer cliente (que pode ser outra aplicação, rede, serviço, utilizador ou periférico) pode requerer a sua utilização sem que seja necessário instalar ou configurar o que quer que seja.

Como foi desenvolvida em linguagem Java, a arquitectura Jini tira partido de todas as funcionalidades e vantagens inerentes à linguagem, como o Java Remote Method Invocation (JRMI) e a programação orientada a objectos. Herda, também, as suas desvantagens como, *e.g.*, o baixo desempenho em termos de velocidade de execução quando em comparação com outras arquitecturas. Duma forma geral, a arquitectura Jini implementa uma máquina virtual da linguagem Java em toda a rede.

Outro aspecto da tecnologia Jini é que, apesar de ter sido criada em linguagem Java, nem os serviços nem os clientes necessitam de ser desenvolvidos nesta linguagem porque implementa a especificação Common Object Request Broker Architecture (CORBA).

2.5.3 JASON

O JASON é uma plataforma *open source* baseada em Java para o desenvolvimento de sistemas multi-agente. JASON significa Java-based AgentSpeak interpreter used with Simple Agent Communication Infrastructure (SACI) for multi-agent distribution *Over the Net* e foi desenvolvida por Jomi F. Hübner e Rafael H. Bordini. O Jason é um interpretador de uma versão alargada da linguagem AgentSpeak que é uma linguagem abstracta baseada na arquitectura BDI (*Beliefs, Desires, Intentions*) [15]. Algumas das extensões à linguagem AgentSpeak que foram implementadas nesta plataforma são:

Negação forte: permite adoptar, de acordo com as características do cenário a modelar, a suposição de um mundo fechado ou de um mundo aberto. Assim, em JASON, um agente pode modelar o seu conhecimento em termos do que acredita ser verdadeiro, do que acredita ser falso e do que desconhece;

Tratamento de falhas de planos: como os sistemas multi-agente actuam em ambientes dinâmicos e imprevisíveis, os planos usados para atingir objectivos do agente frequentemente falham. Por isto, é fundamental que as linguagens de agentes forneçam mecanismos para programar o comportamento do agente quando estas situações acontecem;

Comunicação baseada em actos de fala [16]: todo o trabalho na área da comunicação de agentes é baseado na teoria dos actos de fala. Em sistemas multi-agente a interacção entre os agentes é fundamental, sendo importante

que a linguagem forneça construções para escrita de mensagens em que a força ilocucionária seja explícita, e principalmente que os agentes possam interpretar tais mensagens no contexto de sua arquitectura BDI;

Anotações: tanto as crenças como os planos podem ter anotações que são utilizadas, *e.g.*, para registar qual a fonte de informação que adoptou uma determinada crença ou informações que podem ser utilizadas na escolha de planos a serem executados.

Para além das extensões à linguagem AgenteSpeak, o JASON apresenta as seguintes características:

Distribuição: a plataforma permite a definição dos agentes que participam no sistema e em que máquinas cada um dos agentes será executado. Actualmente, estão disponíveis dois tipos de infra-estrutura: a execução de todos os agentes na mesma máquina e a sua distribuição por múltiplas máquinas usando o SACI;

Ambientes: fornece suporte para a criação e desenvolvimento de ambientes que são programados em Java;

Personalização: os programadores podem redefinir as funções de selecção, funções de confiança e a arquitectura geral do agente (incluindo percepções, acções, revisão de crenças e comunicação entre agentes);

Extensibilidade: a extensão do AgentSpeak disponibilizada com o JASON permite a construção de novas acções internas que podem ser programadas em Java e, posteriormente, ser utilizadas no código AgentSpeak;

Integrated Development Environment (IDE): a plataforma JASON é distribuída com um IDE que oferece uma interface gráfica para realizar a gestão de projectos, a alteração e depuração do código fonte dos agentes e do funcionamento do sistema assim como a monitoração do estado de execução dos vários agentes.

Na página oficial da plataforma JASON é possível encontrar informação e documentação sobre a plataforma, um manual desactualizado da plataforma, tutoriais de instalação e de configuração do JASON e da possível integração de agentes JASON e JADE e um fórum de discussão.

2.5.4 JACK

O JACK Intelligent AgentsTM é uma plataforma em Java para o desenvolvimento de sistemas multi-agente criada pela Agent Oriented Software Pty. Ltd. (AOS),

sediada em Melbourne, Austrália. O objectivo é fornecer uma plataforma de alto desempenho para o desenvolvimento de sistemas multi-agente comerciais, industriais e de investigação. Inclui uma implementação da arquitectura BDI e apresenta a facilidade de expansão para suportar diferentes modelos de agentes e requisitos específicos das aplicações [17] e [18].

A linguagem utilizada, a JACK Agent Language, foi desenvolvida em linguagem Java e suporta um novo paradigma de programação, *i.e.*, é uma linguagem de programação orientada a agentes. Algumas das extensões realizadas à linguagem Java são as seguintes:

- Definição de novas classes, interfaces e métodos;
- Extensões à sintaxe Java para suportar novas classes, definições e comandos orientados a agentes, já que possui um compilador que converte as adições sintácticas em classes e comandos escritos em Java;
- Extensões semânticas para dar suporte ao modelo de execução requerido por um sistema orientado a agentes. O conjunto de classes responsável por suportar a execução de programas feito na linguagem JACK é designado *kernel*. Essas classes fazem a gestão automática da concorrência das tarefas executadas em paralelo. Fornecem ainda um comportamento por omissão do agente na reacção a eventos, falhas de acções e tarefas assim como uma infra-estrutura de comunicação para aplicações multi-agente.

No caso do desenvolvimento de agentes BDI, a linguagem JACK possui os seguintes blocos funcionais que permitem:

Agent — definir o comportamento do agente e incluir as capacidades do agente, a que tipo de mensagens e eventos ele responde e que planos utilizará para atingir os seus objectivos;

Capability — representar as funcionalidades do agente;

BeliefSet — permite representar as crenças do agente de forma a ser possível efectuar consultas fazendo uso de um modelo relacional;

View — efectuar consultas de propósito geral sobre o modelo de dados utilizando vários `BeliefSets` ou estruturas de dados da linguagem Java;

Event — descrever uma ocorrência para a qual o agente deve tomar uma determinada acção de reposta. Os eventos podem ser classificados em três tipos:

(i) ***Internal Stimuli*:** são eventos que o agente envia para si mesmo, normalmente como resultado da execução de um método existente num plano que o agente possui;

(ii) ***External Stimuli*:** são mensagens vindas de outros agentes ou percepções relativas ao ambiente em que se encontra;

(iii) ***Motivations*:** são as motivações do agente como, *e.g.*, os objectivos que o agente quer atingir.

Plan — estabelecer planos (instruções) que o agente deve seguir para tentar atingir os seus objectivos.

O JACK é constituído pelos seguintes componentes e ferramentas:

JACK Runtime Environment: suporta a execução de agentes JACK, a gestão do *multi-threading* e da concorrência e inclui uma camada de comunicação com um serviço de nomes de agentes;

JACK Compiler: permite compilar a linguagem de programação JACK Agent em Java e depois, invoca o compilador Java, para gerar as classes;

Agente BDI Model: adiciona suporte para agentes BDI;

SimpleTeam Model: adiciona suporte para o *SimpleTeam*;

Agent Development Environment: disponibiliza um ambiente gráfico de desenvolvimento para visualização e manipulação das aplicações criadas com esta plataforma;

Agent Debugging Environment: permite a visualização das mensagens trocadas entre agentes e mostrar o seu estado interno de execução;

JACOB: o JACK *Object Modeller* é um conjunto de ferramentas de modelação de objectos para o apoio ao transporte de objectos e interacção com as aplicações pré-existentes em Java e C++.

2.6 Serviços Web

Um serviço *Web* é, segundo a definição do World Wide Web Consortium (W3C), "um sistema de *software* projectado para suportar interacção aplicação-a-aplicação sobre uma rede. Possui uma interface descrita num formato processável por máquinas, mais especificamente na linguagem Web Services Description Language (WSDL). Os outros sistemas interagem com o *Web Service* recorrendo a mensagens SOAP, tipicamente transportadas sobre HyperText Transfer Protocol

(HTTP), sob a forma de uma serialização eXtensible Markup Language (XML), em conjugação com outras normas relacionadas com a *Web*" [19].

Para [20] "os *Web Services* são aplicações modulares auto-descritivas, que podem ser disponibilizadas, localizadas e invocadas a partir de qualquer ponto da rede (local ou *Web*).

Segundo [21] "os *Web* Services são aplicações *Web* distribuídas que disponibilizam e descrevem serviços definidos de acordo com regras bem definidas através dos protocolos da Internet para aplicações *Web*".

Os serviços *Web* herdam dos sistemas orientados aos objectos as noções fundamentais de encapsulamento e reutilização. São componentes de *software* que encapsulam uma funcionalidade discreta, *i.e.*, representam uma "caixa-preta" que pode ser reutilizada sem se ter de saber como funciona internamente. O facto de encapsularem funcionalidades discretas determina que sejam auto-contidos e executem apenas a actividade para a qual estão perfeitamente definidas as entradas e saídas como, *e.g.*, invocar uma operação e obter o resultado.

Destinam-se a ser consumidos por outras aplicações e não directamente por humanos e são baseados em protocolos padrão e abertos, *i.e.*, são independentes dos sistemas operativos, plataformas e linguagens de desenvolvimento (SOAP), são auto-descritivos (WSDL) e passíveis de serem descobertos em repositórios de serviços Universal Description Discovery and Integration (UDDI).

Uma das características mais relevantes deste novo paradigma é a noção de que tudo que é um serviço tem uma interface publicada e que comunica através de mensagens. Desta forma, estes serviços podem ser implementados em qualquer plataforma e linguagem de programação, desde que possam produzir e consumir as mensagens definidas através da interface. Podem ainda ser alterados sem qualquer impacto nas aplicações clientes, desde que mantenham a mesma interface.

Para permitir que as aplicações cliente possam descobrir de forma dinâmica a interface dos serviços *Web*, estes devem ser descritos utilizando a especificação WSDL. Esta norma permite publicar as informações sobre a sintaxe, métodos, parâmetros e localização dos serviços [22]. Desta forma, para construir uma aplicação que interaja com um determinado WS, descarrega-se primeiro o ficheiro WSDL associado e, com base nas informações nele contidas, desenvolve-se a aplicação.

Os ficheiros WSDL podem ser armazenados num directório geral de registo e pesquisa de serviços designado UDDI. Os clientes interessados em encontrar um determinado serviço podem interagir com este directório e obter informações relativas à interface e à localização do serviço.

As principais vantagens dos serviços *Web* face aos sistemas distribuídos tradicionais, *e.g.*, CORBA, são as seguintes:

- Permitem a interacção entre aplicações sem a necessidade de intervenção humana;
- Baseiam-se em protocolos padrão abertos, tais como o HTTP, XML, Simple Object Access Protocol (SOAP),WSDL e UDDI;
- São acessíveis como componentes a partir de qualquer local da Internet;
- Se for adoptado o protocolo HTTP, conseguem funcionar mesmo com *firewalls* e servidores *proxy* porque o tráfego é transmitido através da porta TCP 80;
- Conseguem tirar partido da autenticação e encriptação Secure Socket Layer (SLL);
- Combinam a programação orientada a objectos com a programação *Web*;
- São independentes do sistema operativo (Windows, Linux ou MAC) da plataforma de desenvolvimento (.Net, WASP, NuSOAP, SOAP-Lite, etc.) e da linguagem de programação (C, C#, Visual Basic, PHP, Java, Perl, *etc.*);
- Apresentam os resultados na forma de documentos XML;
- Diminuem a complexidade e custos de integração de sistemas proprietários.

2.6.1 XML

A XML é uma meta-linguagem de anotação utilizada para definir mensagens de representação de informação, que se assume como norma no intercâmbio dessa informação em ambientes heterogéneos.

O W3C definiu a XML baseando-se na linguagem Standard Generalized Markup Language (SGML) da norma ISO 8879. Foi desenhada com o intuito de permitir a definição de documentos estruturados, auto-descritivos e legíveis — como texto — sendo suficientemente formal para permitir a sua validação e processamento de forma automática, tendo estrutura suficiente para suportar documentos complexos e grandes quantidades de informação. Um documento XML usa uma sintaxe simples e de fácil compreensão. É constituído por uma árvore de elementos relacionados designados anotações que utiliza para caracterizar os dados e a sua formatação [23].

2.6.2 SOAP

O SOAP é um protocolo baseado em XML para a troca de informação num ambiente descentralizado e distribuído. Uma mensagem SOAP consiste em três partes: (*i*) um envelope que define a estrutura para descrever o conteúdo da mensagem e como processá-la; (*ii*) um conjunto de regras de codificação (serialização) para expressar instâncias dos tipos de dados utilizados na aplicação; e (*iii*) um mecanismo (convenção) que define as chamadas e respostas através de procedimentos remotos Remote Procedure Call (RPC) [24].

O SOAP foi desenvolvido com o objectivo de ser um protocolo de comunicação simples e leve. O cliente, quando invoca um método remoto usando o SOAP, não necessita de se preocupar com alocação de memória, sistema operativo, *etc.*, executando o método no servidor como se fosse um processo local. O SOAP é o padrão utilizado para realizar RPC em *Web Services*. Foi desenvolvido a partir do XML-RPC, que é uma especificação desenvolvida pela Microsoft em 1998 mais simples mas também mais limitada que o SOAP. A especificação SOAP é actualmente mantida pelo consórcio W3C. Fazendo a analogia com uma carta de correio, a mensagem SOAP encontra-se envolta num envelope e é constituído por um cabeçalho (identificação) e por um corpo (conteúdo). Assim, os elementos constituintes da mensagem SOAP são:

Envelope: É o elemento raiz de uma mensagem SOAP, identificando o documento XML como uma mensagem SOAP. A anotação <`soap:Envelope`> define o envelope SOAP e contém obrigatoriamente o `namespace` `xmlns:soap` com o valor `http://www.w3.org/2001/12/soap-envelope`. Pode também incluir atributos adicionais como o `encodingStyle`, que define os tipos de dados no documento XML.

Cabeçalho: É um elemento opcional definido através da anotação <`soap:Header`> que contém informações específicas da mensagem SOAP, tais como autenticação, transacções e encriptação. Dentro deste elemento existem dois atributos importantes: (*i*) o actor que identifica o `endpoint` a que se destina o `header` e (*ii*) o `mustUnderstand` que permite garantir que o destinatário reconhece o cabeçalho.

Corpo: É um elemento obrigatório da mensagem SOAP que transporta a informação da mensagem. É definido através da anotação <`soap:Body`> e contém os métodos, parâmetros ou respostas do serviço *Web*. No caso de ocorrerem erros no processamento de mensagens SOAP, recorre-se ao subelemento definido pela etiqueta <`soap:Fault`>.

Na Figura 2.8 apresenta-se uma mensagem SOAP.

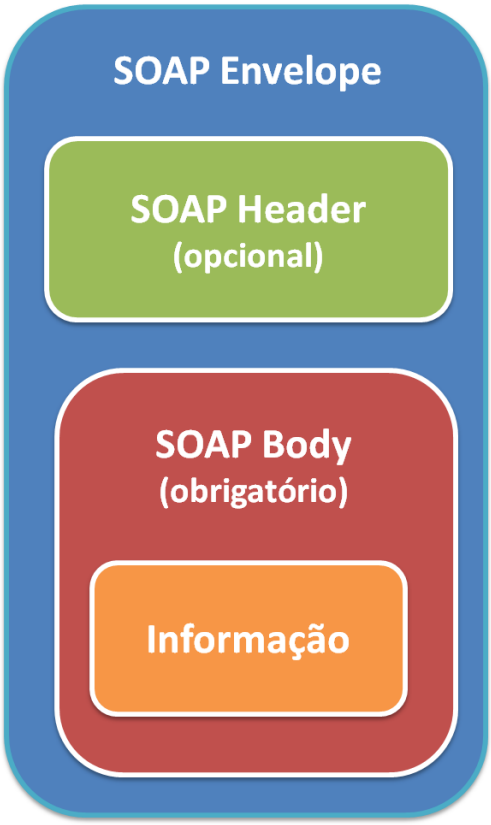

Figura 2.8: Estrutura de uma mensagem SOAP

De forma a não comprometer a independência relativamente às plataformas de desenvolvimento, o protocolo SOAP não define qual o protocolo de transporte a utilizar. Apesar do protocolo mais utilizado ser, sem dúvida, o HTTP, é possível utilizar outros protocolos como o File Transfer Protocol (FTP) ou o Simple Mail Transfer Protocol (SMTP).

2.6.3 WSDL

Segundo [20], a "WSDL é uma linguagem padrão, representada num documento XML, que permite aos serviços *Web* serem auto-descritivos".

O ficheiro WSDL de um serviço *Web* descreve os métodos (operações) que disponibiliza e como se podem utilizar. Uma aplicação cliente que pretenda utilizar uma determinada operação, só necessita de conhecer o respectivo ficheiro WSDL, normalmente disponibilizado através de um Uniform Resource Locator (URL).

Cada documento WSDL obedece a uma estrutura bem definida, composta pelos seguintes elementos [25]:

`definitions:` é o elemento raiz do documento e contém os atributos `name`, para atribuir um nome aos documentos assim como os vários `namespaces` utilizados em cada documento;

`documentation:` é utilizado para a inserção de comentários em texto ou XML destinados aos humanos;

`types:` este elemento define os tipos de dados transportados nas mensagens (`message`) de pedido (`input`) ou de resposta (`output`) e possui um atributo `name` que especifica o nome da mensagem e o conjunto de parâmetros de entrada e de resultado das mensagens;

`operation:` este elemento define a assinatura do método, *i.e.*, permite associar pares de mensagens pedido/ resposta, identificando os parâmetros de entrada e saída;

`portType:` é o elemento que define as operações (métodos) disponibilizadas pelo serviço *Web*. A cada operação, definida através do elemento `operation`, estão associados um pedido e uma resposta identificados respectivamente pelos elementos `input` e `output`;

`binding:` este elemento especifica qual o protocolo concreto de uma operação. A especificação WSDL permite ligações com protocolos e formatos de mensagem como SOAP, HTTP GET /POST e Multipurpose Internet Mail Extensions (MIME);

`service:` define o nome e o endereço do serviço disponibilizado através de um conjunto de elementos `port`.

2.7 Conclusão

Neste capítulo foram apresentados os conceitos relacionados com agentes, sistemas multi-agente e serviços *Web*. Descreveram-se ainda as principais plataformas de código aberto disponíveis para a realização de sistemas multi-agente.

Este estudo e levantamento permitiu, no Capítulo 3, efectuar a comparação entre plataformas de desenvolvimento de sistemas multi-agente e escolher a plataforma que foi adoptada neste trabalho de forma justificada.

Capítulo 3

Arquitectura

Neste capítulo descrevem-se e justificam-se as decisões tomadas nas fases de desenho e desenvolvimento do projecto em relação à arquitectura para assegurar que o sistema cumpre os objectivos determinados.

3.1 Arquitectura Funcional

Para definir correctamente a arquitectura efectuou-se uma análise funcional dos requisitos. Para isso foram definidos cenários de utilização bem como casos de uso para analisar se a solução corresponde às metas traçadas.

3.1.1 Casos de Uso

Os casos de uso definidos correspondem às funcionalidades principais que o sistema multi-agente deve suportar. Estes também serão utilizados para definir os testes de sistema.

Existem fundamentalmente quatro serviços distintos que o sistema deve disponibilizar: (*i*) o Serviço de Posicionamento; (*ii*) o Serviço de Navegação; (*iii*) o Serviço de Seguimento; (*iv*) e o Serviço de Meteorologia.

Associado ao Serviço de Posicionamento está a capacidade de disponibilizar informação sobre a localização actual de um dispositivo, permitindo ainda a actualização da informação com a alteração da sua localização.

Ao Serviço de Navegação cabe oferecer informação sobre o local de destino, o local actual e um trajecto que permita interligar estes dois locais.

Quanto ao Serviço de Seguimento, deve apresentar informação sobre a localização actual e final de um outro dispositivo que não aquele que realiza o pedido.

Finalmente, o Serviço de Meteorologia, deve disponibilizar informação sobre o estado meteorológico numa determinada localização.

Para além destes serviços o sistema deve ainda: (*i*) levar em conta as preferências do utilizador quer em relação aos serviços, quer em relação a outros aspectos; (*ii*) proporcionar uma sensação de continuidade de serviço, *i.e.*, o sistema deve guardar informação sobre o último serviço oferecido a cada utilizador. Para garantir esta personalização, é necessário identificar cada utilizador através de uma autenticação junto do módulo servidor.

