UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Comunicações Pessoais Unificadas

João Pedro Guerreiro da Graça Patriarca
(Licenciado)

Dissertação para obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores

Orientador: Doutor Alberto Manuel Rodrigues da Silva

Presidente: Doutor Paulo Jorge Pires Ferreira

Vogais: Doutor Luís Filipe Lourenço Bernardo
Doutor Paulo Rogério Barreiros D'Almeida Pereira
Doutor Alberto Manuel Rodrigues da Silva
Doutor Rui Fuentecilla Maia Ferreira Neves

Maio 2005

Resumo

Na sequência dos avanços na área das tecnologias de informação e comunicação, das grandes expectativas que foram criadas no período especulativo das *"dot coms"*, e consequentemente dos enormes investimentos financeiros que têm sido realizados pelas operadoras de telecomunicações, existe actualmente uma necessidade imperiosa de se criarem novos serviços que sustentem e rentabilizem os investimentos realizados.

É neste âmbito que se enquadra o trabalho descrito nesta tese de mestrado. Este trabalho demonstra a capacidade de concepção e implementação de novos serviços capazes de satisfazer as necessidades actuais e futuras do mercado.

Esta tese foi uma consequência do sistema desenvolvido no âmbito do contrato de colaboração entre a empresa PT Inovação e o Instituto de Investigação e Desenvolvimento Inesc-ID. A implementação do sistema, nomeadamente a componente de comunicação, decorreu durante o ano 2002 com conclusão no ano 2003. O sistema englobou no total cinco grupos de trabalho que decorreram em alturas diferentes.

Este trabalho focou-se na concepção e no desenvolvimento de serviços de comunicação denominado por *"myComs"*. Existiram outros grupos de trabalho, nomeadamente na área dos serviços de gestão de informação pessoal (agenda e contactos). No contexto de serviços de comunicação foram desenvolvidos os serviços de correio electrónico e de mensagens instantâneas e notificações de presença. Em paralelo, foi testada e utilizada uma plataforma de aplicações (*"JBoss"*) que garante serviços de sistema (gestão do ciclo de vida de componentes, segurança, transacções, ...) e que permite também uma rápida e fácil integração de novos serviços.

Este trabalho, representando uma plataforma de serviços de comunicação e de serviços de gestão de informação pessoal, visa contribuir no despertar para a concepção de novos e futuristas serviços, que com certeza aliciarão as operadoras de telecomunicações que são o grande alvo deste sistema.

Abstract

In the sequence of the progress in the area of the information and communication technologies, of the great expectations created in the speculative period of the *"dot coms"* and consequently of the extremely high financial investments which have been made by the telecommunication operators, there exists in the present an imperious need to create new services that may support and make profitable the investments meanwhile effected.

This is the scope of the work described in this master thesis. This work demonstrates the capacity of conception and implementation of new services being able to satisfy the actual and the future market needs.

This thesis was a consequence of the system developed in the scope of the collaboration agreement between PT Inovação and the Investigation and Development Center Inesc-ID. The implementation of the system, namely the communication component, ran during the year 2002 and had its conclusion in 2003. The system comprehended, in the whole, five working groups and occurred in different times.

This work had its focus in the conception and development of communication services denominated *"myComs"*. There were other working groups, namely in the area of management services of personal information (agenda and contacts). In the scope of communication services there have been developed the service of electronic mail and the service of instant messaging and presence notifications. At the same time an application platform (*"JBoss"*) was tested and used which provides system services (management of the life cycle of the components, security, transactions, ...) and which provides also a fast and easy integration of new services.

This work, representing a platform of communication and personal information management services, aims to contribute to the conception of new and future services which certainly will seduce the telecommunication operators who are the great scope of this system.

Agradecimentos

Ao Professor Doutor Alberto Silva, orientador deste trabalho, pela proposta deste trabalho e pela disponibilidade constante e excelente orientação.

Ao Engº Paulo Chainho e demais colegas pela excelente experiência vivida na PT Inovação enquanto realização e implementação deste sistema.

Ao colectivo do Centro de Cálculo do Instituto Superior de Engenharia de Lisboa, pelas facilidades temporais concedidas para a realização do mestrado, pelos comentários e sugestões realizados.