3.1.2 Diagramas de Sequência

Após a identificação dos casos de uso, das entidades e sistemas externos relacionados com cada transacção bem como da sua sequência, foram desenhados os diagramas de sequência.

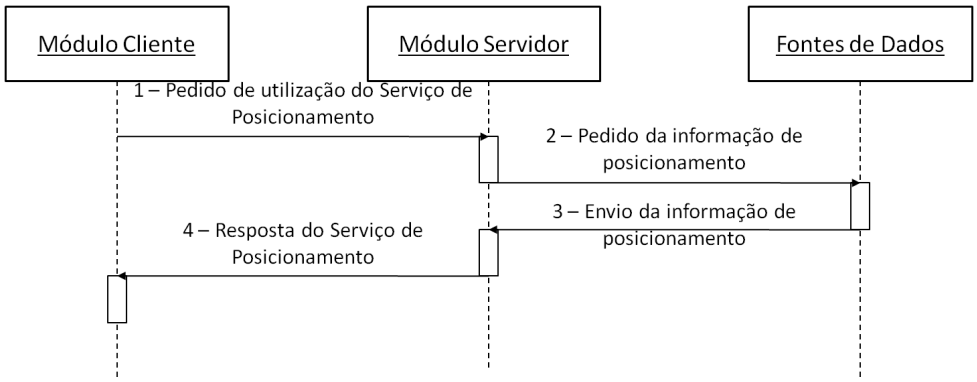

Figura 3.1: Diagrama de sequência da utilização do Serviço de Posicionamento

Tabela 3.1: Interacções durante a utilização do Serviço de Posicionamento

Interacção	Descrição	Entidades
1	É enviada uma mensagem a requisitar o serviço de Posicionamento	Módulo Cliente Módulo Servidor
2	O Módulo Servidor pede a informação de posicionamento	Módulo Servidor Fontes de Dados
3	A Fonte de Dados responsável pela informação de posicionamento envia-a ao Módulo Servidor	Fontes de Dados Módulo Servidor
4	O Módulo Servidor responde ao pedido efectuado com a informação de posicionamento	Módulo Servidor Módulo Cliente

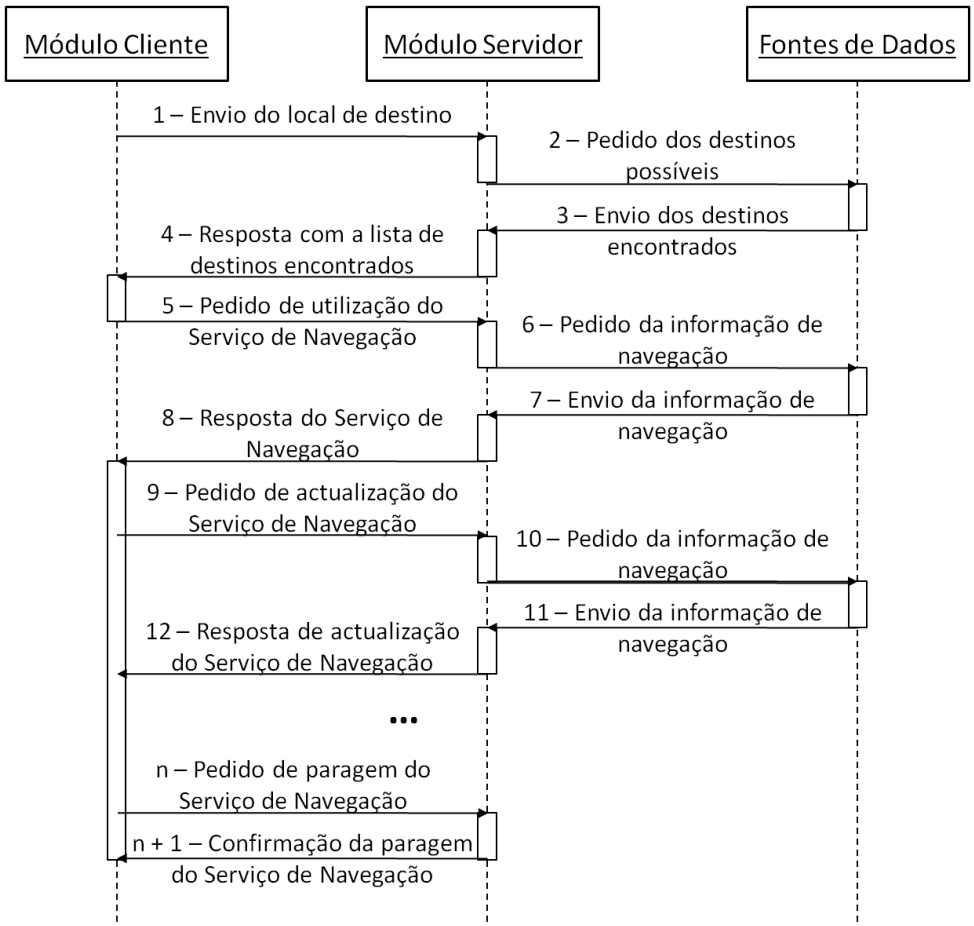

Figura 3.2: Diagrama de sequência da utilização do Serviço de Navegação

Tabela 3.2: Interacções durante a utilização do Serviço de Navegação

Interacção	Descrição	Entidades
1	O Módulo Cliente envia uma mensagem com o local de destino desejado	Módulo Cliente Módulo Servidor
2	O Módulo Servidor requisita a lista dos destinos possíveis mediante o local indicado pelo Módulo Cliente	Módulo Servidor Fontes de Dados
3	Envio dos destinos encontrados pela Fonte de Dados	Fontes de Dados Módulo Servidor
4	O Módulo Servidor retorna a lista desses locais	Módulo Servidor Módulo Cliente
5	É enviada uma mensagem a requisitar o serviço de navegação	Módulo Cliente Módulo Servidor
6	O Módulo Servidor pede a informação de navegação	Módulo Servidor Fontes de Dados
7	A Fonte de Dados responsável pela informação de navegação envia-a ao Módulo Servidor	Fontes de Dados Módulo Servidor
8	O Módulo Servidor responde ao pedido efectuado com a informação de navegação	Módulo Cliente Módulo Servidor
9	É enviada uma mensagem a requisitar a actualização do Serviço de Navegação	Módulo Cliente Módulo Servidor
10	Pedido de actualização da informação de navegação à Fonte de Dados	Módulo Servidor Fontes de Dados
11	A Fonte de Dados responsável envia a informação de actualização ao Módulo Servidor	Fontes de Dados Módulo Servidor
12	Resposta ao pedido efectuado com a actualização da informação de navegação	Módulo Servidor Módulo Cliente
...		
n	O Módulo Cliente envia uma mensagem para parar o Serviço de Navegação	Módulo Cliente Módulo Servidor
$n+1$	O Módulo Servidor confirma a recepção da mensagem de paragem do Serviço de Navegação	Módulo Servidor Módulo Cliente

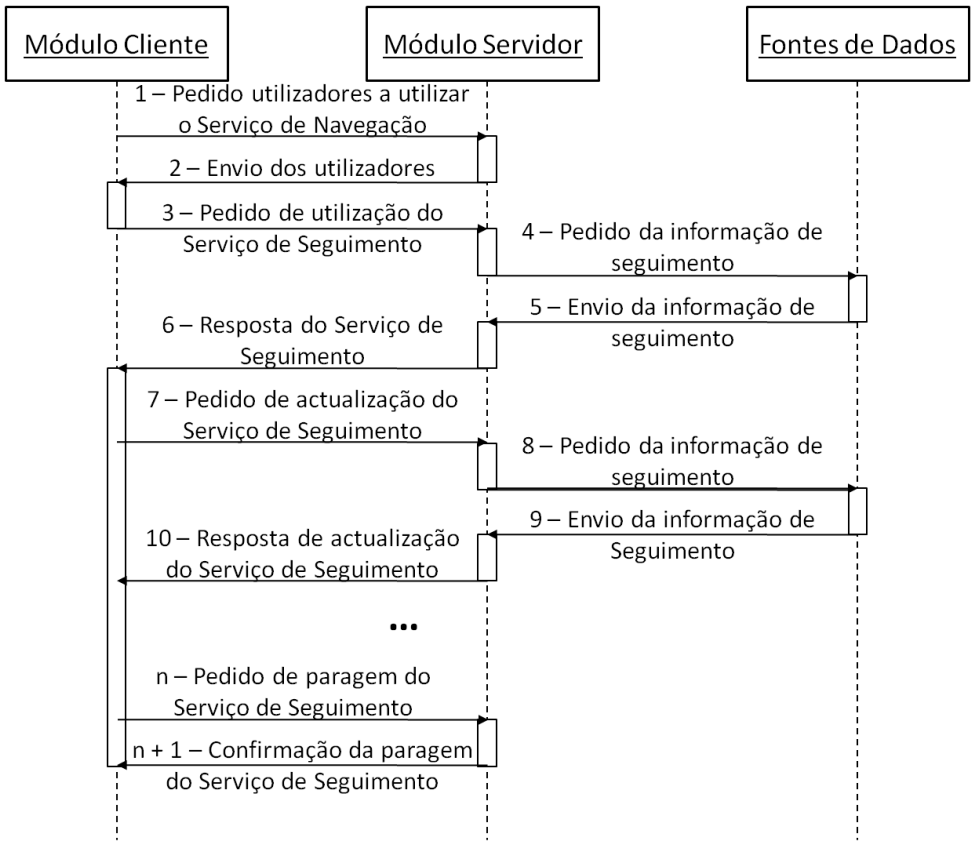

Figura 3.3: Diagrama de sequência da utilização do Serviço de Seguimento

Tabela 3.3: Interacções durante a utilização do Serviço de Seguimento

Interacção	Descrição	Entidades
1	O Módulo Cliente envia uma mensagem a pedir a lista de utilizadores autenticados e que estão a utilizar o serviço de navegação naquele momento	Módulo Cliente Módulo Servidor
2	O Módulo Servidor retorna a lista dos utilizadores nessas condições	Módulo Servidor Módulo Cliente
3	É enviada uma mensagem a requisitar o serviço de seguimento	Módulo Cliente Módulo Servidor
4	O Módulo Servidor pede a informação de seguimento	Módulo Servidor Fontes de Dados
5	A Fonte de Dados responsável pela informação de seguimento envia-a ao Módulo Servidor	Fontes de Dados Módulo Servidor
6	O Módulo Servidor responde ao pedido efectuado com a informação de seguimento	Módulo Cliente Módulo Servidor
7	É enviada uma mensagem a requisitar a actualização do Serviço de Seguimento	Módulo Cliente Módulo Servidor
8	Pedido de actualização da informação de seguimento à Fonte de Dados	Módulo Servidor Fontes de Dados
9	A Fonte de Dados responsável envia a informação de actualização ao Módulo Servidor	Fontes de Dados Módulo Servidor
10	Resposta ao pedido efectuado com a actualização da informação de seguimento	Módulo Servidor Módulo Cliente
	...	
n	O Módulo Cliente envia uma mensagem para parar o serviço de seguimento	Módulo Cliente Módulo Servidor
$n+1$	O Módulo Servidor confirma a recepção da mensagem de paragem do serviço de seguimento	Módulo Servidor Módulo Cliente

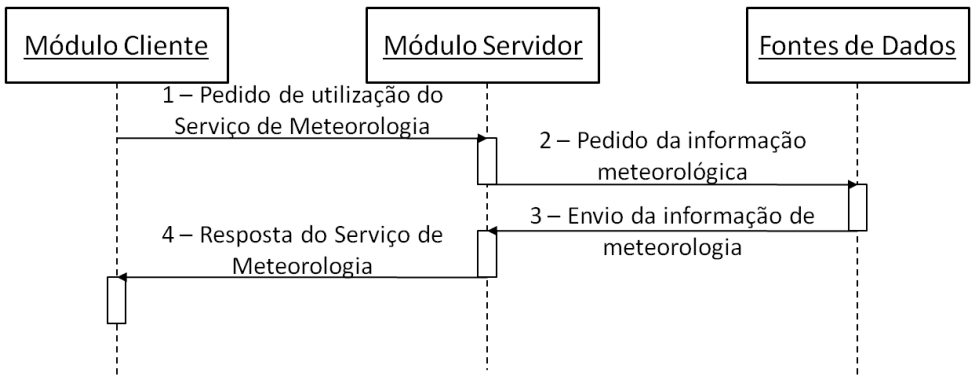

Figura 3.4: Diagrama de sequência da utilização do Serviço de Meteorologia

Tabela 3.4: Interacções durante a utilização do Serviço de Meteorologia

Interacção	Descrição	Entidades
1	É enviada uma mensagem a requisitar o serviço de meteorologia	Módulo Cliente Módulo Servidor
2	O Módulo Servidor pede a informação meteorológica	Módulo Servidor Fontes de Dados
3	A fonte de Dados responsável pela informação de meteorologia envia-a ao Módulo Servidor	Fontes de Dados Módulo Servidor
4	O Módulo Servidor responde ao pedido efectuado com a informação de meteorologia	Módulo Servidor Módulo Cliente

3.2 Arquitectura de Implementação

A arquitectura foi elaborada tendo em conta os objectivos expressos no Capítulo 1. Pretende-se então desenvolver um sistema multi-agente onde cada agente possui uma interface do tipo serviço *Web* e é responsável pela prestação dos meta-serviços de intermediação, continuidade de serviço e personalização a um utilizador. Com o intuito de atingir estes requisitos foi necessário proceder à selecção da plataforma Multi-agente a utilizar assim como da sua integração com a interface serviço *Web*.

3.2.1 Escolha da Plataforma Multi-agente

Para escolher a plataforma de desenvolvimento de sistemas multi-agente foi realizado um estudo comparativo com as opções consideradas, tendo em conta os seguintes critérios:

1. Facilidade da criação de agentes que é quantificada em termos da existência ou não de uma API que ofereça um modelo interno para os agentes e protocolos de comunicação inter-agentes;

2. Disponibilização de serviços característicos de sistemas multi-agente como, *e.g.*, serviços de páginas amarelas e de nomes;

3. Compatibilidade e portabilidade do sistema, *i.e.*, se o sistema é compatível com algum padrão (*e.g.*, FIPA), se o sistema é facilmente portável, se o sistema necessita de outros programas para serem executados (*e.g.*, JVM ou DLL);

4. Existência de informação e documentação sobre o funcionamento da ferramenta, *i.e.*, a existência de manuais de utilização, fóruns de discussão, *etc.*

Este estudo é apresentado na Tabela 3.5:

Tabela 3.5: Resumo das plataformas de sistemas multi-agente

Critérios SMA	Facilidade criação agentes	Serviços disponibilizados	Compatibilidade e portabilidade	Informação e documentação
JADE	– Utiliza classes intuitivas que devem ser estendidas ou instanciadas para a criação do agente, o acesso e envio de mensagens, a criação de comportamentos entre outros	– Interface gráfica que permite a monitorização, depuração e *logging* – Facilmente distribuído pela rede – Mobilidade de agentes – Gestão do ciclo de vida dos agentes – Utiliza troca de mensagens baseadas em *speech act* – Serviço de nomes – Serviço de páginas amarelas – Serviço de ontologia	– Compatível com plataformas que implementem a especificação FIPA – Pode ser executado em qualquer ambiente com JVM	– Possui um serie de documentos no site oficial da plataforma, tais como, manuais e tutoriais
Jini		– Serviço de *look-up*	– Pode ser executado em qualquer ambiente com JVM	– Possui alguma informação no site oficial
JASON	– Implementa a arquitectura BDI	– IDE com interface gráfica que permite depuração – Facilmente distribuído pela rede – Utiliza troca de mensagens baseadas em speech act – Suporta o desenvolvimento de ambientes – Não possui serviço de nomes nem de páginas amarelas	– Pode ser executado em qualquer ambiente com JVM	– Possui alguma mas pouca informação e documentação no site oficial, apenas existe um manual e um tutorial
Jack	– Possibilita a Implementação de agentes do tipo BDI	– IDE com interface gráfica que permite depuração e *logging* – Serviço de nomes – Não possui serviço de páginas amarelas	– Pode ser executado em qualquer ambiente com JVM	– Possui um serie de manuais no site oficial da plataforma

Após a realização deste estudo e da leitura de alguns artigos da especialidade nomeadamente [26] [27] [28], a escolha recaiu sobre o framework JADE visto ser uma das ferramentas mais utilizadas na criação de sistemas multi-agente, disponibilizar os serviços característicos, utilizar a especificação FIFA, oferecer

um boa documentação e material de apoio e de permitir a integração entre agentes e serviços *Web*.

3.2.2 Interface Serviço Web

Um dos requisitos deste projecto é que os serviços criados possuam uma interface tipo serviço *Web*. Procedeu-se então à pesquisa, identificação e analise da opções existentes para interligar a plataforma JADE com o mundo dos serviços *Web*, tendo resultado três hipóteses:

Web Service Agent Integration (WSAI) — O projecto WSAI foi desenvolvido com o objectivo de dotar os agentes JADE de interaces do tipo serviço *Web*, *i.e.*, permitir que um agente seja visto como um serviço *Web*.

A implementação deste projecto é baseada em dois componentes fundamentais: (*i*) o `Agent Gateway`, denominado WSAG (Web Service Agent Gateway), e (*ii*) o `Agent Generator`. O `Agent Gateway` é responsável por transformar os agentes em serviços *Web*. O `Agent Generator` é uma ferramenta de suporte que permite criar os `Agent Gateway` — entidades que fornecem um determinado serviço *Web* — de um agente específico. Tanto o `Agent Gateway` quanto o `Agent Generator` foram implementados utilizando o Framework JADE.

Entretanto, com a criação por parte do JADE de um *add-on* que permite a integração de serviços *Web* e de agentes designado WSIG, este projecto foi desactivado.

WSDL2AGENT — Esta ferramenta, após analisar o ficheiro WSDL de um qualquer serviço *Web*, cria um agente *proxy* capaz de aceitar pedidos de agentes, invocar o serviço *Web* e enviar os resultados obtidos para o agente.

O `Wsdl2Agent` gera automaticamente tanto a ontologia do agente *proxy*, que está directamente relacionada com o próprio serviço *Web*, quanto o código do agente *proxy* a alojar na plataforma JADE. A ontologia produzida pode ser distribuída entre os agentes clientes para que possam realizar pedidos e receber as respostas do agente *proxy*. Quanto ao agente *proxy*, realiza todas as traduções necessárias entre as mensagens na linguagem de comunicação de agentes, *i.e.*, os pedidos e respostas do agente, e as mensagens SOAP, *i.e.*, os pedidos e respostas do serviço *Web*, antes e depois da realização de um pedido ao serviço *Web*.

Com esta ferramenta, os agentes que desejem aceder a serviços *Web* não necessitam de implementar quaisquer protocolos de chamada de serviços *Web*, mas apenas enviar uma mensagem na linguagem de comunicação de agentes para o agente *proxy* e depois processar a resposta recebida.

Web Service Integration Gateway (WSIG) — O objectivo do WSIG é expor os serviços oferecidos pelos agentes, que estão publicados no `Directory Facilitator` (DF) `Agent` do JADE, como serviços *Web* com um esforço adicional mínimo. A transformação de um agente JADE num serviço *Web* envolve a criação do ficheiro WSDL do respectivo serviço registado no `DF Agent`, possibilitando assim também a publicação dos serviços expostos num registo UDDI.

O WSIG suporta a pilha padrão dos serviços *Web*, constituída pela utilização de WSDL para a descrição dos serviços, mensagens SOAP para a realização do transporte e repositórios UDDI para publicar os serviços *Web*.

Este *add-on* é formado por dois elementos principais: (i) o `WSIG Servlet` que funciona de *front-end* para a Internet; e (ii) o `WSIG Agent` que é o *gateway* entre o mundo *Web* e os agentes JADE.

A escolha recaiu sobre o *add-on* do JADE WSIG porque implementa o padrão serviço *Web* na sua totalidade e transforma os agentes em serviços *Web* de uma forma elegante para o programador. Nesta abordagem, o `WSIG Servlet` é responsável por:

- Receber os pedidos HTTP/SOAP;
- Extrair a mensagem SOAP;
- Preparar e passar o pedido de realização da respectiva acção do agente ao `WSIG Agent`.

Após a acção do agente ter sido realizado é ainda responsável por:

- Converter o resultado da acção numa mensagem SOAP;
- Preparar a resposta HTTP/SOAP a enviar ao cliente.

Por seu lado, `WSIG Agent` é responsável por:

- Encaminhar os pedidos de realização de acções recebidas do `WSIG Servlet` para os agentes que são capazes de as realizar e obter as respectivas respostas;
- Receber as notificações de registo e da remoção do registo de agentes no `DF Agent`;
- Criar o ficheiro WSDL correspondente aos serviços de cada agente registado no `DF Agent` e, se necessário, publicar esses serviços num registo UDDI.

Durante o funcionamento do WSIG, há três processos essenciais que permanecem ininterruptamente activos:

- `WSIG Agent` monitora a actividade de registo/cancelamento de registo no `DF Agent` e cria/remove os respectivos ficheiros WSDL;
- `WSIG Servlet` recebe os pedidos efectuados aos serviços *Web*, prepara os pedidos de acções correspondentes aos agentes, submete-os ao `WSIG Agent` e realiza as traduções necessárias;
- `WSIG Agent` encaminha os pedidos de realização de acções para os agentes com capacidade de os atender.

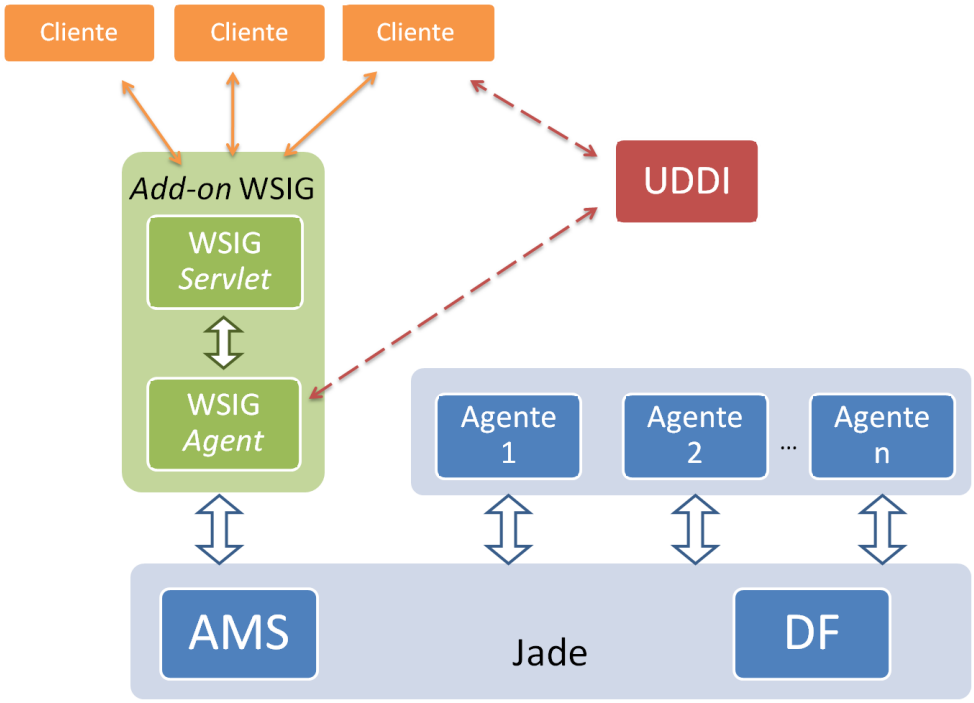

Figura 3.5: Arquitectura do *add-on* WISG

3.2.3 Aplicações Cliente

Outro dos objectivos deste trabalho é a criação de aplicações para diversos tipos de dispositivos. Para cumprir este requisito foi decidido criar duas aplicações:

- Uma aplicação cliente para dispositivos móveis como telemóveis, *smartphones*, PDA ou *handhelds*;

- Uma aplicação cliente para computadores de secretária, portáteis ou, até mesmo, *netbooks*.

A escolha recaiu sobre o desenvolvimento destes dois tipos de aplicações, uma vez que permitem abranger uma grande fatia, senão mesmo a totalidade, dos dispositivos existentes actualmente no mercado com as características necessárias para a utilização dos serviços que serão disponibilizados.

Nesta fase da dissertação e para permitir a execução de aplicações Java nos dispositivos móveis, foi também necessário escolher a Kilo Virtual Machine (KVM) a instalar nos móveis. Para isso foi realizado um estudo, que se apresenta na Tabela 3.6, envolvendo várias KVM e diferentes versões.

Tabela 3.6: Comparação das KVM

KVM	Versão	Programas	
		AppMobile	FileTextApplication
Esmertec Jbed	20080222.3.1	- (the permission javax.microedition.location.location is not available on this device)	- (Security Exception)
	20080704.5.1	+	- (Security Exception)
	20080912.5.1	+	- (Security Exception)
	20081203.2.1	- (the permission javax.microedition.location.location is not available on this device)	- (IOException)
	20090217.5.1	+	- (Security Exception)
	20090416.5.1	+	- (Security Exception)
	20090506.2.1	- (the permission javax.microedition.location.location is not available on this device)	- (IOException)
JBlend	20080515	Falha na instalação dos MIDlets: falha na autorização do aplicativo	
	20090429.2.1	Durante o processo de instalação dos MIDlets a JVM encerra automaticamente	
	20090619.2.1	Erro durante o processo de instalação desta JVM (o programa não pode ser instalado porque não possui permissões de sistema suficientes)	
Intent	11.1.7.1036	Falha na instalação dos MIDlets	
Mysaifu	0.4.8	Falha na instalação dos MIDlets	
IBM J9	20061109_1110	-	+

Os símbolos + e − indicam se a KVM suporta ou não a biblioteca em causa. Os testes consistiram na instalação, num telemóvel, de cada uma dessas KVM

e na execução de duas aplicações desenvolvidas em Java para determinar se suportavam as bibliotecas necessárias à realização da aplicação cliente. A primeira aplicação, designada `AppMobile`, determina se a KVM suporta a biblioteca `Location`, responsável pela obtenção da localização Global Positioning System (GPS) do dispositivo móvel. A segunda aplicação, designada `FileTextApplication`, verifica se a KVM suporta a biblioteca `File Connection` responsável por permitir o acesso aos ficheiros do telemóvel.

A KVM escolhida foi a Esmertec Jbed versão 20090416.5.1 porque não só é a versão mais recente à data da realização do estudo, mas, sobretudo, porque permite aceder às coordenadas GPS do telemóvel. A IBM J9, que foi a única KVM que acedeu com sucesso aos ficheiros do telemóvel, não conseguiu aceder às coordenadas geográficas recebidas pelo GPS incorporado no dispositivo móvel.

3.3 Arquitectura Proposta

A arquitectura geral proposta para o sistema é apresentada na Figura 3.6.

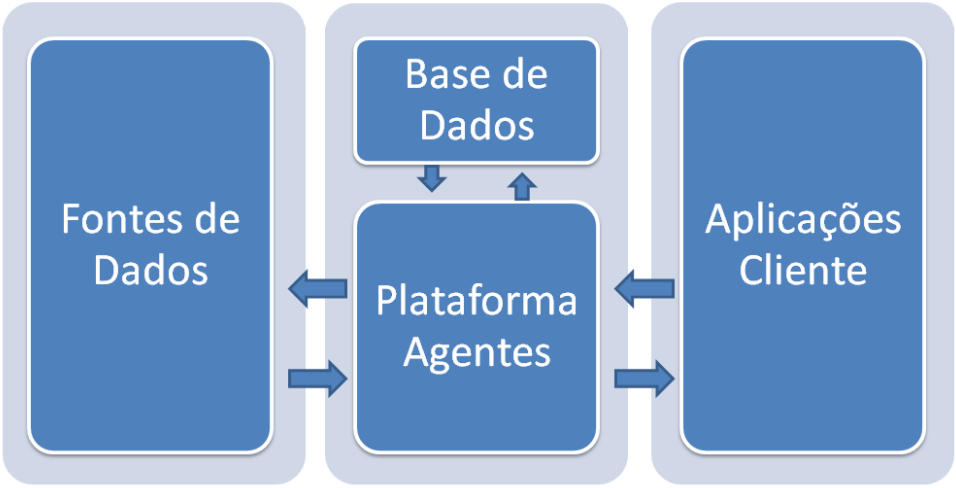

Figura 3.6: Diagrama conceptual da arquitectura proposta

Como se poder ver, o sistema proposto é constituído por três grandes blocos:

Fontes de Dados que fornecem em tempo útil toda a informação necessária para os diversos serviços disponibilizados. Estas fontes são módulos servidores que respondem aos pedidos dos agentes;

Sistema Multi-agente que, por um lado, recebe e responde às solicitações das aplicações cliente e, por outro, realiza pedidos às fontes de dados e processa

a informação obtida. Mantém ainda a informação necessária à prestação dos serviços de intermediação, continuidade de serviço e personalização numa base de dados;

Aplicações Cliente que utilizam os serviços *Web* colocados à disposição pela plataforma de agentes. As mensagens trocadas entre estas duas entidades são do tipo SOAP.

Uma outra representação da arquitectura proposta é apresentada na Figura 3.7. Esta, tal como a anterior, está dividida em três grandes blocos: Fontes de Dados, Módulo Servidor e Módulo Cliente.

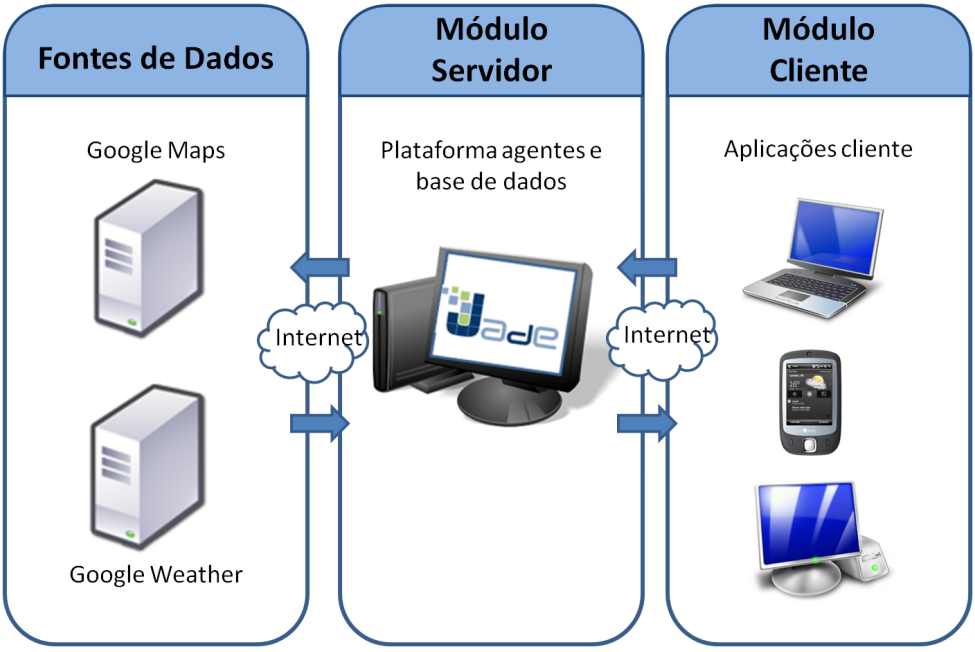

Figura 3.7: Arquitectura proposta

O primeiro bloco, o das fontes de dados, é constituído por vários servidores da Google. Os serviços prestados pelo sistema implementado durante esta dissertação utilizam, por sua vez, vários serviços da Google como fontes da informação apresentada aos utilizadores. Alguns dos recursos utilizados foram os seguintes:

- Serviço de *Geocoding* — utilizado para traduzir endereços ou moradas em coordenadas geográficas;
- Serviço de *Reverse Geocoding* — utilizado para traduzir coordenadas geográficas em endereços ou moradas;

- Serviço de Direcções — utilizado para obter as diversas direcções entre dois locais;

- Serviço de Meteorologia — utilizado para obter as condições meteorológicas de uma localização.

Quanto ao segundo bloco, o módulo servidor, encontra-se alojado um computador onde estão instalados a plataforma de agentes JADE e o servidor de base dados MySQL. O sistema multi-agente presta os serviços de mediação, continuidade de serviço e personalização.

No último bloco, o módulo cliente, caso se trate de um dispositivo móvel, a aplicação será um `MIDlet` J2ME, caso contrário, será um `Applet`. As aplicações cliente fornecem a interface do utilizador.

Todos estes blocos estão interligados, *i.e.*, comunicam e/ou trocam mensagens entre si através da Internet.

3.3.1 Módulo Cliente

O módulo cliente pode ser uma aplicação móvel ou um `Applet`. A aplicação móvel consiste num `MIDlet` J2ME residente na máquina virtual Java do dispositivo móvel. Este tipo de máquinas virtuais é denominado de KVM e são uma implementação Virtual Machine (VM) da Java optimizada para dispositivos limitados. No caso do `Applet`, a execução decorre na JVM instalada no computador onde for invocado. Ambos os tipos de aplicação cliente devem disponibilizar aos utilizadores o acesso transparente aos Serviços de Posicionamento, Navegação, Seguimento e de Meteorologia.

3.3.2 Módulo Servidor

O módulo servidor é constituído pelo sistema multi-agente de intermediação e respectiva base de dados.

Arquitectura da Plataforma Multi-agente

A arquitectura da plataforma multi-agente é constituída pelos agentes necessários ao funcionamento do JADE assim como pelos agentes criados durante o desenvolvimento desta dissertação (ver Figura 3.8).

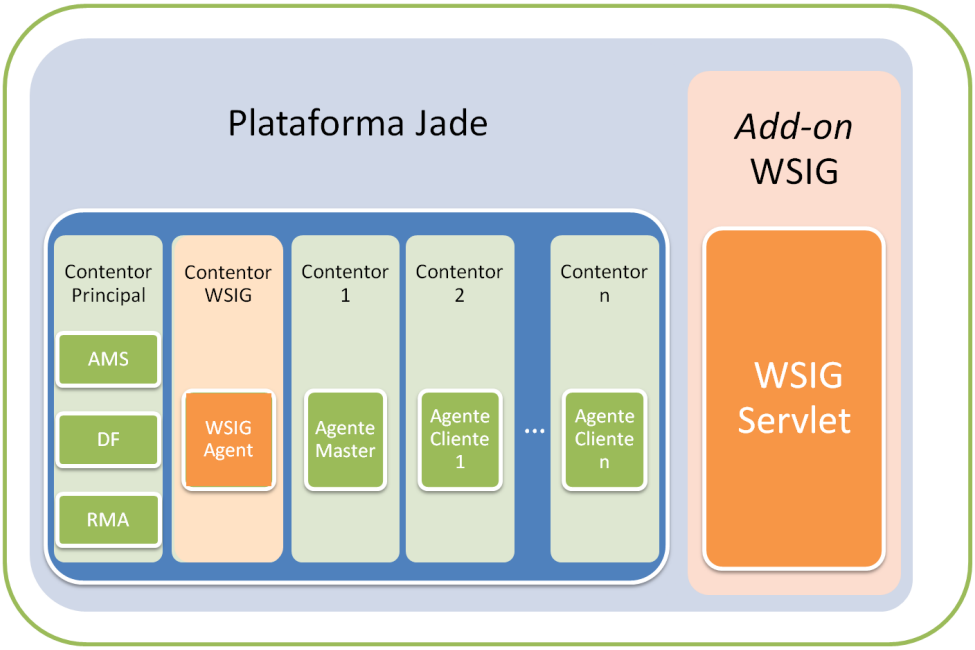

Figura 3.8: Arquitectura da Plataforma JADE

Como foi referido no Capítulo 2, o JADE é constituído por contentores que comportam agentes. Por omissão, a plataforma contém o contentor principal e um contentor por cada utilizador autenticado ligado. O número total de contentores criados pode ser obtido pela equação 3.1:

$$N_{contentores} = 3 + N_{clientes\_autenticados} \tag{3.1}$$

No caso deste trabalho, são criados os seguintes contentores:

- Contentor principal;
- Contentor WSIG;
- Contentor que contém o agente `master`;
- N contentores que albergam os N agentes criados.

O contentor principal é constituído pelos agentes descritos de seguida:

`Agent Management System` **(AMS):** supervisiona toda a plataforma e disponibiliza o serviço de páginas brancas (*white pages*) onde todos os agentes são obrigados a registar-se. Este registo no AMS é automático e resulta na atribuição de um AID (`Agent Identifier`) único a cada agente;

`Directory Facilitator` **(DF)** `Agent:` oferece o serviço de páginas amarelas (*yellow pages*) que permite a qualquer agente registar os serviços que oferece ou consultar os serviços disponíveis;

`Remote Management Agent` **(RMA):** disponibiliza uma interface gráfica para a administração da plataforma e dos agentes que a compõem. Permite, de forma local ou remota, controlar o ciclo de vida dos agentes em execução e utilizar as ferramentas oferecidas pelo JADE.

O contentor WSIG contém o agente WSIG do *add-on* que realiza as funções discriminadas na Secção 2.5.1. O contentor do agente `master` é, por seu lado, responsável pela autenticação dos utilizadores. Sempre que esta acção sucede, o agente `master` cria um contentor e o respectivo agente JADE que irá processar os pedidos da aplicação cliente. O agente intermediário regista-se no `DF Agent` para permitir que os seus serviços possam ser descobertos.

3.3.3 Base de Dados

Para o sistema multi-agente suportar o armazenamento permanente de informação foi criada uma base dados onde estão representadas as várias entidades do sistema:

- Utilizadores;
- Dispositivos;
- Rotas;
- Serviços;
- Preferências;

O objectivo da divisão de entidades foi relacionar um utilizador com diversas rotas que este realize. Uma outra necessidade foi relacionar as preferências e serviços de um determinado utilizador. Adicionalmente, também se pretende guardar os diversos dispositivos utilizados.

Na Figura 3.9 está representado o modelo da base de dados que suporta a persistência da informação do módulo servidor.

Figura 3.9: Modelo da base de dados

Este diagrama entidade-relação, que foi gerado através do MySQL Workbench, encontra-se na notação *inverted arrow*.

3.4 Diagrama de Classes

O diagrama de classes do sistema multi-agente e o seu relacionamento é apresentado na Figura 3.10.

O nome de cada classe do sistema encontra-se a negrito. Sobre o nome de cada classe, *i.e.*, as palavras a itálico e alinhadas à direita, informam que essa classe é uma subclasse da classe indicada e/ou implementa uma determinada interface, *e.g.*, a classe `SOntology` é uma subclasse da classe Ontology e implementa a interface `ServicoVoca`.

A seta a tracejado indica que a classe de origem faz uso da classe de destino, *e.g.*, a classe `SlaveAgent` faz uso, entre outras, da classe `SOntology`.

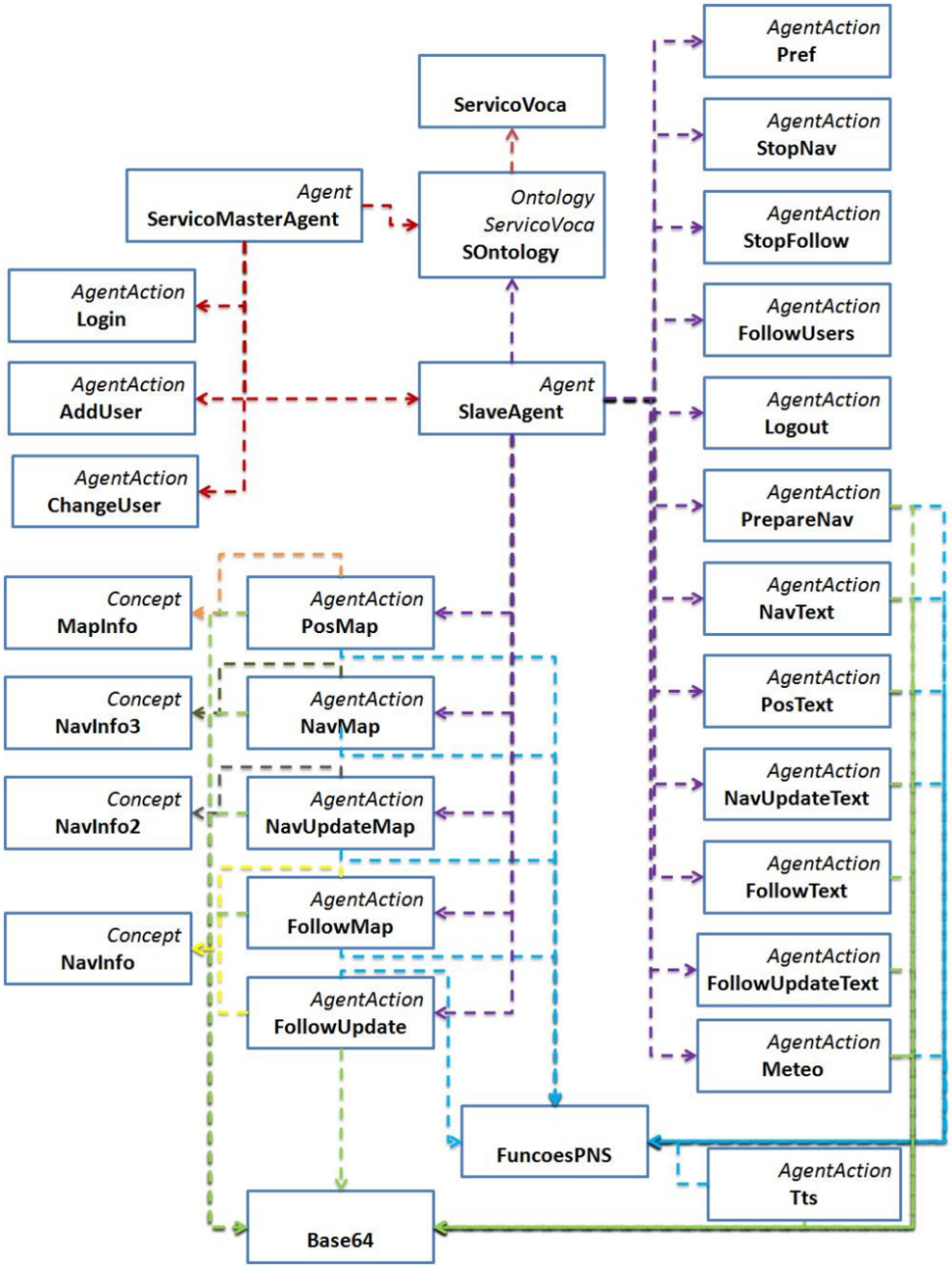

Figura 3.10: Diagrama de classes do sistema multi-agente

Os diagramas apresentados nas Figuras 3.11 e 3.12 representam o conjunto de classes da aplicação `MIDlet` J2ME e `Applet`, respectivamente.