Aos meus pais, João Gilberto Patriarca e Maria Teresa Patriarca, pela constante preocupação e acompanhamento da evolução do mestrado. Ficará na memória a célebre pergunta: *"Então? É este ano?"*

Aos meus amigos, por serem meus amigos, pelo acompanhamento nos bons e maus momentos.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Capítulo 1

Introdução

Nos dias actuais a informação é global e de fácil acesso a todos. Pelo menos isto é verdade, e sem querer fazer qualquer associação política, se estivermos a pensar num contexto de civilizações abertas e democráticas.

Os humanos habituaram-se à facilidade de obter informação, e nem imaginam o mundo de outra forma. É um facto desde cedo adquirido.

Os humanos são por natureza exigentes com o que os rodeia. Comodidade, bem-estar, lazer, entretenimento é algo que as pessoas exigem também cada vez mais. De uma forma geral, coisas que são consideradas supérfluas hoje passarão a ser imprescindíveis amanhã.

Conhecimento, cultura, informação são temas também imprescindíveis ao bem estar de uma pessoa. Respostas eficientes, rápidas, naturais a problemas é algo que também proporciona bem estar a uma pessoa. Principalmente porque liberta-a de preocupações desnecessárias.

A evolução da tecnologia tem alimentado esta ambição dos humanos. Verifica-se que esta tem sofrido evoluções galopantes nas últimas décadas, tendo nos últimos anos atingido níveis estonteantes. Digamos que, o fenómeno tecnológico que está a acontecer hoje, por exemplo o surgimento da Internet, representa o mesmo fenómeno que aconteceu por exemplo com o surgimento e difusão da televisão.

Mais! A generalização da oferta permitiu o acesso a equipamentos e serviços por uma população cada vez mais global. Hoje em dia, qualquer pessoa tem acesso a tecnologia de topo com uma naturalidade assumida. O próprio mercado isso proporcionou. Portanto, para satisfazer e manter alimentado o sistema, é necessário criar, dinamizar, diversificar novos serviços. Por outro lado, e dada a diversidade de oferta, para ganhar o mercado, obrigatoriamente os serviços têm que surpreender pela positiva, têm que ser originais.

É indiscutível que um dos ramos associado às novas tecnologias é a área das telecomunicações. Nesta área temos vindo a observar uma oferta crescente de serviços. Em tempos

idos a transmissão de voz era o serviço base, praticamente único; hoje em dia, esse serviço é apenas um entre muitos outros, onde todos têm o mesmo grau de importância, senão maior. Hoje observamos uma evolução da infra–estrutura a nível de redes - por exemplo, redes móveis: GSM900, GSM1800, GSM1900, GPRS, UMTS - que vem permitir uma oferta crescente de novos serviços, e mesmo de serviços existentes com melhor qualidade.

Sem dúvida que, dada a diversidade de oferta de operadoras e de serviços, a cativação de clientes consegue-se à custa de preços apelativos, imposição no mercado, e disponibilização de mais serviços. Para esta última característica, uma vez que a oferta é tão elevada, a imaginação é imprescindível.

Dos diferentes tipos de serviços que hoje são disponibilizados aos consumidores há uns cujas características comuns são a ubiquidade e baseados em contexto. Para além de ser um segmento onde pode e deve haver uma forte intervenção são também estes tipos de serviços que interessam no contexto desta tese de mestrado.

Ubiquidade significa (extraído directamente do dicionário) [16]:

> s. f.
> dom de estar ao mesmo tempo em vários lugares;
> omnipresença.

Admitindo que o "estar" significa estar conectado a uma rede comum, portanto comunicável, então uma pessoa passa a ter o dom de "estar" em qualquer altura, em qualquer lugar.

Por sua vez, o uso de contexto é importante na utilização de aplicações interactivas, principalmente em situações em que o contexto do utilizador está a mudar rapidamente, como seja a computação móvel e ubíqua. É essencial perceber-se o uso de contexto para facilitar a construção de aplicações baseadas em contexto.

O texto que se segue é baseado essencialmente num documento que descreve de uma forma concisa a importância de **contexto**, assim como a sua definição e a sua categorização [13]:

"Entre humanos um diálogo pode ser influenciado por diversos factores: o conhecimento comum de como o mundo funciona, a compreensão implícita de situações diárias, no fundo a utilização de contexto. É este conceito que é importante conseguir traduzir para um diálogo entre um humano e um computador. Quanto mais uma aplicação recorrer a contexto maior riqueza poderá proporcionar na interacção com o humano.