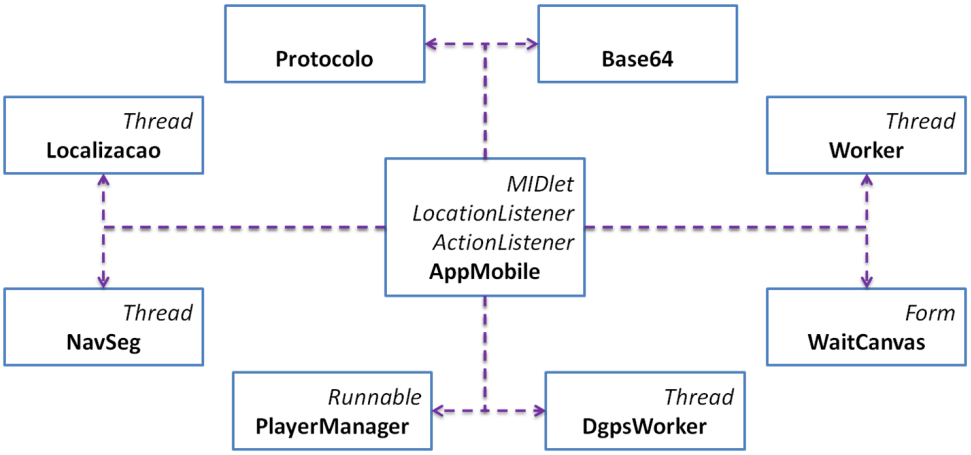

Figura 3.11: Diagrama de classes da aplicação `MIDlet` J2ME

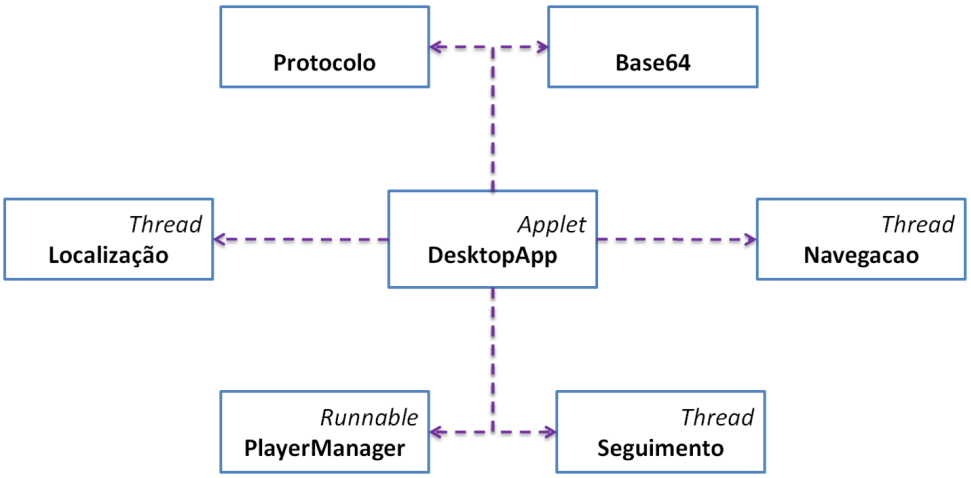

Figura 3.12: Diagrama de classes da aplicação `Applet` J2SE

3.5 Conclusão

Neste capítulo, dedicado à apresentação da arquitectura do sistema, analisou-se a vertente funcional e de implementação. Em relação à vertente funcional, foram definidos os casos de uso do sistema. No que respeita à implementação, procedeu-se à escolha justificada da plataforma de desenvolvimento do sistema multi-agente e foram descritos os diversos módulos da arquitectura assim como os respectivos diagramas de classes.

A garantia da criação de um sistema multi-agente conforme com as especificações FIPA e de elevada interoperabilidade foi assegurada através da adopção do

ambiente de desenvolvimento constituído pela plataforma JADE e pelo módulo complementar WSIG.

Capítulo 4

Desenvolvimento

Este capítulo descreve a etapa de realização do sistema e são justificadas as decisões técnicas tomadas.

O projecto desenvolvido, tal como foi referido no capítulo anterior, é constituído por três blocos principais: as fontes de dados, o módulo servidor e o módulo cliente. Dado que apenas as duas últimas foram implementadas no âmbito desta dissertação, apenas estas serão abordadas. No entanto, antes de o fazer, serão referidas as ferramentas utilizadas e será descrito o protocolo de nível de aplicação adoptado entre os módulos cliente e servidor.

4.1 Ferramentas Utilizadas

As principais ferramentas utilizadas no desenvolvimento do sistema foram obtidas de forma gratuita e ofereceram o suporte necessário e uma vasta documentação.

4.1.1 NetBeans

O NetBeans é um projecto *open-source* de sucesso, com um elevado número de utilizadores, uma crescente comunidade e inúmeros parceiros mundiais. A Sun MicroSystems fundou o projecto NetBeans em Junho de 2000 e continua sendo o seu principal patrocinador. Hoje, existem dois produtos: o IDE NetBeans e a Plataforma NetBeans [29]. Ambos os produtos são *open source* e livres para uso comercial e não comercial. O código fonte está disponível através da licença Common Development and Distribution License (CDDL).

Este IDE é um ambiente de desenvolvimento que permite criar, compilar, depurar e instalar programas. O IDE, que é completamente desenvolvido em

Java, é capaz de suportar outras linguagens de programação. De acordo com o *site* oficial, o NetBeans suporta as seguintes tecnologias: AJAX, C/C++, Databases, Debugger, Desktop, Editor, GUI Builder, Groovy, Java EE, Java ME, Java SE, Javascript, Mobile, Profiler, Refactor, REST, Rich Client Platform, Ruby, SOA, SOAP, UML, WSDL e XML. Existe ainda um elevado número de módulos que permitem expandir as capcidades do IDE NetBeans. O NetBeans IDE é um produto livre, sem restrições de utilização.

A NetBeans Platform é uma base modular e extensível que pode ser usada como infra-estrutura para a criação de grandes aplicações de secretária. Vários parceiros fornecem *plug-ins* que podem ser facilmente integrados na plataforma e que podem ser utilizados para desenvolver ferramentas e soluções próprias.

4.1.2 Java ME SDK

O Java ME SDK 3.0 é um pacote suplementar que oferece ferramentas para criar aplicações Java Micro Edition compatíveis com as tecnologias Connected Limited Device Configuration (CLDC) e Mobile Information Device Profile (MIDP), fornecendo um ambiente de programação similar para aparelhos móveis.

4.1.3 kXML

O kXML é um *parser* XML, especialmente concebido para ambientes com limitações como o J2ME (MIDlets) e para algumas aplicações J2SE (Applets). É caracterizado por implementar a API XMLPull. Actualmente, o kXML está na versão 2 (a versão 3 encontra-se em desenvolvimento) e é disponibilizado sobre a licença BSD de forma gratuita [30].

4.1.4 LWUIT

A Light Weight User Interface Toolkit (LWUIT) é uma biblioteca orientada para o mercado dos dispositivos móveis que permite criar Graphical User Interfaces (GUI) atractivas e amigáveis. A LWUIT, que é inspirada na API Swing do JSE, oferece recursos avançados para a construção de interfaces com o utilizador para dispositivos compatíveis com os ambientes de execução MIDP 1.1 e CLDC 2.0 [31].

Esta biblioteca é uma tecnologia aberta, com o respectivo código fonte e ficheiros de instalação de acesso gratuito para uso pessoal ou comercial.

4.1.5 Tomcat

O Tomcat é um servidor de aplicações desenvolvido no âmbito do projecto Apache Jakarta da SUN. Desenvolvido integralmente em Java, constitui a Implementação de Referência (RI) das tecnologias Servlet e Java Server Pages (JSP) para o desenvolvimento de aplicações *Web*.

4.1.6 eSpeak

O eSpeak é um sintetizador de voz *open source* para sistemas operativos Windows, Linux, Solaris e Mac OSX. Esta aplicação permite utilizar várias línguas e usa o designado método "formant synthesis". Apesar de a voz emulada não ser tão natural como a dos sintetizadores mais complexos, que são baseados em gravações da fala humana, este método gera num curto espaço de tempo uma voz emulada bastante perceptível [32].

4.1.7 Switch Sound File Converter

A aplicação Switch Sound File Converter permite converter, codificar e comprimir arquivos de som de forma rápida e fácil, suportando uma grande variedade de formatos áudio [33].

4.2 Protocolo do Nível de Aplicação

Uma vez que o módulo servidor disponibiliza os seus serviços através de uma interface do tipo serviço *Web*, foi necessário especificar e implementar um protocolo de nível de aplicação que defina as mensagens, neste caso do tipo SOAP sobre HTTP, trocadas entre cliente e servidor. No Anexo A são apresentados exemplos das mensagens SOAP de cada operação de utilização dos serviços oferecidos pelo sistema multi-agente.

Nas Tabelas 4.1 e 4.2 são descritas as operações oferecidas pelos agentes às aplicações cliente.

Tabela 4.1: Operações módulo servidor

Operações	Parâmetros da Operação		Descrição
	Nome	Tipo de dados	
login	user	String	Método para autenticação de um cliente e criação do agente de atendimento desse utilizador
	pass	String	
	dev	String	
addUser	user	String	Método para a criação de um novo utilizador
	newUser	String	
	pass	String	
	level	int	
	servicesIntRef	String	
changeUser	clild	int	Método para alterar os dados de um utilizador
	newUser	String	
	pass	String	
	level	int	
pref	clild	int	Método para actualizar as preferências do utilizador
	servicesIntRef	String	
posMap	latMarker	String	Método para a utilização do Serviço de Posicionamento, devolvendo informação gráfica e textual
	lngMarker	String	
	zoom	int	
posText	latMarker	String	Método para a utilização do Serviço de Posicionamento, devolvendo apenas informação textual
	lngMarker	String	
prepareNav	clild	int	Método para descobrir os destinos encontrados tendo em conta o local especificado
	destination	String	
navMap	user	String	Método para a utilização do Serviço de Navegação, devolve informação gráfica, textual e sonora
	origin	String	
	destination	String	
navUpdateMap	routeInfo	String	Método para requisitar a actualização do Serviço de Navegação
	latMarker	String	
	lngMarker	String	
navText	origin	String	Método para a utilização do Serviço de Navegação, devolve informação textual
	destination	String	
navUpdateText	routeInfo	String	Método para requisitar a actualização do Serviço de Navegação
	latMarker	String	
	lngMarker	String	
navStop	clild	int	Método para parar a utilização do Serviço de Navegação
	routeInfo	String	

Tabela 4.2: Operações do módulo servidor (cont.)

Operações	Parâmetros da Operação		Descrição
	Nome	Tipo de dados	
followUsers	user	String	Método para descobrir os utilizadores a utilizar o serviço de Navegação
followMap	clildFol user	int String	Método para a utilização do Serviço de Seguimento, devolve informação gráfica e textual
followUpdate	routeInfo	int	Método para requisitar a actualização do Serviço de Seguimento
followText	clildFol user	int String	Método para a utilização do Serviço de Seguimento, devolve informação textual
followUpdateText	routeInfo	int	Método para requisitar a actualização do Serviço de Seguimento
followStop	routeInfo clildFol	int int	Método para parar a utilização do Serviço de Seguimento
tts	type text	String String	Método utilizado para criar as indicações de voz
meteo	destination	String	Método para utilizar o Serviço de Meteorologia
logout	clild	int	Método para terminar o agente de atendimento de um utilizador autenticado

4.3 Módulo Cliente

O módulo cliente desenvolvido é constituído por duas aplicações que constituem a interface do utilizador com o sistema e, consequentemente, permitem aceder aos serviços prestados pelo módulo servidor. Estes dois módulos comunicam entre si através das mensagens SOAP referidas na secção anterior.

4.3.1 Aplicação `MIDlet` J2ME

A aplicação `MIDlet` permite a interacção com o sistema através de um dispositivo móvel, *i.e.*, a partir de um telemóvel, PDA ou *smartphone*.

Este módulo foi desenvolvido recorrendo ao IDE NetBeans e à biblioteca adicional kXML 2.0 e foi implementado com a Java Platform, Micro Edition 3.0.

A escolha do NetBeans IDE para o desenvolvimento do `MIDlet` prende-se com: (*i*) o facto de permitir a integração da ferramenta de desenvolvimento de

aplicações móveis Java ME SDK; (*ii*) ser grátis e de código aberto; (*iii*) e da experiência de utilização adquirida durante o percurso académico.

A biblioteca kXML 2.0 foi escolhida em detrimento de outras como, *e.g.*, a ASXMLP ou a NanoXML, pelas seguintes razões [34]: (i) ser um dos *parsers* de tamanho mais pequeno existentes; (ii) ter sido projectada para dispositivos MIDP; (iii) ser um *parser* do tipo *pull*, o que significa que as aplicações podem processar e exibir as informações enquanto o documento XML é analisado, não sendo assim necessário analisar previamente a totalidade do documento.

Durante o desenvolvimento deste módulo foi criada uma versão que utilizava a biblioteca LWUIT 1.4, destinada a criar uma interface gráfica para o utilizador mais atractiva, mas que não foi implementada na versão final porque tornava a aplicação cliente demasiado pesada, provocando um mau funcionamento e, consequentemente, constituindo uma má experiência de utilização para o utilizador.

De referir, a título de curiosidade, que o tamanho da aplicação com esta biblioteca era aproximadamente de 1 MiB, enquanto a versão final, desenvolvida usando os componentes gráficos disponíveis por omissão, possui apenas 250 KiB.

Nas Figuras 4.1, 4.2 e 4.3 apresentam-se os ecrãs que a aplicação teria caso a versão final utilizada tivesse sido a que inclui a biblioteca LWUIT.

Figura 4.1: Ecrãs do `MIDlet` J2ME com a biblioteca LWUIT

Figura 4.2: Ecrãs do `MIDlet` J2ME com a biblioteca LWUIT (cont.)

Figura 4.3: Ecrãs do `MIDlet` J2ME com a biblioteca LWUIT (cont.)

O módulo J2ME é uma aplicação com diversos ecrãs que permitem efectuar a autenticação, a selecção do serviço(s) pretendido(s) e a definição das preferências do utilizador.

A autenticação é obrigatória não só como medida de segurança, mas também como forma de identificação do utilizador. Assim, no início, o utilizador tem de se autenticar para ter acesso aos serviços. Na Figura 4.4 é apresentado o ecrã que permite a alteração do endereço IP do módulo servidor e a autenticação do utilizador mediante a introdução do nome de utilizador e respectiva senha.

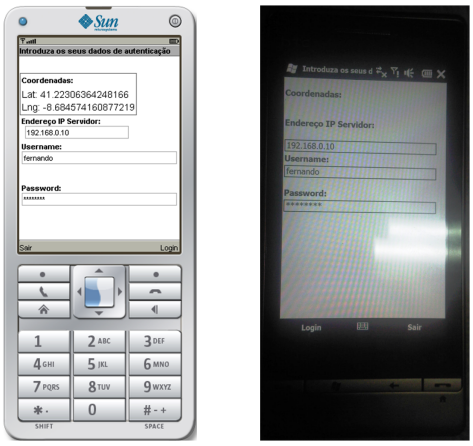

Figura 4.4: Ecrã de autenticação do `MIDlet` J2ME no emulador e telemóvel

Após a correcta autenticação, o utilizador é presenteado com um ecrã contendo o último serviço utilizado ou, caso se trate da primeira autenticação de um novo utilizador, um menu com as diversas funcionalidades da aplicação. Este menu é composto por seis opções: Posicionamento, Navegação, Seguimento, Meteorologia e DGPS (ver Figuras 4.5 e 4.6).

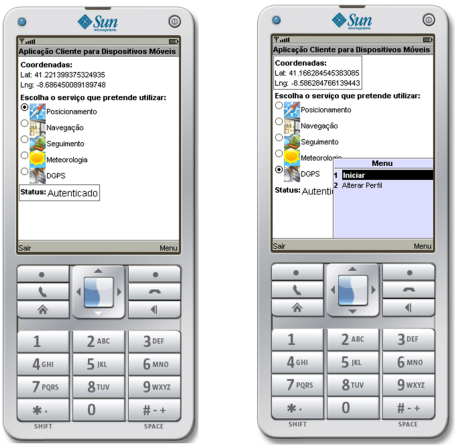

Figura 4.5: Ecrã do menu principal do `MIDlet` J2ME no emulador

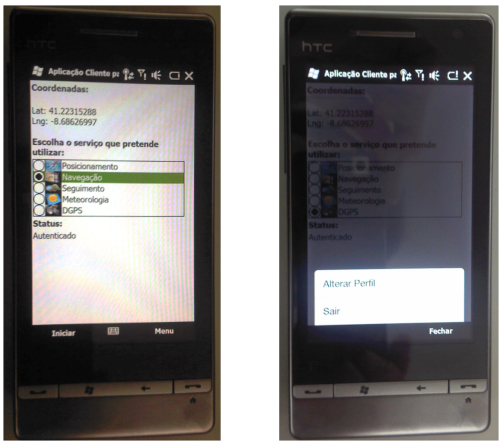

Figura 4.6: Ecrã do menu principal do `MIDlet` J2ME no telemóvel

Serviço de Posicionamento — A selecção do Serviço de Posicionamento resulta na apresentação de uma imagem que contém: (*i*) um mapa da localização actual; (*ii*) informação textual, incluindo a morada aproximada do local onde se encontra o móvel; e (*iii*) informação sonora da morada aproximada especificada na informação textual. Neste ecrã estão ainda disponíveis as opções de zoom do mapa e de actualização da localização actual do utilizador como se ilustra na Figuras 4.7, 4.8.

Figura 4.7: Ecrãs da opção posicionamento do `MIDlet` J2ME no emulador

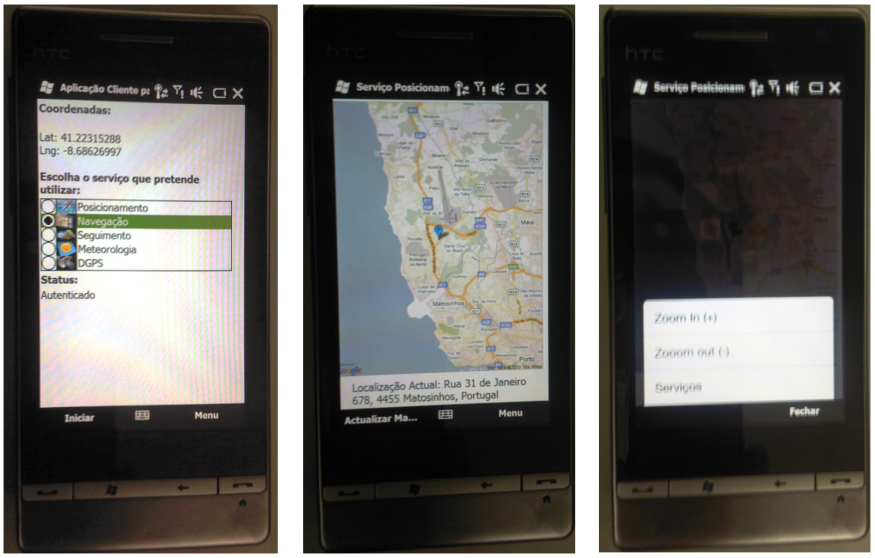

Figura 4.8: Ecrãs da opção posicionamento do `MIDlet` J2ME no telemóvel

Serviço de Navegação — A selecção do Serviço de Navegação permite obter um trajecto entre a localização actual e um local especificado pelo utilizador. A informação disponibilizada é composta por: (*i*) uma imagem contendo um mapa com a indicação do local de partida, da localização actual e do local de chegada; (*ii*) algumas informações textuais sobre o trajecto como, *e.g.*, a distância total e o tempo de duração previsto, assim como as instruções que o utilizador deve seguir para chegar ao destino; (*iii*) as indicações sonoras do trajecto. Nas Figuras 4.9 e 4.10 apresentam-se os ecrãs do Serviço de Navegação no emulador e no telemóvel.

Figura 4.9: Ecrãs da opção navegação do `MIDlet` J2ME no emulador

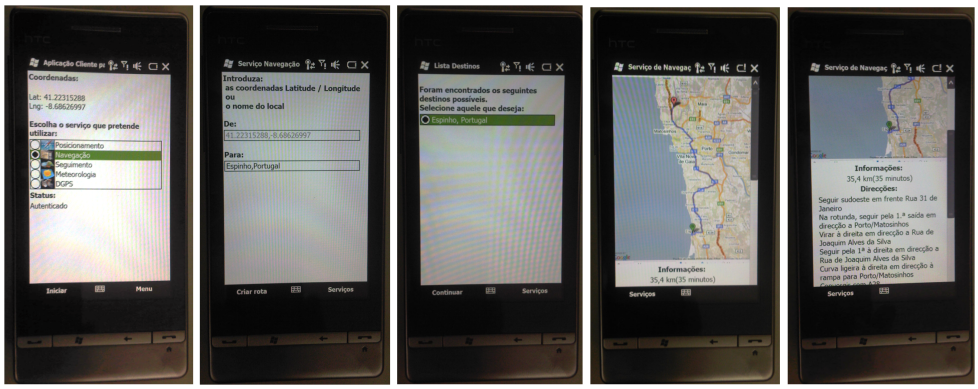

Figura 4.10: Ecrãs da opção navegação do `MIDlet` J2ME no telemóvel

Serviço de Seguimento — A selecção do Serviço de Seguimento devolve: (*i*) um mapa com a localização actual e o trajecto que o utilizador está a realizar; e (*ii*) informação textual com as moradas aproximadas dos locais de partida, actual e do destino especificado. Nas Figuras 4.11 e 4.12 apresentam-se os ecrãs do Serviço de Navegação no emulador e no telemóvel.

Figura 4.11: Ecrãs da opção seguimento do `MIDlet` J2ME no emulador

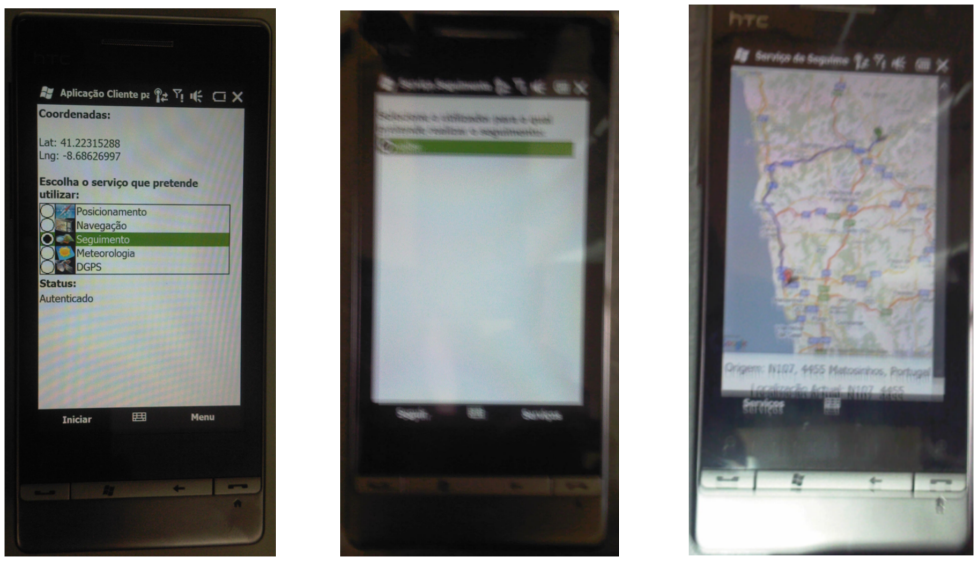

Figura 4.12: Ecrãs da opção seguimento do `MIDlet` J2ME no telemóvel

Serviço de Meteorologia — A selecção do Serviço de Meteorologia permite obter a temperatura actual, a percentagem de humidade, a temperatura mínima e a máxima prevista, a direcção e velocidade do vento e a previsão meteorológica para os próximos três dias de um local (Figuras 4.13 e 4.14).

Figura 4.13: Ecrãs da opção meteorologia do `MIDlet` J2ME no emulador

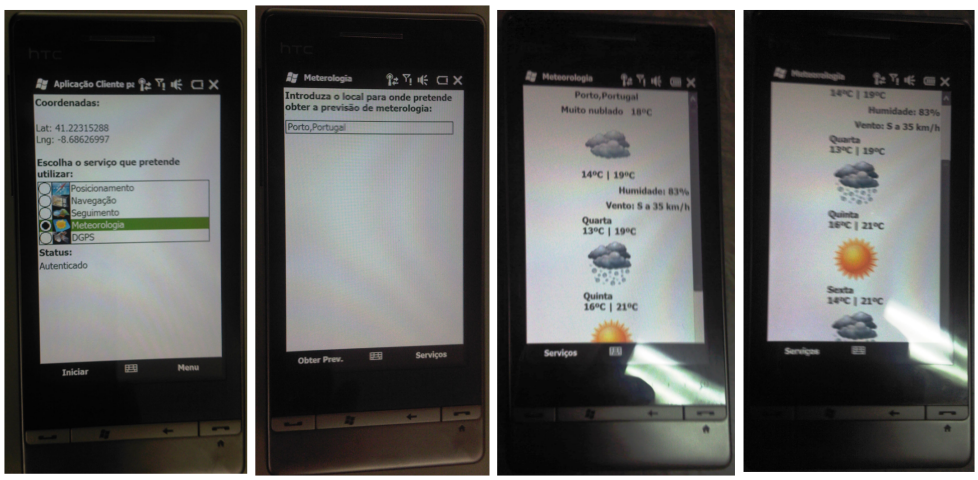

Figura 4.14: Ecrãs da opção meteorologia do `MIDlet` J2ME no telemóvel

DGPS — O Serviço DGPS permite estabelecer a ligação a um servidor de correcções diferenciais DGPS e receber as respectivas correcções. Esta informação deve ser enviada e aplicada directamente pelo receptor GPS, não apresentando qualquer tipo de informação ao utilizador. Nas Figuras 4.15 e 4.16 apresentam-se os ecrãs do Serviço de DGPS no emulador e no telemóvel.

Figura 4.15: Ecrãs da opção DGPS do `MIDlet` J2ME no emulador

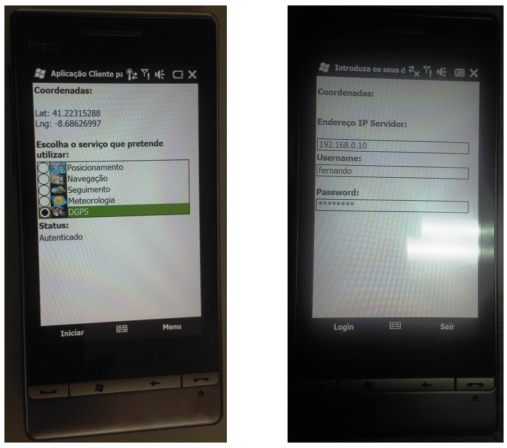

Figura 4.16: Ecrãs da opção DGPS do `MIDlet` J2ME no telemóvel

Perfil — Opção Perfil permite alterar as preferências do utilizador. Nas Figuras 4.17 e 4.18 apresentam-se os ecrãs do Serviço de DGPS no emulador e no telemóvel.

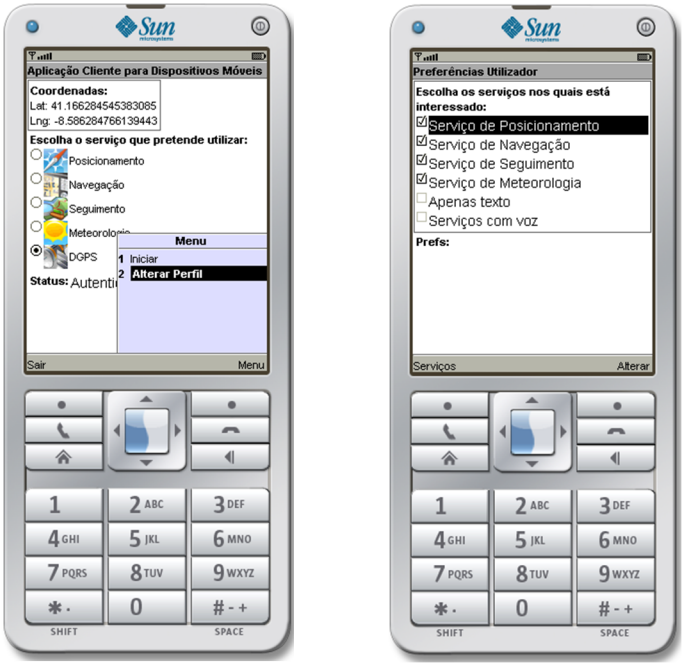

Figura 4.17: Ecrãs da opção alterar perfil do `MIDlet` J2ME no emulador

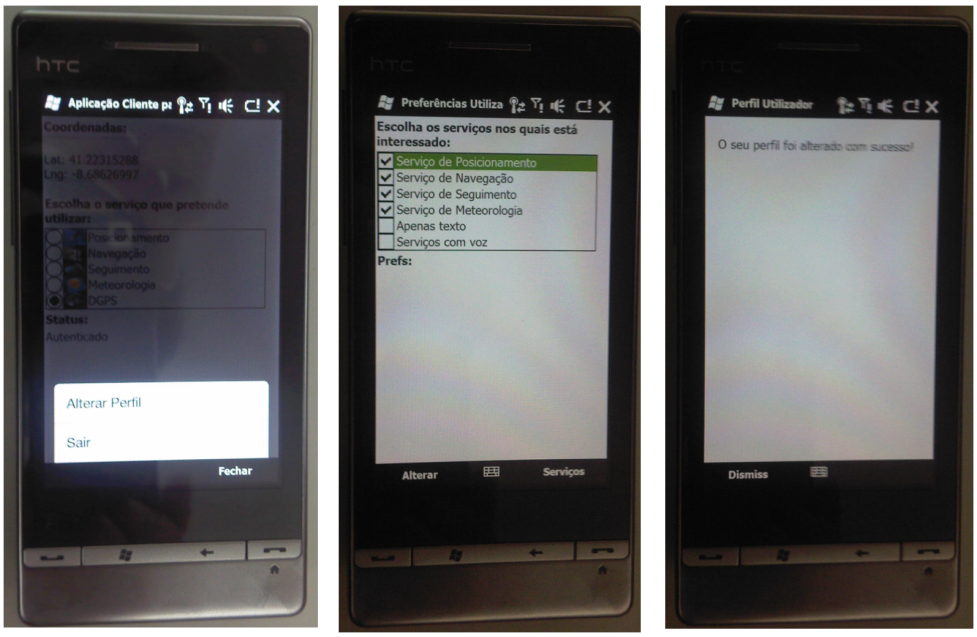

Figura 4.18: Ecrãs da opção alterar perfil do `MIDlet` J2ME no telemóvel

4.3.2 Aplicação `Applet` J2SE

O módulo `Applet` implementado permite aos utilizadores interagir com o sistema multi-agente a partir de um computador de secretária, portátil ou *netbook*.

Esta aplicação foi desenvolvida recorrendo ao IDE NetBeans, à API Java SE 6.0, a componentes gráficos da biblioteca `Swing` e ao modelo de processamento de eventos 1.1.

Na Figura 4.19 apresenta-se a interface do `Applet` desenvolvida que é constituída uma barra de menus e quatro janelas internas.

As quatro janelas internas são usadas para mostrar, respectivamente, a informação do Serviço de Posicionamento, do Serviço de Navegação, do Serviço de Seguimento e do histórico das mensagens enviadas e recebidas.

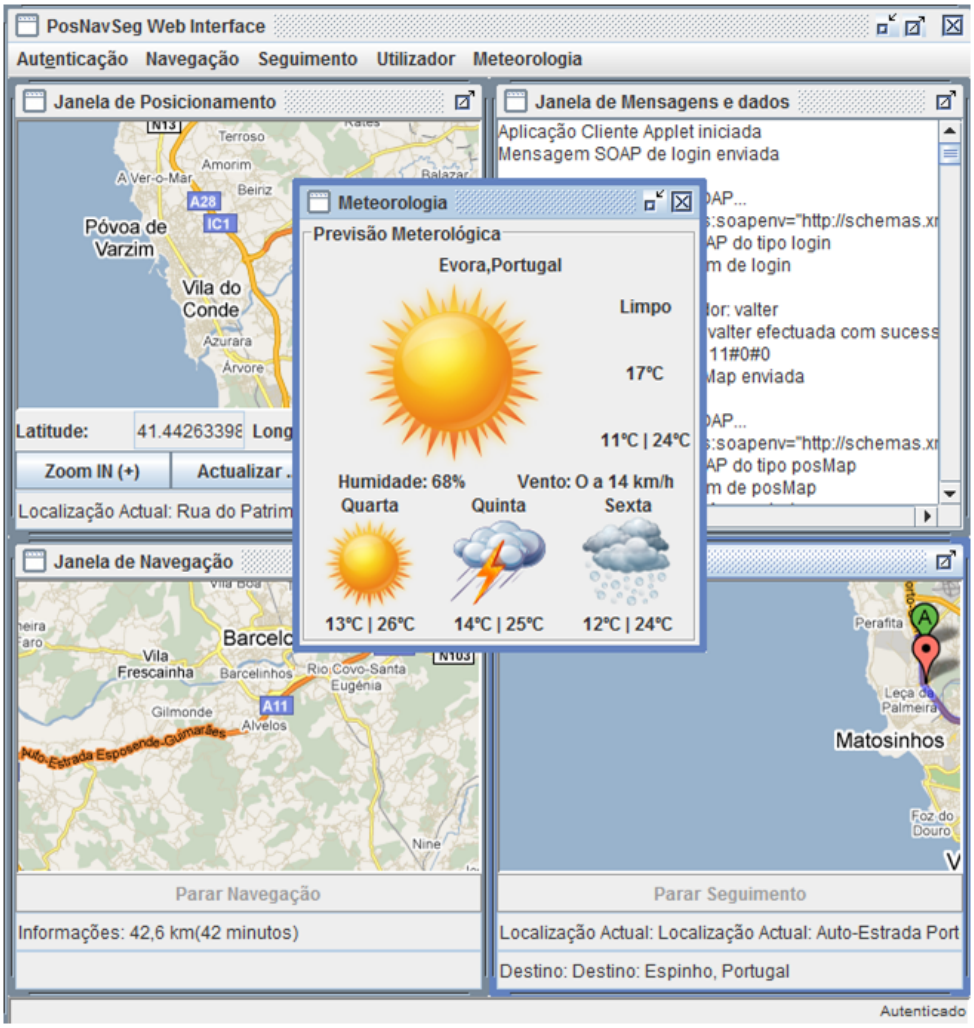

Figura 4.19: Ecrã da aplicação `Applet` J2SE

A barra de menus é composta: (*i*) o Menu de Autenticação para realizar a autenticação do utilizador; (*ii*) o Menu de Navegação que permite utilizar o Serviço de Navegação, criando uma nova rota; (*iii*) o Menu de Seguimento para aceder ao Serviço de Seguimento, escolhendo o utilizador que se pretende seguir; (*iv*) o Menu de Meteorologia para obter informações meteorológicas do local especificado; e (*v*) o Menu do Utilizador para criar um novo utilizador e alterar os dados e as preferências do utilizador autenticado.

4.4 Módulo Servidor

O módulo servidor, na qualidade de sistema intermediário entre clientes e fontes de dados, suporta as operações realizadas pelas aplicações cliente (móveis ou fixas)

e é responsável pela gestão de todo o sistema. Este módulo, que foi desenvolvido na plataforma de desenvolvimento de agentes JADE, obrigou à instalação prévia da J2SE, do MySQL e da API Conector J [35] de acesso ao sistema de gestão de bases de dados MySQL [36]. A base de dados foi criada de acordo com o modelo de dados apresentado no capítulo anterior.

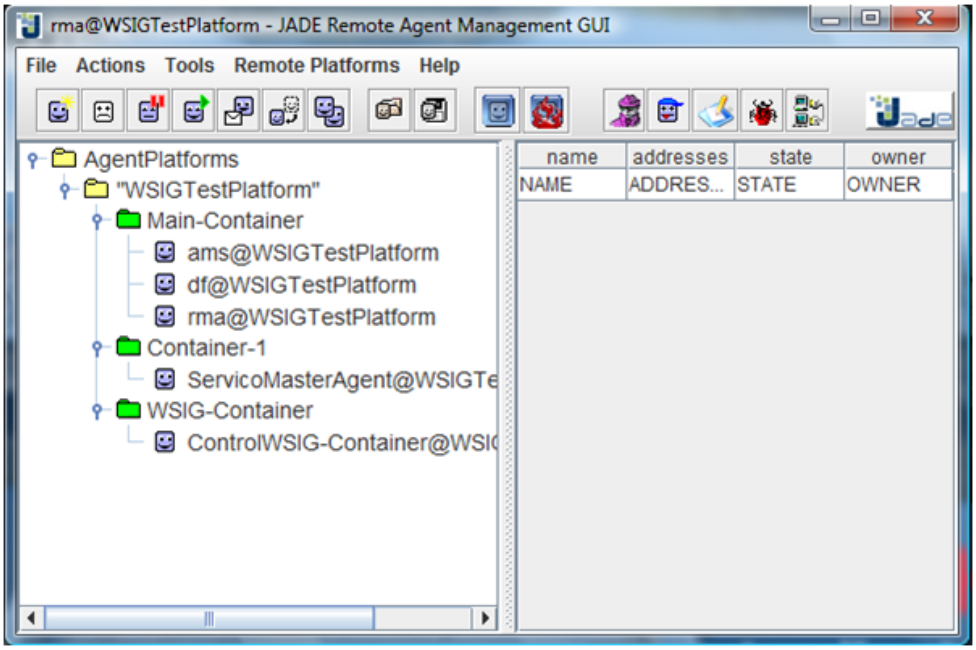

Figura 4.20: Janela GUI da plataforma JADE

Quando se lança o sistema multi-agente, são criados, para além dos agentes `AMS`, DF e `RMA` da plataforma JADE, o agente `ControlWSIG` e o agente `ServicoMasterAgent` como se ilustra na Figura 4.20. Após a autenticação com sucesso de um utilizador junto do agente `ServicoMasterAgente`, este cria um novo agente, do tipo `SlaveAgent`, que passa a atender os pedidos desse utilizador. A Figura 4.21 representa a situação em que três utilizadores se encontram autenticados. Como se pode ver, a aplicação é agora composta por mais três agentes designados `AgenteSlave2`, `AgenteSlave1` e `AgenteSlave5`, respectivamente.