Para se retirar maior partido do uso de contexto é necessário perceber o que é contexto e como o usar, nomeadamente perceber em que pontos é que uma aplicação poderá e

deverá ser influenciada por contexto, ou como é que o poderá utilizar. Outro ponto-chave é perceber como é que se poderá dar a conhecer contexto a uma aplicação. Uma forma seria o humano fornecer essa informação à aplicação; no entanto, parece razoável não ser esta a forma mais elegante de o fazer. Quanto mais automatizado for esse processo tanto melhor, não apenas para o humano, mas também para a própria aplicação uma vez que pode ser ela a seleccionar a informação que considerar mais relevante abstraindo o que o humano consideraria relevante como informação para a aplicação. O problema que se põe é conseguir caracterizar contexto de forma a sistematizar a informação a ser processada pela aplicação."

Como referido é importante perceber então o que é contexto para conseguir sistematizar e importar estes conceitos para uma aplicação. De uma forma geral, o entendimento do que é contexto é relativamente simples; a maior dificuldade consiste em caracterizá-lo, consiste em defini-lo. Nesse sentido existem já uma série de definições, até porque não se trata de um assunto propriamente recente. Foram investigados documentos datados desde 1994 [79]. No entanto, de todos houve um documento [13] que de si era uma análise de outros documentos e que apresentou a definição mais concreta, compreensível a qual passo a citar:

> *"**Context** is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves."*

Todas as definições encontradas tendem, para facilitar a definição de contexto, atribuir categorias ao contexto e que de uma forma geral é de comum acordo que há determinadas tipos que são mais relevantes que outros, nomeadamente **localização**, **identidade**, **actividade** e **tempo**. Estes tipos de contexto respondem às perguntas, que de uma forma geral, as aplicações baseadas em contexto andam à procura (*Where*, *Who*, *What*, *How*, *When*). Para além de responderem a estas perguntas podem também indiciar outras fontes de informação de contexto. Por exemplo, através da identidade de uma pessoa pode-se saber a sua morada, telefone, entre outras informações.

Estas caracterizações ajudam aos desenhadores de uma aplicação a escolher o tipo de contexto a usar nas suas aplicações, a estruturá-lo, inclusive a procurar outros mais relevantes. Portanto estes quatro tipos principais de contexto indicam a informação necessária para caracterizar uma situação assim como indiciam como a usar e organizar. Uma vez verificado o que é contexto e como é que se pode defini-lo, é importante perceber como é que se pode usá-lo, nomeadamente o que é que caracteriza uma aplicação, quais poderão ser as suas funcionalidades acrescidas para se poder dizer que é baseada em contexto. Tal como contexto, existem muitas definições para ***context-awareness***. Muitas dessas definições

encontram-se resumidas no documento [13] sendo mais uma vez eleita a definição que a seguir se cita:

> *"A system is* ***context-aware*** *if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."*

Para que uma aplicação seja considerada como uma aplicação baseada em contexto deverá suportar um conjunto de funcionalidades e ter um conjunto de características reconhecidas como funcionalidades baseadas em contexto [76, 13]. Nomeadamente essas funcionalidades estão divididas nas seguintes categorias:

- *apresentação* de informação e de serviços ao utilizador;
- *execução* automática de um serviço;
- *marcação* de contexto como informação para consulta posterior.

1.1 Motivação

Actualmente existe uma forte necessidade de criar novos serviços na área das tecnologias de informação e comunicação. Esta necessidade existe porque foram criadas grandes expectativas no período especulativo das *"dot coms"*, e daí resultaram grandes investimentos financeiros realizados pelas operadoras de telecomunicações.

As operadoras de telecomunicações têm vindo progressivamente, e em todo o mundo, a substituir as suas redes antigas (analógicas e ou digitais, orientadas fundamentalmente para serviços de voz baseados na telefonia) por redes de nova geração, onde se estima que os serviços de dados e multimédia ultrapassem, em termos de tráfego da rede, os tradicionais serviços de voz. Estas redes integram e disponibilizam um conjunto de características técnicas extremamente avançadas, integrando serviços de comunicação fixos com móvel, voz com dados simples (e.g., SMS) ou multimédia (e.g., MMS), e ainda providenciando acesso a outras redes externas (e.g., Internet, iTV).