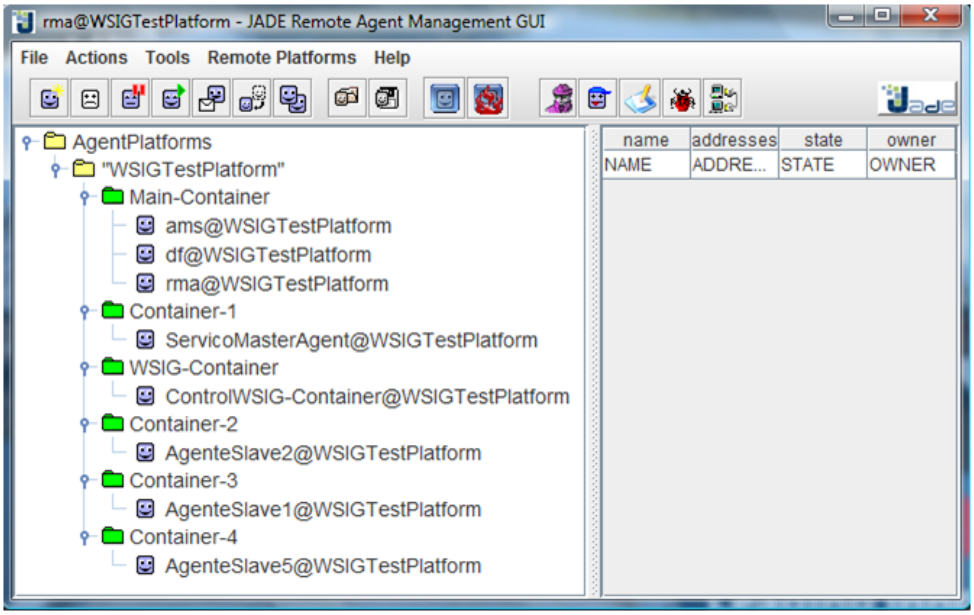

Figura 4.21: Janela GUI da plataforma JADE II

4.4.1 Operações

Nesta secção são apresentadas as acções realizadas pelo módulo servidor em relação a cada uma das operações mencionadas na secção do protocolo de nível de aplicação.

Operação *login*

- Verifica se um utilizador já se encontra autenticado no sistema multi-agente;
- Comprova, junto da base de dados, se o par nome de utilizador e senha introduzido está correcto;
- Contacta a base de dados para verificar se o modelo do dispositivo (que é caracterizado pelas dimensões do seu ecrã) utilizado pelo cliente existe; caso contrário, cria-o;
- Cria, caso a autenticação esteja correcta, um agente do tipo `SlaveAgent` para atender os pedidos do utilizador.

Operação *addUser*

- Contacta a base de dados com o intuito de:

  - averiguar se o utilizador que pretender criar um novo utilizador possui permissões para o efeito;
  - confirmar se o novo nome de utilizador especificado está disponível;

- Cria, se possível, o novo utilizador.

Operação *pref*

- Contacta a base de dados para alterar as preferências do utilizador.

Operação *posMap*

- Actualiza as coordenadas geográficas com a localização do utilizador;
- Contacta o servidor do Google Maps e obtém o mapa de posicionamento;
- Ao receber um pedido de actualização do mapa de posicionamento, verifica se a localização actual ainda se encontra dentro da região verde do mapa enviado anteriormente (ver Figura 4.22). Em caso afirmativo, obtém o novo mapa de posicionamento por reutilização do mapa que possui, calculando a diferença entre o mapa já enviado e o novo mapa com a posição actual. Caso contrário, se o utilizador se encontra na região a vermelho do mapa anteriormente enviado, contacta de novo o servidor do Google Maps e adquire um novo mapa;

Figura 4.22: Actualização do mapa de posicionamento

- Contacta o servidor do Google Maps e, utilizando o serviço de *geocoding*, converte as coordenadas geográficas actuais do utilizador numa morada aproximada. A resposta consiste num documento XML de onde é extraída a informação pretendida;

- Codifica numa `string` do tipo Base 64 a imagem do mapa de posicionamento;
- Altera na base de dados o último serviço utilizado pelo cliente.

Operação *posText*

- Actualiza as coordenadas geográficas com a localização do utilizador;
- Contacta o servidor do Google Maps e, utilizando o serviço de *geocoding*, converte as coordenadas geográficas actuais do utilizador numa morada aproximada. A resposta consiste num documento XML de onde é extraída a informação pretendida;
- Codifica numa `string` do tipo Base 64 a informação obtida do documento XML;
- Actualiza na base de dados o último serviço utilizado pelo cliente.

Operação *prepareNav*

- Contacta o servidor do Google Maps e, utilizando o serviço de *geocoding*, obtém todos os destinos encontrados que satisfazem a informação submetida pelo utilizador. A resposta consiste num documento XML de onde são extraídos cada um desses destinos;
- Codifica numa `string` do tipo Base 64 a informação retirada do documento XML.

Operação *navMap*

- Comunica com a base de dados e cria uma nova rota;
- Constrói uma página HTML que cria o mapa de navegação;
- Contacta o servidor do Google Maps e, utilizando o serviço de direcções, obtém as indicações de navegação entre os locais de partida e de chegada sob a forma de um documento XML. De seguida, contacta o serviço de geocoding e adquire as moradas aproximadas das localizações actual e de destino sob a forma de um documento XML;
- Codifica em `strings` do tipo Base 64 a imagem do mapa de navegação e a informação retirada dos documentos XML;
- Actualiza na base de dados o último serviço utilizado pelo cliente.

Operação *navUpdateMap*

- Actualiza as coordenadas geográficas do utilizador;
- Actualiza na base de dados as coordenadas geográficas actuais do cliente na rota criada na operação anterior (*navMap*);
- Constrói uma página HTML que cria o mapa de navegação actualizado e calcula a imagem diferença entre o novo mapa e o anterior;
- Codifica em `strings` do tipo Base 64 a imagem do mapa de navegação assim como a informação retirada do documento XML.

Operação *navText*

- Comunica com a base de dados e cria uma nova rota;
- Contacta o servidor do Google Maps e obtém, em primeiro lugar, do serviço de direcções, as indicações de navegação entre os locais de partida e o de chegada e, em segundo lugar, do serviço de *geocoding*, as moradas aproximadas das localizações actual e de destino;
- Codifica numa `string` do tipo Base 64 a informação retirada do documento XML;
- Actualiza na base de dados o último serviço utilizado pelo cliente.

Operação *navUpdateText*

- Actualiza as coordenadas geográficas do utilizador;
- Comunica com a base de dados e actualiza as coordenadas geográficas actuais do cliente na rota criada na operação anterior (*navText*);
- Contacta o servidor do Google Maps e utilizando o serviço de *geocoding* deste adquire a morada aproximada da localização actual;
- Codifica numa `string` do tipo Base 64 a informação retirada do documento XML.

Operação *navStop*

- Actualiza na base de dados o estado da rota para inactiva.

Operação *followUsers*

- Contacta a base de dados e verifica se o utilizador possui permissão para utilizar este serviço. Em caso afirmativo, obtém a lista dos utilizadores que estão a utilizar o serviço de Navegação.

Operação *followMap*

- Actualiza na base de dados o conjunto de utilizadores que se encontram a seguir uma determinada rota;
- Constrói uma página HTML que cria o mapa de seguimento;
- Actualiza na base de dados o último serviço utilizado pelo cliente;
- Contacta o servidor do Google Maps e, utilizando o serviço de *geocoding*, converte os locais de partida, actual e de chegada em moradas aproximadas;
- Codifica em `strings` do tipo Base 64 a imagem do mapa de navegação assim como a informação retirada do documento XML.

Operação *followUpdate*

- Caso as dimensões do ecrã do dispositivo do utilizador que está a ser seguido sejam iguais às do utilizador que está a utilizar o Serviço de Seguimento, não é necessário obter um novo mapa de seguimento. Caso contrário, é criada uma página HTML com o novo mapa de seguimento;
- Contacta o servidor do Google Maps e, utilizando o serviço de *geocoding*, converte a localização actual do utilizador seguido numa morada aproximada;
- Codifica em `strings` do tipo Base 64 a imagem com mapa de navegação assim como a informação retirada do documento XML.

Operação *followText*

- Actualiza na base de dados o último serviço utilizado pelo cliente;
- Contacta o servidor do Google Maps e, utilizando o serviço de *geocoding*, converte os locais de partida, actual e de chegada em moradas aproximadas;
- Codifica numa `string` do tipo Base 64 a informação retirada do documento XML.

Operação *followUpdateText*

- Contacta o servidor do Google Maps e, utilizando o serviço de *geocoding*, converte a localização actual do utilizador seguido numa morada aproximada;
- Codifica numa `string` do tipo Base 64 a informação retirada do documento XML.

Operação *followStop*

- Altera junto da base de dados os utilizadores que se encontram a seguir uma determinada rota.

Operação *meteo*

- Contacta o servidor do Google Weather e obtém a informação meteorológica para o local especificado pelo cliente;
- Codifica numa `string` do tipo Base 64 a informação retirada do documento XML.

Operação *logout*

- Actualiza na base de dados a data da última sessão do utilizador
- Elimina o agente de atendimento deste utilizador.

4.5 Conclusão

Neste capítulo, consagrado à etapa do desenvolvimento do sistema, aduziu-se o ambiente de desenvolvimento, as plataformas e pacotes de linguagem Java adoptadas, referiram-se as aplicações externas utilizadas, descreveu-se a implementação dos módulos cliente e servidor e apresentou-se o protocolo de comunicação baseado em SOAP criado.

Capítulo 5

Avaliação do Sistema

Neste capítulo são descritos os testes efectuados ao sistema para avaliar o grau de cumprimento dos objectivos pré-definidos. Detalha-se também todo o ambiente de hardware e software utilizado na execução dos testes.

5.1 Ambiente de Testes

Os testes decorreram em ambiente público e recorreram a *hardware* de carácter pessoal, designadamente para o desenvolvimento do sistema multi-agente, e a equipamento cedido pelo DEE, o dispositivo móvel, onde foi instalado e executado o módulo `MIDlet`. De referir que não foi possível efectuar testes em ambiente de produção, visto que isso implicaria a aquisição de diverso material para distribuir pelos utilizadores.

Os testes descritos neste capítulo foram realizados com recurso aos seguintes módulos:

- Sistema multi-agente – Uma máquina com o Windows Vista, o J2SE, o JADE, o sistema de gestão de base de dados MySQL, o servidor de aplicações *Web* Tomcat e os programas eSpeak e Switch;
- Cliente `MIDlet` – Um HTC Diamond Touch 2 com o Windows Mobile 6.5 e a KVM Esmerteb Jbed v20090416.5.1;
- Cliente `Applet` – Uma máquina com o Windows XP, JRE e um receptor GPS.

Em todos os testes considerou-se que os módulos clientes (`MIDlet` e `Applet`) se encontravam devidamente instalados no dispositivo cliente e que a autenticação tinha sido realizada com sucesso.

5.1.1 Resposta Temporal

O tempo é um factor crítico na maior parte das aplicações que devem apresentar resultados em tempo útil. Num cenário composto por um grande número de utilizadores, alguns segundos podem ser determinantes para o sucesso ou fracasso de um sistema. Por essa razão, apresentam-se os resultados da resposta temporal dos testes relativos às principais operações do sistema.

Serviço de Posicionamento: O teste do Serviço de Posicionamento consistiu na selecção no menu principal da respectiva opção e na consequente recepção, processamento e apresentação da resposta ao utilizador. Nas Tabelas 5.1, 5.2 e 5.3 são apresentados os tempos de cada uma das actividades realizadas tanto com o emulador J2ME integrado no NetBeans, com o `MIDlet` no dispositivo móvel e com o módulo `Applet` num computador. Na realização deste teste foram feitas 10 amostragens.

Tabela 5.1: Tempos de utilização do Serviço de Posicionamento no emulador

Actividade	Tempo Mínimo (s)	Tempo Médio (s)	Tempo Máximo (s)
Envio do Pedido e Recepção da Resposta	3,022	4,488	6,350
Processamento da Resposta	0,194	0,291	0,402
Apresentação do Ecrã	0,012	0,017	0,027
Total	3,228	4,796	6,779

Tabela 5.2: Tempos de utilização do Serviço de Posicionamento no telemóvel

Actividade	Tempo Mínimo (s)	Tempo Médio (s)	Tempo Máximo (s)
Envio do Pedido e Recepção da Resposta	2,922	3,391	4,076
Processamento da Resposta	3,161	3,594	4,275
Apresentação do Ecrã	0,073	0,082	0,127
Total	6,156	7,066	8,478

Tabela 5.3: Tempos de utilização do Serviço de Posicionamento no `Applet`

Actividade	Tempo Mínimo (s)	Tempo Médio (s)	Tempo Máximo (s)
Envio do Pedido e Recepção da Resposta	0,953	1,431	2,531
Processamento da Resposta	0,047	0,134	0,390
Apresentação do Ecrã	0,047	0,320	0,844
Total	1,047	1,886	3,765

Serviço de Navegação: O teste do Serviço de Navegação, que é composto pelas operações *prepareNav, navMap, navUpdateMap e navStop*, consistiu na selecção no menu principal da respectiva opção e na consequente recepção, processamento e apresentação das respostas ao utilizador. Nas Tabelas 5.4, 5.5 e 5.6 são apresentados os tempos de cada uma das actividades realizadas tanto no emulador J2ME integrado no NetBeans e nos módulos cliente `MIDlet`, executado no dispositivo móvel HTC, e `Applet`, executado num computador. Foram realizadas 10 amostragens.

Neste caso, simulou-se a utilização do serviço da seguinte forma:

- Criou-se um ficheiro com um percurso de extensão aproximada de 45 km;
- Executou-se o Serviço de Navegação utilizando o ficheiro anterior que emula a recepção das coordenadas do módulo cliente.

De referir, que durante esta simulação, o módulo cliente e o módulo servidor trocaram as seguintes mensagens:

- 1 (uma) mensagem de pedido (módulo cliente) e outra de resposta (módulo servidor) à operação *prepareNav*;
- 1 (uma) mensagem de pedido e outra de resposta à operação *navMap*;
- 45 (quarenta e cinco) mensagens de pedido e outras tantas de resposta relativas à operação *navUpdateMap*;
- 1 (uma) mensagem de pedido e outra de resposta à operação *navStop*.

No total, a utilização deste serviço correspondeu à troca de 96 (noventa e seis) mensagens. É ainda de realçar que o intervalo de tempo entre os pedidos de actualização da informação (operação *navUpdateMap*) é de 30 s.

Tabela 5.4: Tempos de utilização do Serviço de Navegação no emulador

Actividade		Tempo Mínimo (s)	Tempo Médio (s)	Tempo Máximo (s)
1 x	Envio e Recepção mensagens operação *prepareNav*	0,296	0,377	0,378
1 x	Envio e Recepção mensagens operação *navMap*	7,296	7,449	7,601
45 x	Intervalo entre cada envio das mensagens da operação *navUpdateMap*	30 s		
45 x	Envio e Recepção mensagens operação *navUpdateMap*	6,399	6,403	6,407
1 x	Envio e Recepção mensagens operação *navStop*	0,234	0,244	0,254
Total		1652,18 ≈ 27,53 min	1652,57 ≈ 27,54 min	1652,99 ≈ 27,55 min

Tabela 5.5: Tempos de utilização do Serviço de Navegação no telemóvel

Actividade		Tempo Mínimo (s)	Tempo Médio (s)	Tempo Máximo (s)
1 x	Envio e Recepção mensagens operação *prepareNav*	0,707	0,823	0,929
1 x	Envio e Recepção mensagens operação *navMap*	9,529	10,078	10,628
45 x	Intervalo entre cada envio das mensagens da operação *navUpdateMap*	30 s		
45 x	Envio e Recepção mensagens operação *navUpdateMap*	7,226	7,229	8,233
1 x	Envio e Recepção mensagens operação *navStop*	0,545	0,577	0,608
Total		1693,19 ≈ 28,22 min	17172,04 ≈ 28,62 min	1740,88 ≈ 29,01 min

Tabela 5.6: Tempos de utilização do Serviço de Navegação no `Applet`

Actividade		Tempo Mínimo (s)	Tempo Médio (s)	Tempo Máximo (s)
1 x	Envio e Recepção mensagens operação *prepareNav*	0,328	0,352	0,375
1 x	Envio e Recepção *mensagens operação navMap*	10,031	12,039	14,047
45 x	Intervalo entre cada envio das mensagens da operação *navUpdateMap*	30 s		
45 x	Envio e Recepção mensagens operação *navUpdateMap*	8,023	9,893	11,762
1 x	Envio e Recepção mensagens operação *navStop*	0,156	0,188	0,219
Total		1729,57 ≈ 28,83 min	1817,63 ≈ 30,29 min	31,762 ≈ 31,76 min

Serviço de Seguimento: O teste da utilização do Serviço de Seguimento consistiu na selecção no menu principal da opção correspondente, que desencadeia as operações *followUsers, followMap, followUpdate e followStop*, e na consequente recepção, processamento e apresentação da informação recebida ao utilizador. Nas Tabelas 5.7, 5.8 e 5.9 são apresentados os tempos de cada uma das actividades realizadas tanto no emulador J2ME integrado no NetBeans, no `MIDlet` no dispositivo móvel e no módulo `Applet` num computador. Foram feitas 10 amostragens.

Este teste só pode ser efectuado se, pelo menos, um cliente estiver a executar o serviço de navegação. O teste foi realizado em conjunto com o teste do Serviço de Navegação descrito acima.

Tabela 5.7: Tempos de utilização do Serviço de Seguimento no emulador

Actividade		Tempo Mínimo (s)	Tempo Médio (s)	Tempo Máximo (s)
1 x	Envio e Recepção mensagens operação *followUsers*	0,132	0,217	0,303
1 x	Envio e Recepção mensagens operação *followMap*	8,260	8,720	9,180
45 x	Intervalo entre cada envio das mensagens da operação *followUpdate*	30 s		
45 x	Envio e Recepção mensagens operação *followUpdate*	7,579	7,729	8,233
1 x	Envio e Recepção mensagens operação *followStop*	0,545	0,576	0,608
Total		1707,04 ≈ 28,455 min	1726,07 ≈ 28,767 min	1744,8 ≈ 29,08 min

Tabela 5.8: Tempos de utilização do Serviço de Seguimento no telemóvel

Actividade		Tempo Mínimo (s)	Tempo Médio (s)	Tempo Máximo (s)
1 x	Envio e Recepção mensagens operação *followUsers*	0,496	0,521	0,546
1 x	Envio e Recepção mensagens operação *followMap*	10,595	10,619	10,643
45 x	Intervalo entre cada envio das mensagens da operação *followUpdate*	30 s		
45 x	Envio e Recepção mensagens operação *followUpdate*	8,147	8,190	8,233
1 x	Envio e Recepção mensagens operação *followStop*	0,5	0,726	0,953
Total		1736,35 ≈ 28,93 min	1738,6 ≈ 28,976 min	1740,86 ≈ 29,01 min

Tabela 5.9: Tempos de utilização do Serviço de Seguimento no módulo `Applet`

	Actividade	Tempo Mínimo (s)	Tempo Médio (s)	Tempo Máximo (s)
1 x	Envio e Recepção mensagens operação *followUsers*	0,496	0,521	0,546
1 x	Envio e Recepção mensagens operação *followMap*	10,595	10,619	10,643
45 x	Intervalo entre cada envio das mensagens da operação *followUpdate*	30 s		
45 x	Envio e Recepção mensagens operação *followUpdate*	8,147	8,190	8,233
1 x	Envio e Recepção mensagens operação *followStop*	0,5	0,726	0,953
Total		1736,35 ≈ 28,93 min	1738,6 ≈ 28,976 min	1740,86 ≈ 29,01 min

Serviço de Meteorologia: O teste da utilização do Serviço de Meteorologia consistiu na selecção no menu principal da opção correspondente e na subsequente recepção, processamento e apresentação da informação recebida ao utilizador. Nas Tabelas 5.10, 5.11 e 5.12 são apresentados os tempos de cada uma das actividades realizadas tanto no emulador J2ME integrado no NetBeans como nos módulos cliente `MIDlet`, executado no dispositivo móvel HTC, e `Applet`, executado num computador. Foram tidas em conta 10 amostragens.

Tabela 5.10: Tempos de utilização do Serviço de Meteorologia no emulador

Actividade	Tempo Mínimo (s)	Tempo Médio (s)	Tempo Máximo (s)
Envio do Pedido e Recepção da Resposta	0,244	0,430	1,180
Processamento da Resposta	0,006	0,123	0,021
Apresentação do Ecrã	0,004	0,144	0,033
Total	0,254	0,457	1,234

Tabela 5.11: Tempos de utilização do Serviço de Meteorologia no telemóvel

Actividade	Tempo Mínimo (s)	Tempo Médio (s)	Tempo Máximo (s)
Envio do Pedido e Recepção da Resposta	0,421	0,584	0,729
Processamento da Resposta	0,418	0,544	0,731
Apresentação do Ecrã	0,425	0,583	0,733
Total	1,264	1,465	2,193

Tabela 5.12: Tempos de utilização do Serviço de Meteorologia no `Applet`

Actividade	Tempo Mínimo (s)	Tempo Médio (s)	Tempo Máximo (s)
Envio do Pedido e Recepção da Resposta	0,188	0,310	0,625
Processamento da Resposta	0,031	0,045	0,062
Apresentação do Ecrã	0,015	0,016	0,016
Total	0,234	0,371	0,703

5.1.2 Quantidade de Informação

A quantidade de informação que é trocada entre os diversos módulos do sistema é outro factor importante. Num cenário constituído por um grande número de utilizadores, a quantidade de informação consumida pela aplicação torna-se um factor determinante para o seu desempenho. Por essa razão, realizaram-se testes para determinar a quantidade de informação que o sistema gera na rede. Para este efeito, quantificou-se o tráfego de *download* e *upload* produzido pelo módulo servidor. Foram realizadas 10 amostragens em cada teste.

Serviço de Posicionamento:

Tabela 5.13: Tráfego gerado durante o Serviço de Posicionamento

	Quantidade Informação Mínima (KiB)	Quantidade Informação Média (KiB)	Quantidade Informação Máxima (KiB)
Download	45	73,8	130
Upload	85	104,5	140
Total	130	178,3	270

Serviços de Navegação e de Seguimento: Neste teste apresenta-se a quantidade de informação gerada pela utilização de ambos os serviços, *i.e.*, os resultados apresentados referem-se à utilização conjunta do Serviço de Navegação por parte de um cliente e do Serviço de Seguimento por outro cliente.

Tabela 5.14: Tráfego gerado durante o Serviços de Navegação e Seguimento

	Quantidade Informação Mínima (MiB)	Quantidade Informação Média (MiB)	Quantidade Informação Máxima (MiB)
Download	1,62	1,72	1,82
Upload	9,72	9,785	9,85
Total	11,34	11,505	11,67

Serviço de Meteorologia:

Tabela 5.15: Tráfego gerado durante o Serviço de Meteorologia

	Quantidade Informação Mínima (KiB)	Quantidade Informação Média (KiB)	Quantidade Informação Máxima (KiB)
Download	2,9	3,21	3,5
Upload	1,2	1,36	1,5
Total	4,1	4,57	5

5.2 Conclusão

Neste capítulo foram descritos os testes realizados e apresentados os resultados relativos à utilização dos serviços de Posicionamento, Navegação, Seguimento e Meteorologia oferecidos pelo sistema multi-agente.

Tendo em conta os tempos médios de resposta da utilização de cada um dos serviços assim como os valores médios da quantidade de informação gerada pelo Módulo Servidor sobre a rede, consideram-se positivos os resultados obtidos.

Capítulo 6

Conclusões

Ao longo desta Dissertação descreveram-se os vários passos do desenvolvimento deste projecto: o estudo do estado da arte, a definição da arquitectura, a escolha justificada das tecnologias de suporte e o desenvolvimento e avaliação do sistema.

Neste último capítulo retiram-se as principais conclusões e sugerem-se eventuais desenvolvimentos futuros.

6.1 Balanço

No início deste projecto foi definido o objectivo da Dissertação: conceber e criar um sistema multi-agente provedor de meta-serviços de intermediação, continuidade de serviço e personalização face a serviços pré-existentes de posicionamento, navegação, seguimento e de informação meteorológica e adaptado a múltiplos dispositivos cliente. Adicionalmente, os agentes devem possuir uma interface do tipo serviço *Web*.

O sistema realizado proporciona ao utilizador uma experiência de interacção e de utilização de diferentes tipos de dispositivos cliente transparente e amigável (interface gráfica, textual e de voz), tomando em consideração as preferências de cada utilizador e as limitações de cada dispositivo. Esta informação é armazenada numa base de dados a fim de assegurar a continuidade de serviço entre sessões consecutivas.

Desenvolveram-se também dois tipos de aplicações cliente destinadas, respectivamente, a dispositivos móveis e tradicionais.

O primeiro passo da realização deste projecto consistiu na pesquisa e estudo da informação relacionada com agentes, sistemas multi-agente e serviços *Web*.

De seguida, procedeu-se à definição da arquitectura e desenvolvimento do sistema multi-agente e das aplicações cliente. Por último, procedeu-se ao teste e validação ao sistema desenvolvido.

Adoptou-se a plataforma JADE e o uso do *add-on* WSIG para implementar o sistema multi-agente com interfaces do serviço *Web* para todos os serviços oferecidos.

De forma a permitir a validação e avaliação das funcionalidades implementadas, foram identificados os casos de uso bem como os diferentes cenários de utilização. Realizou-se, por fim, um conjunto de testes ao sistema multi-agente e às aplicações cliente que demonstraram a capacidade do sistema em prestar os seus serviços de forma bastante satisfatória.

6.2 Melhoramentos Futuros

Em relação ao sistema multi-agente pode-se propor: (*i*) a adição de novos serviços como, *e.g.*, um serviço de seguimento de múltiplos utilizadores em simultâneo ou um serviço de informação sobre locais de interesse para uma determinada localização; ou (*ii*) a obtenção por omissão da informação meteorológica do local onde se encontra o dispositivo.

Em relação às aplicações cliente seria interessante proceder ao aumento das características e refinamento das preferências dos utilizadores como, *e.g.*, a escolha do intervalo de tempo de actualização das coordenadas geográficas realizada pela aplicação, o intervalo de tempo entre o envio das mensagens de actualização dos Serviços de Navegação e Seguimento, ou a possibilidade de escolha do tipo de mapa mostrado ou até mesmo a fonte de dados utilizada (Yahoo Maps, Microsoft Maps, *etc.*).

Bibliografia

[1] Porto Editora, *Dicionário da Língua Portuguesa 7ª Edição.* Porto Editora. [citado na p. 5]

[2] RUSSEL, Stuart and NORVIG, Peter, *Artificial Intelligence: A modern Approach.* Prentice Hall, 1995. [citado na p. 5, 7, 19]

[3] MAES, Pattie and DARRELL, Trevor and BLUMBERG, Bruce and PENTLAND, Alex, *The ALIVE System: Wireless, full-body interaction with autonomous agents.* MIT Media Laboratory, 1996. [citado na p. 5]

[4] GROSOF, Benjamin and KEPHART, Jeff, *Intelligent Agents Project.* IBM Thomas J. Watson Research Center, 1998. [citado na p. 6]

[5] WOOLDRIGE, Michael and JENNINGS, Nicholas, *Intelligent Agents: Theory and Practice.* Knowledge Engineering Review, 1994. [citado na p. 6, 9, 13]

[6] NWANA, Hyacinth, *Software Agents: An Overview.* Knowledge Engineering Review, 1996. [citado na p. 7]

[7] DEUTSCH, Peter. The Eight Fallacies of Distributed Computing. [Online]. Available: http://blogs.sun.com/jag/resource/Fallacies.html [citado na p. 14, 24]

[8] O'HARE, Greg and JENNINGS, Nicholas, *Foundations of Distributed Artificial Intelligence.* Wiley, 1996. [citado na p. 14]

[9] LESSER, Victor, *Cooperative Multi-Agent Systems: A Personal View of the State of the Art.* IEEE Transactions on Knowledge and Data Engineering, 1999. [citado na p. 15]

[10] STONE, Peter and VELOSO, Manuela, *User-guided interleaving of planning and execution.* European Workshop on Planning. [citado na p. 15]

[11] WOOLDRIGE, Michael and SYCARA, Katia and JENNINGS, Nicholas, *A Roadmap of Agent Research and Development.* Journal of Autonomous Agents and Multi-Agent Systems, 1998. [citado na p. 16]

[12] M. HUHNS and L. STEPHENS, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* MIT Press, 1999. [Online]. Available: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=8273 [citado na p. 16]

[13] Telecom Italia. JADE. [Online]. Available: http://jade.tilab.com/ [citado na p. 21]

[14] Sun Microsystems. Jini. [Online]. Available: http://www.jini.org/wiki/Main_Page [citado na p. 24]

[15] HÜBNER, Jomiand and BORDINI, Rafael. JASON. [Online]. Available: http://jason.sourceforge.net/Jason/Jason.html [citado na p. 25]

[16] J. R. SEARLE, *Speech acts : an essay in the philosophy of language.* Cambridge University Press, London :, 1969. [Online]. Available: http://www.loc.gov/catdir/toc/cam025/68024484.html [citado na p. 25]

[17] HOWDEN, Nick and RÖNNQUIST, Ralph and HODGSON, Andrew and LUCAS, Andrew, *Intelligent Agents - Summary of an Agent Infrastructure.* 5th International Conference on Autonomous Agents, 2001. [citado na p. 27]

[18] AOS Group. Jack. [Online]. Available: http://www.agent-software.com.au/products/jack/ [citado na p. 27]

[19] World Wide Web Consortium. http://www.w3.org/TR/wsa-reqs/. [Online]. Available: http://www.w3.org/TR/wsdl20/ [citado na p. 29]

[20] LOPES, Carlos and RAMALHO, José, *Web Services: Metodologias de Desenvolvimento.* XATA 2004: XML: Aplicações e Tecnologias Associadas, 2004. [citado na p. 29, 32]

[21] SHKLAR, Leon and ROSEN, Rich, "Web Application Architecture: Principles, Protocols, and Practices." [citado na p. 29]

[22] World Wide Web Consortium. Web Services Description Language (WSDL) Version 2.0. [Online]. Available: http://www.w3.org/TR/wsdl20/ [citado na p. 29]

[23] World Wide Web Consortium. XML. [Online]. Available: http://www.w3.org/XML/ [citado na p. 30]

[24] ——. SOAP Version 1.2. [Online]. Available: http://www.w3.org/TR/soap [citado na p. 31]

[25] W3Schools. WSDL Tutorial. [Online]. Available: http://www.w3schools.com/wsdl/default.asp [citado na p. 32]

[26] FERNANDES, Pedro and NUNES, Urbano, *Multi-agent Architecture for Simulation of Traffic with Communications*. ICINCO 2008 - International Conference on Informatics in Control, Automation and Robotics, 2008. [citado na p. 42]

[27] F. L. BELLIFEMINE, G. CAIRE, and D. GREENWOOD, *Developing Multi-Agent Systems with JADE*. Wiley, 2007. [Online]. Available: http://onlinelibrary.wiley.com/book/10.1002/9780470058411 [citado na p. 42]

[28] WEYNS, Danny and PARUNAK, H. Van Dyke and MICHEL, Fabian and HOLVOET, Tom and FERBER, Jacques, *Environments for Multiagent Systems: State-of-the-Art and Research Challenges*. Environments in multiagent systems, 2004. [citado na p. 42]

[29] NetBeans Organization. NetBeans. [Online]. Available: http://netbeans.org/ [citado na p. 57]

[30] HAUSTEIN, Stefan. kXML. [Online]. Available: http://kxml.sourceforge.net/index.shtml [citado na p. 58]

[31] Oracle Corporation. LWUIT. [Online]. Available: https://lwuit.dev.java.net/ [citado na p. 58]

[32] Free Software Foundation, Inc. eSpeak. [Online]. Available: http://espeak.sourceforge.net/license.html [citado na p. 59]

[33] NCH Software. Switch Sound File Converter. [citado na p. 59]

[34] KNUDSEN, Jonathan. Parsing XML in J2ME. [Online]. Available: http://developers.sun.com/mobility/midp/articles/parsingxml/ [citado na p. 62]

[35] Oracle Corporation. MySQL Connector J. [Online]. Available: http://dev.mysql.com/downloads/connector/j/ [citado na p. 73]

[36] Oracle Corporation. MySQL. [Online]. Available: http://www.mysql.com/ [citado na p. 73]

Anexos

Anexo A

Mensagens do Protocolo da Aplicação

Neste anexo apresentam-se as mensagens do protocolo de nível de aplicação desenvolvido para a interface Web Service.

Devido à elevada quantidade de dados recebidos nos campos da informação gráfica nas mensagens de resposta das operações que enviam este tipo de informação é apresentado apenas um excerto da informação enviada nesses campos Esta situação ocorre nas mensagens de resposta das seguintes operações: posMap, navMap, navUpdateMap, followMap e followUpdate.

A.1 Operação login

A.1.1 Pedido

Excerto de Código A.1 Mensagem de pedido da operação *login*

```
1   <?xml version="1.0" encoding="UTF-8" ?>
2   <soapenv:Envelope
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
6   xmlns:urn="urn:ServicosMasterAgent"> <soapenv:Header/>
7       <soapenv:Body> <urn:login soapenv:
8       encodingSytle="http://schemas.xmlsoap.org/soap/encoding/">
9           <user xsi:type="xsd:string">username</user>
10          <pass xsi:type="xsd:string">password</pass>
11          <dev xsi:type="xsd:string">disp240x270</dev>
12      </urn:login>
13      </soapenv:Body>
14  </soapenv:Envelope>
```

A.1.2 Resposta

Excerto de Código A.2 Mensagem de resposta da operação *login*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <autenticarResponse xmlns="urn:ServicosMasterAgent">
            <autenticarReturn xmlns="">1,Autenticado,1111
            </autenticarReturn>
        </autenticarResponse>
        </soapenv:Body>
</soapenv:Envelope>
```

A.2 Operação addUser

A.2.1 Pedido

Excerto de Código A.3 Mensagem de pedido da operação *addUser*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosMasterAgent">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:addUser soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <user xsi:type="xsd:string">username</user>
            <newUser xsi:type="xsd:string">newuser</newUser>
            <pass xsi:type="xsd:string">password</pass>
            <nivel xsi:type="xsd:int">0</nivel>
            <servicesIntRef xsi:type="xsd:string">1001#0#0
            </servicesIntRef>
        </urn:addUser>
    </soapenv:Body>
</soapenv:Envelope>
```

A.2.2 Resposta

Excerto de Código A.4 Mensagem de resposta da operação *addUser*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <addUserResponse xmlns="urn:ServicosMasterAgent">
            <addUserReturn xmlns="">OK</addUserReturn>
        </addUserResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.3 Operação changeUser

A.3.1 Pedido

Excerto de Código A.5 Mensagem de pedido da operação *changeUser*

```
<?xml version='1.0' encoding='UTF-8' ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosMasterAgent">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:changeUser soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <cliId xsi:type="xsd:string">1</cliId>
            <newUser xsi:type="xsd:string">valter</newUser>
            <pass xsi:type="xsd:string">valter</pass>
            <level xsi:type="xsd:int">1</level>
        </urn:changeUser>
    </soapenv:Body>
</soapenv:Envelope>
```

A.3.2 Resposta

Excerto de Código A.6 Mensagem de resposta da operação *changeUser*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <changeUserResponse xmlns="urn:ServicosMasterAgent">
            <changeUserReturn xmlns="">OK</changeUserReturn>
        </changeUserResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.4 Operação pref

A.4.1 Pedido

Excerto de Código A.7 Mensagem de pedido da operação *pref*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:pref soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <cliId xsi:type="xsd:string">maria</cliId>
            <servicesIntRef xsi:type="xsd:string">1001#0#0
            </servicesIntRef>
        </urn:pref>
    </soapenv:Body>
</soapenv:Envelope>
```

A.4.2 Resposta

Excerto de Código A.8 Mensagem de resposta da operação *pref*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <prefResponse xmlns="urn:ServicosPNS1">
            <prefReturn xmlns="">1101#0#0</prefReturn>
        </prefResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.5 Operação posMap

A.5.1 Pedido

Excerto de Código A.9 Mensagem de pedido da operação *posMap*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:posMap soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <latMarker xsi:type="xsd:string">42</latMarker>
            <lngMarker xsi:type="xsd:string">-8.61</lngMarker>
            <zoom xsi:type="xsd:int">12</zoom>
        </urn:posMap>
    </soapenv:Body>
</soapenv:Envelope>
```

A.5.2 Resposta

Excerto de Código A.10 Mensagem de resposta da operação *posMap*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <posMapResponse xmlns="urn:ServicosPNS1">
            <posMapReturn xmlns="">
                <mapData xmlns="">iVBORw0KGgoAAAANSUhEUgAAAoAAAAH
                7CAMAAABrHt6OAAADAFBMVEUAAAACBAYEBAQGBgUHBwYDCAwH
                CQoJCgcKCgoLDAkLDAwQEBASEhETExITFhEYGBgaGhofHx4LG
                ygNIjIiIiEjIyMkJCQrKikrKysvLSkuLi4xMTEzMzE2NTQ9PT
                0jO3kaQGAeSm8eS3AiVH5DTjpJSUlPTkxUVFRaWVdZWVlMWG9
                SYEdVY0laaU5eblF5SUNjY0RhcFNpeltlZWVpaGVubWpteGlx
                cXF1c3B4dHF5fnd5eXl7e3tXsFK3FBNhE2XwCzUMjg3lwVRUQ
                Bm52acx5yMPxI/9rr4WL1lNgZI8Ul8U0YIVDCXUPIr2DGo2Ot
                ps4bxa8G5+1sjHTDUXXEAZncV8l5zgqdGVRXNCDDZGKB4OQRW
                EadbJc8MlDkPSXZO7Lhmx3KNux7InM/E4Ed6KJipZSiWpSONL
                IG1dxQoBoBeLFOz5xqZvUqBL0lLjUdUsoWWxRMUyJRbaEthgB
                iBbYm6JBbkMi6xtC1SeWOi/tKtKaa2DLCRA8FjHRUFoNCVOOv
                G3HJmBUOSTjUUCH/qpOQ1nPgbPpC7uyTVYqImoRlxtWXJezUR
                k4vzls2+n6NSyWm2VRKA3ryhPnqFi6SuasCA6ITlVqQO4MT7n
                N2tMOvj6AjQPXH84VoUpYd
                ...
                </mapData>
                <mapType xmlns="">new</mapType>
                <textInfo xmlns="">TG9jYWxpemHn428gQWNOdWFsOiBWaW
                xhciBkYSBWZWlnYSwgVGVycmFzIGRlIEJvdXJvLCBQb3J0dWd
                hbA==</textInfo>
            </posMapReturn>
        </posMapResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.6 Operação posText

A.6.1 Pedido

Excerto de Código A.11 Mensagem de pedido da operação *posText*

```
<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:posText soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <latMarker xsi:type="xsd:string">42</latMarker>
            <lngMarker xsi:type="xsd:string">-8.61</lngMarker>
        </urn:posText>
    </soapenv:Body>
</soapenv:Envelope>
```

A.6.2 Resposta

Excerto de Código A.12 Mensagem de resposta da operação *posText*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <posTextResponse xmlns="urn:ServicosPNS1">
            <posTextReturn xmlns="">TG9jYWxpemHn428gQWN0dWFsOiBUcmF2
            ZXNzYSBkZSBT4SAxNzksIDQ0NzUgTWFpYSwgUG9ydHVnYWw=
            </posTextReturn>
        </posTextResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.7 Operação prepareNav

A.7.1 Pedido

Excerto de Código A.13 Mensagem de pedido da operação *prepareNav*

```
<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:prepareNav soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <cliId xsi:type="xsd:string">1</cliId>
            <destination xsi:type="xsd:string">Espinho,Portugal
            </destination>
        </urn:prepareNav>
    </soapenv:Body>
</soapenv:Envelope>
```

A.7.2 Resposta

Excerto de Código A.14 Mensagem de resposta da operação *prepareNav*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <prepareNavResponse xmlns="urn:ServicosPNS1">
            <prepareNavReturn xmlns="">RXNwaW5obywgUG9ydHVnYWw=
            </prepareNavReturn>
        </prepareNavResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.8 Operação navMap

A.8.1 Pedido

Excerto de Código A.15 Mensagem de pedido da operação *navMap*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:navMap soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <user xsi:type="xsd:string">username</user>
            <origin xsi:type="xsd:string">42.0,-8.61</origin>
            <destination xsi:type="xsd:string">Espinho,Portugal
            </destination>
        </urn:navMap>
    </soapenv:Body>
</soapenv:Envelope>
```

A.8.2 Resposta

Excerto de Código A.16 Mensagem de resposta da operação *navMap*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <navMapResponse xmlns="urn:ServicosPNS1">
            <navMapReturn xmlns="">
                <mapNavData xmlns="">iVBORw0KGgoAAAANSUhEUgAAAPAAAA
                FACAIAAAANimYEAACAAElEQVR42uydD5xVc/7/y5/1b5eWtWRZI
                WsJfQmhSKJIf9AKSdqUtlIR/R9NSqY1GWoXW4zFhghDw44MpgzG
                GpeZuFxm3bjVrbm5ze742dnd+dLv9Tmve9/3cz+fc87cqUl9zb2
                Pz+M+zj333HPvPed53uf9ef9ts7j4vR/wqAlH6uLReCyCURNZUx
                kq51hTU1lfV8uBDbhNfTzKNYFQeUX50sjSQ7eW98JzbeQ92dh/R
                GrKojW5WxvzI6HcaLgitT5aI1/NEY2Gjc/GYpFA+jb6qKqpwE7w
                fPPjlmQPN7/K/+HWmQb9xJbsfouQt1JVfOQHD=
                ...
                </mapNavData>
                <routeInfo xmlns="">269</routeInfo>
                <textInfo xmlns="">MzUsMyBrbSMzNSBtaW4uI1NlZ3VpciBzd
                WRvZXN0ZSBlbSBmcmVudGUgTjEwNz40MS4yMTg0OTAwPi04LjY4O
                Tc2MDAjTmEgcm90dW5kYSwgc2VndWlyIHBlbGEgMi6qIHNh7WRhI
                GVtIGRpcmVj5+NvIGEgUG9ydG8vTWF0b3NpbmhvczQ0MS4yMTc3N
                zAwPi04LjY4OTcxMDAjVmlyYXIg4CBkaXJlaXRhIGVtIGRpcmVj5
                jIwODMwMCNTZWd1aXIgcGVsYSBz
                ...
                </textsInfo>
            </navMapReturn>
        </navMapResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.9 Operação navUpdateMap

A.9.1 Pedido

Excerto de Código A.17 Mensagem de pedido da operação *navUpdateMap*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNSvalter">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:navUpdateMap soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <routeInfo xsi:type="xsd:int">7</routeInfo>
            <latMarker xsi:type="xsd:string">42.0</latMarker>
            <lngMarker xsi:type="xsd:string">-8.61</lngMarker>
        </urn:navUpdateMap>
    </soapenv:Body>
</soapenv:Envelope>
```

A.9.2 Resposta

Excerto de Código A.18 Mensagem de resposta da operação *navUpdateMap*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <navUpdateMapResponse xmlns="urn:ServicosPNS1">
            <navUpdateMapReturn xmlns="">
                <mapNavData xmlns="">iVBORw0KGgoAAAANSUhEUgAAAPAAAA
                FACAIAAAANimYEAAAMVU1EQVR42u3ceVCUd57Hcehuumla7G45RC
                6lucEGuQ81bohOZjLrbKOpN86Mk8mYOaiJM5MYnWNjMIklIkTUiW
                cwHkAOXdGgjuYQPMFjxAPUBOQQPJBTbvreHOclzu7fKFa9X/Xl4e
                mn4Z9vfepb3+dpCjs7AAAAAAAAAAAAAAAAAAAAAAAAAE+a82gP7w
                lxPn7xbh4her0+Jjo6Ni7a29uLzuApo1Aop017xkmlNUpst9XGq+
                69l917a9VGg9Sm0agnJyeqVCq6hKdkMKvHeXgGdMgtBwMf/m3qvU
                Uzbi+cUS1q8Yzbf51y93+C2h4qLDq/CePH+9IrjHS+vr6ubh7fux
                iWPXNn0fPVC5+rWZhS+2OJlz+pemta3TW33jFjtCLWdAwjl8zBOW
                2s/00XwxspNQun90d50fTb/xLowZpes/i5mqtuvQxpjGg+uoR2uW
                ZtPrBNP9Q4VcPq3fvHAz3UMSfq3k7pb5FaaZpGKEclWqlSpsf0jY
                ovzfTKcjK46FlBaICis7JEnf7PLZnp8pz4s0/zCzxULyaVjLGK3W
                3Mc3cOIEx/iZJTYXmg45bRlu3r9jqSc/BcPHPvjN6ePVdfXdXRWd
                V/WHl9xYz7Q2kWsX6uRszyHpk1MTGe7mHECR2vuNrW6rnxk7XnLt
                3dFrt7ezkDqkpk43LFtUvfnldyhRv51Hra2/OCSz/qsjLzMrUarz
                1tR3tbT09vdwFYgSZk3pj/4KX9Bfn7cyZLZPKxRWFXP6gsTn7kz1
                ...
                </mapNavData>
                <routeInfo xmlns="">269</routeInfo>
            </navUpdateMapReturn>
        </navUpdateMapResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.10 Operação navText

A.10.1 Pedido

Excerto de Código A.19 Mensagem de pedido da operação *navText*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:navText soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <origin xsi:type="xsd:string">42.0,-8.61</origin>
            <destination xsi:type="xsd:string">Espinho,Portugal
            </destination>
        </urn:navText>
    </soapenv:Body>
</soapenv:Envelope>
```

A.10.2 Resposta

Excerto de Código A.20 Mensagem de resposta da operação *navText*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <navTextResponse xmlns="urn:ServicosPNS1">
            <navTextReturn xmlns="">
                <routeInfo xmlns="">461</routeInfo>
                <textInfo xmlns="">MTQsNSBrbSMyMyBtaW4uI1NlZ3VpciBub
                3JOZSBlbSBmcmVudGUgUnVhIGRhIElncmVqYS9DTTEOODkgZW0gZ
                GlyZWPn428gYSBUcmF2ZXNzYSBkZSBNYW51ZWwgU29hcmVzIExla
                XQ+NDEuNDcwNzgwMD4tOC4ONDA2OTAwIONvbnRpbnVlIGFO6SBOM
                zA5PjQxLjQ4NTA1MDA+LTguNDI1NjYwMCNWaXJhciDgIGVzcXVlc
                mRhIHBhcmEgY29udGludWFyIG5hIE4zMD4OMS41MTQzMDAwPi04L
                jQzNDIzMDAjVmlyYXIg4CBkaXJlaXRhIG5hIFJ1YSBkbyBBbOltI
                GRvIFJ1YW8+NDEuNTE2NDgwMD4tOC4OMjkxMTAwI1NlZ3VpciBwZ
                WxhIHJhbXBhIHBhcmEgQnJhZ2E+NDEuNTE2NTIwMD4tOC4OMjg5M
                DAwI1Blcm1hbmVjZXIg4CBkaXJlaXRhIG5hIGJpZnVyY2Hn428+N
                DEuNTE3NjUwMD4tOC4OMjY3MTAwIONvbnRpbnVhciBlbSBmcmVud
                GU+NDEuNTIzNjAwMD4tOC4OMTc2NzAwIONvbnRpbnVhciBlbSBmc
                mVudGUgcGFyYSBBdmVuaWRhIE1pZ3VlbCBUb3JnYT4OMS41MzU3O
                DAwPiO4LjQwOTU4MDAjQ29udGludWUgYXTpIEF2ZW5pZGEgRG91d
                G9yIEZyYW5jaXNjbyBTYWxnYWRvIFplbmhhPjQxLjUONDkOMDA+L
                28gWFhJPjQxLjUOOTI4MDA+LTguNDE1NjYwMCNTZWd1aXIgcGVsY
                SAxqiDgIGRpcmVpdGEgZW0gZGlyZWPn428gYSBBdmVuaWRhIDMxI
                DA+LTguNDIyODgwMCM=</textInfo>
            </navTextReturn>
        </navTextResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.11 Operação navUpdateText

A.11.1 Pedido

Excerto de Código A.21 Mensagem de pedido da operação *navUpdateText*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:navUpdateText soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <routeInfo xsi:type="xsd:int">7</routeInfo>
            <latMarker xsi:type="xsd:string">42.0</latMarker>
            <lngMarker xsi:type="xsd:string">-8.61</lngMarker>
        </urn:navUpdateText>
    </soapenv:Body>
</soapenv:Envelope>
```

A.11.2 Resposta

Excerto de Código A.22 Mensagem de resposta da operação *navUpdateText*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <navUpdateTextResponse xmlns="urn:ServicosPNS1">
            <navUpdateTextReturn xmlns="">
                <routeInfo xmlns="">464</routeInfo>
                <textInfo xmlns="">TG9jYWxpemHn428gQWN0dWFsOiBSdWEgZ
                GUgQW1hZGV1IENvc3RhIDE3LCAONDc1IE1haWEsIFBvcnR1Z2Fs
                </textInfo>
            </navUpdateTextReturn>
        </navUpdateTextResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.12 Operação navStop

A.12.1 Pedido

Excerto de Código A.23 Mensagem de pedido da operação *navStop*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:navStop soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <cliId xsi:type="xsd:string">1</cliId>
            <routeInfo xsi:type="xsd:int">90</routeInfo>
        </urn:navStop>
    </soapenv:Body>
</soapenv:Envelope>
```

A.12.2 Resposta

Excerto de Código A.24 Mensagem de resposta da operação *navStop*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <navStopResponse xmlns="urn:ServicosPNS1">
            <navStopReturn xmlns="">OK</navStopReturn>
        </navStopResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.13 Operação followUsers

A.13.1 Pedido

Excerto de Código A.25 Mensagem de pedido da operação *followUsers*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNSvalter">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:followUsers soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <user xsi:type="xsd:string">username2</user>
        </urn:followUsers>
    </soapenv:Body>
</soapenv:Envelope>
```

A.13.2 Resposta

Excerto de Código A.26 Mensagem de resposta da operação *followUsers*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <followUsersResponse xmlns="urn:ServicosPNS1">
            <followUsersReturn xmlns="">fernando</followUsersReturn>
        </followUsersResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.14 Operação followMap

A.14.1 Pedido

Excerto de Código A.27 Mensagem de pedido da operação *followMap*


```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNSf1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:followMap soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <cliIdFol xsi:type="xsd:string">2</cliIdFol>
            <user xsi:type="xsd:string">fernando</user>
        </urn:followMap>
        </soapenv:Body>
</soapenv:Envelope>
```


A.14.2 Resposta

Excerto de Código A.28 Mensagem de resposta da operação *followMap*


```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <followMapResponse xmlns="urn:ServicosPNS1">
            <followMapReturn xmlns="">
                <mapNavData xmlns="">iVBORw0KGgoAAAANSUhEUgAAAoAAAAH7
                AADTornrAACAAElEQVR42uy9D5hbZZ33za7yR6FaQFbqwiMo1WWx8
                8h2+yCLyiroumtdeAFBaQVKKW1poS3DUGodSmmlSJUqo1sddGSQIE
                YymIEZmDOQQAhpIkeIEJnROM/mep4r73XNYt/v7/7e55c7J5nM9A/
                JdV+5Tk7OOTk5mTmf+/v7e9D2qNcczfH/6xEdSJfGRzHyfjKfHeEy
                UBzLY2CBL/NZD9tgjW6T9ke89MD0RzLRwyHL/ZtHu05MtJ++5AvH7
                /OS2e2rTy4PLRwb9f30AAY/uvYjsCaZsCMajfb19XG5fdu2SCSCl9
                d63WZiYiKdTre0t0xyHh0dHXjG+mXLli1cuBB74SX2wr54iWduxpc
                kGhwE22Pj0I7l9LZyz+npjpPTA6tLxQ1+sgsnP5pPZ4dadk1snChv
                FvwhbJDtlC399iPx9THyHb0wLCujl6aH2rFvNj1Q8JPu9efANSmkt
                sxyaWN6YCt+EaxMJ3ow/OxQPt2T6FmpBy9F55R65mIUI7PNZ52Q7z
                +Zaz8203FKpnO+H12Qjy0fy0RkFHJ+Jp7PDfM6lEolXLfiqK/Xquj
                XYZR7z8C+GNiXw0908SDlUhEDL7EyFV+PNfJudAGP6R7EHROx+kwa
                wxoA4x572DZfORe0JF357tCxxq7C6rcPVr1dXNcBOFt78UMVfVVT7
                XV3AKY8LBK3kgA4NB7TitgDePZW+1Zakb0FFrZ2oI+3edsK8v2uWo
                wjsEDYDBz6uLTNsAmTxgfO0EuDp1nehidhzYPzLxeAVVNNtf9bABz
                8LOrvSwKTABKeMTNXgYa/vKxTamsjtyykrgzAQUlWmrSx4JZE
                ...
                </mapNavData>
                <routeInfo xmlns="">465</routeInfo>
                <textInfo xmlns="">T3JpZ2VtOiAjTG9jYWxpemHn428gQWN0dW
                BdXRvLUVzdHJhZGEgUG9ydG8tQ2FtaW5oYSwgNDQ1MCBNYXRvc2lu
                CBQb3J0dWdhbCNEZXN0aW5vOiBFc3BpbmhvLCBQb3J0dWdhbA==
                </textInfo>
            </followMapReturn>
        </followMapResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.15 Operação followUpdate

A.15.1 Pedido

Excerto de Código A.29 Mensagem de pedido da operação *followUpdate*

```
<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS2">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:followUpdate soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <routeInfo xsi:type="xsd:int">25</routeInfo>
        </urn:followUpdate>
    </soapenv:Body>
</soapenv:Envelope>
```

A.15.2 Resposta

Excerto de Código A.30 Mensagem de resposta da operação *followUpdate*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
    <followUpdateResponse xmlns="urn:ServicosPNS1">
        <followUpdateReturn xmlns="">
            <mapNavData xmlns="">iVBORw0KGgoAAAANSUhEUgAAAoAAAAH7CAIAA
            ADTornrAACAAElEQVR42uydD3wU5Z3/jSa66qJTd1mXtTt3O+GSNvtr9tf
            ES3q33sVeNEdT6Zk7OP8UPJDTnEctcugd9rhWqlWwIkWLQhEQU0UOEVOKi
            AgWKSpUQKigUEGx4n+xgiZtWcnv8zyfmW+ezG5CQNTc/eZ5Pa99zc7OzM5
            OYN7z+f495phgBCMYn/W478njprXuG3np5ceX1i75tbVyw86X3z/me9+77
            rWO+P3Ln3lu6/bx46+dPXtOW1v71VdfI3PTpmfxivXXHHXfcqaeeumzZI3j
            7i1+s3rHjN3iLV27Gt2+99TbX4CDYHhv7dpww+6UJj2VnP/v12YsPTLzxv
            en3rLtx8m03T5szc2H71ePfHj/hbSzg7ZTpD2CDmc+oLac/P+rmdf+A0e3
            Zb2JZrVwdmb3weew7c/ZDU6a33HDjrTgIJhY4p2P97N9Ov08dc8LEt2cv/
            u2Omfdj5eyWxzCnz3xg2uzHWlZ0yMEnrv7biY/VY9649u/0dw2Z9uyw6c/
            /84znh8169muznvnz6avtaUuPnTxrrZpTZk6f1TptxgJeh4kTv4/rduPNd
            8i1unH68qvv0AVzwvK/wL6Y2Jdzess6HmTCxJsw8RYr5z70PtaoT1fbPKZ
            5EHO0X/rnE1d+FacnE+eZ6PfMR+/c8chDN215+odY4PzlI51vd2y6FZ/KR
            +/sug1v8SprcIQp0+6eeMMPMfHtLS3349unz/8DzhxXAF9x49qv4bdc3fp
            lc+JM8OmM+b+eMXcpLgheZ7Usx2XBwrTpP8VfQR3wxh9iYfKUO/DXwY+aN
            F+fVwA/78UMMW/OSmcWvG3LJRLgLexhrXcCEzTF8caxJ+qZxGdxP/Ap3en
            VssOO/xZjMgCMLf8Ltf24LtduLhsAAAAASUVORK5CYII
            ...
            </mapNavData>
            <routeInfo xmlns="">465</routeInfo>
            <textInfo xmlns="">TG9jYWxpemHn428gQWN0dWFsOiBBdXRvLUVzdHJ
            hZGEgUG9ydG8tQ2FtaW5oYSwgNDQ1MCBNYXRvc2luaG9zLCBQb3J0dWdhb
            A==</textInfo>
        </followUpdateReturn>
    </followUpdateResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.16 Operação followText

A.16.1 Pedido

Excerto de Código A.31 Mensagem de pedido da operação *followText*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:followText soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <cliIdFol xsi:type="xsd:int">1</cliIdFol>
            <user xsi:type="xsd:string">fernando</user>
        </urn:followText>
    </soapenv:Body>
</soapenv:Envelope>
```

A.16.2 Resposta

Excerto de Código A.32 Mensagem de resposta da operação *followText*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
    <followTextResponse xmlns="urn:ServicosPNS1">
        <followTextReturn xmlns="">
            <routeInfo xmlns="">465</routeInfo>
            <textInfo xmlns="">T3JpZ2VtOiBBdXRvLUVzdHJhZGEgUG9ydG8tQ
            2FtaW5oYSwgNDQ1NSBNYXRvc2luaG9zLCBQb3J0dWdhbCNMb2NhbGl6Y
            efjbyBBY3R1YWw6IEF1dG8tRXNOcmFkYSBQb3JOby1DYW1pbmhhLCAON
            DU1IE1hdG9zaW5ob3MsIFBvcnR1Z2FsIORlc3Rpbm86IEVzcGluaG8sI
            FBvcnR1Z2Fs</textInfo>
        </followTextReturn>
    </followTextResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.17 Operação followUpdateText

A.17.1 Pedido

Excerto de Código A.33 Mensagem de pedido da operação *followUpdateText*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:followUpdateText soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <routeInfo xsi:type="xsd:int">25</routeInfo>
        </urn:followUpdateText>
    </soapenv:Body>
</soapenv:Envelope>
```

A.17.2 Resposta

Excerto de Código A.34 Mensagem de resposta da operação *followUpdateText*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
    <followUpdateTextResponse xmlns="urn:ServicosPNS1">
        <followUpdateTextReturn xmlns="">
            <routeInfo xmlns="">465</routeInfo>
            <textInfo xmlns="">TG9jYWxpemHn428gQWNOdWFsOiBBdXRvLUVz
            dHJhZGEgUG9ydG8tQ2FtaW5oYSwgNDQ1MCBNYXRvc2luaG9zLCBQb3J
            0dWdhbA==</textInfo>
        </followUpdateTextReturn>
    </followUpdateTextResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.18 Operação followStop

A.18.1 Pedido

Excerto de Código A.35 Mensagem de pedido da operação *followStop*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:followStop soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <routeInfo xsi:type="xsd:int">90</routeInfo>
            <cliIdFol xsi:type="xsd:string">1</cliIdFol>
        </urn:followStop>
    </soapenv:Body>
</soapenv:Envelope>
```

A.18.2 Resposta

Excerto de Código A.36 Mensagem de resposta da operação *followStop*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <followStopResponse xmlns="urn:ServicosPNS1">
            <followStopReturn xmlns="">OK</followStopReturn>
        </followStopResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.19 Operação tts

A.19.1 Pedido

Excerto de Código A.37 Mensagem de pedido da operação *tts*

```
<?xml version="1.0" encoding="UTF-8" ?><soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:tts soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <type xsi:type="xsd:string">pos</type>
            <text xsi:type="xsd:string">TG9jYWxpemHn428gQWN0dWFsOiB
            BdmVuaWRhIGRvIEFlcm9wb3J0byAzMjIsIDQ0NzAgTWFpYSwgUG9ydH
            VnYWw=</text>
        </urn:tts>
    </soapenv:Body>
</soapenv:Envelope>
```

A.19.2 Resposta

Excerto de Código A.38 Mensagem de resposta da operação *tts*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <ttsResponse xmlns="urn:ServicosPNS1">
            <ttsReturn xmlns="">OK#pos</ttsReturn>
        </ttsResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.20 Operação meteo

A.20.1 Pedido

Excerto de Código A.39 Mensagem de pedido da operação *meteo*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:meteo soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <destination xsi:type="xsd:string">Espinho,Portugal
            </destination>
        </urn:meteo>
    </soapenv:Body>
</soapenv:Envelope>
```

A.20.2 Resposta

Excerto de Código A.40 Mensagem de resposta da operação *meteo*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <meteoResponse xmlns="urn:ServicosPNS1">
            <meteoReturn xmlns="">UG9ydG8sUG9ydHVnYWwjTGltcG8+MTg+S
            HVtaWRhZGU6IDg4JT5WZW50bzogTiBhIDIga2OvaCNxdWk+MTc+MjM+
            UGFyY2lhbG1lbnRlIG51YmxhZG8jc2V4PjE2PjIOPlBhcmNpYWxtZW5
            OZSBudWJsYWRvI3PhYj4xNz4yNT5MaW1wbyNkb2O+MTc+Mjc+Q+l1IG
            xpbXBv</meteoReturn>
        </meteoResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

A.21 Operação logout

A.21.1 Pedido

Excerto de Código A.41 Mensagem de pedido da operação *logout*

```
<?xml version="1.0" encoding="UTF-8" ?> <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ServicosPNS1">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:logout soapenv:
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <cliId xsi:type="xsd:int">1</cliId>
        </urn:logout>
    </soapenv:Body>
</soapenv:Envelope>
```

A.21.2 Resposta

Excerto de Código A.42 Mensagem de resposta da operação *logout*

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <logoutResponse xmlns="urn:ServicosPNS1">
            <logoutReturn xmlns="">OK</logoutReturn>
        </logoutResponse>
    </soapenv:Body>
</soapenv:Envelope>
```