Para sustentar os enormes investimentos realizados pelas operadoras de telecomunicações é necessário a criação urgente de novos serviços por forma a manter e angariar novos clientes. É nesse sentido que surge este trabalho. Mostrar a capacidade de concepção e realização de serviços inovadores que respondam a necessidades actuais e futuras.

1.2 Enquadramento

O "PUC - Sistema de Comunicações Pessoais para as Redes de Próxima Geração" é um projecto de I&D, resultado de uma colaboração entre o INESC-ID e a PT-Inovação, tendo

como principal objectivo a concepção de um sistema constituído por múltiplos serviços pessoais que tirem partido do estado emergente das tecnologias telemáticas e das redes de telecomunicações de próxima geração.

O sistema PUC pretende oferecer aos seus subscritores um conjunto alargado de serviços pessoais, de entre os quais se destacam (1) o serviço de comunicações pessoais (*"myComs"*); (2) o serviço de contactos (*"myContacts"*); e (3) o serviço de agenda e de compromissos (*"myAgenda"*). Estes serviços por si só não são factor de inovação. O principal factor de inovação do projecto PUC reside no facto destes serviços serem desenvolvidos e subscritos de forma dinâmica e integrada, serem suportados pelas redes de próxima geração, serem acedidos através de diferentes tipos de terminais (e.g., telefone fixo ou móvel, PC ou PDA) e consequentemente oferecerem diferentes tipos de interacção homem–máquina (e.g., via voz, *Web*, WAP).

Dado o conjunto de serviços que se pretende implementar, o nome de PUC poderá estar algo desajustado, ou pelo menos incompleto, uma vez que alguns dos serviços reflectem não uma gestão de comunicações mas sim uma gestão de informação pessoal. Neste sentido, este projecto é ainda mais ambicioso, pois pretende integrar num único sistema funcionalidades PUC[1] com funcionalidades PIM[2].

No âmbito da colaboração formal estabelecida em Dezembro de 2001, entre o INESC–ID e a PT–Inovação, e com a duração prevista de 12 meses, o projecto focou-se na concepção, desenvolvimento e avaliação do protótipo do serviço *"myComs"*, bem como na exploração de tecnologias emergentes.

O projecto realizado no âmbito deste protocolo foi coordenado por:

- Prof. Alberto Silva (do INESC–ID);
- Prof. Paulo Ferreira (do INESC–ID), no que se refere à componente de análise e avaliação de requisitos não funcionais; e
- Eng. Paulo Chainho, como coordenador do projecto por parte da PT–Inovação.

Por outro lado, o projecto forneceu o contexto e a oportunidade para a realização de dois trabalhos de engenharia e de investigação, designadamente participando:

- Eng. João Pedro Patriarca, que, no âmbito da sua tese de mestrado de engenharia electrotécnica e computadores (MEEC), trabalhou na concepção e desenvolvimento do serviço *"myComs"*; e

[1] *Personal Unified Communication*
[2] *Personal Information Management*

- Eng. João Clemente, que, no âmbito do seu trabalho final de curso (TFC), trabalhou na análise e avaliação de requisitos não funcionais e de arquitecturas tecnológicas de suporte ao sistema[8].

Este projecto conseguiu ainda proporcionar assunto para mais três TFC (Trabalhos Finais de Curso) da LEIC (Licenciatura em Engenharia Informática e Computadores) do IST (Instituto Superior Técnico). Dois deles estão enquadrados no designado PUC2, portanto uma segunda iteração sobre o trabalho realizado no PUC. Estes dois projectos decorreram durante o ano de 2003. O terceiro grupo de trabalho está enquadrado no PUC3 e encontra-se actualmente em desenvolvimento. O PUC1, PUC2 e PUC3 revelam uma sequência temporal, que existiu enquanto trabalho realizado (implementação) e que deixou de existir enquanto trabalho publicado, uma vez que esta tese está a ser escrita, e que corresponde a trabalho realizado no âmbito do PUC1, num período em que já vários TFC foram concluídos, nomeadamente TFC realizados no âmbito do PUC2.

Em relação ao PUC2 foram realizados dois TFC:

- "Personal Unified Communication" – Sistemas Embebidos para o Sistema PUC [15].
- "Personal Unified Communication" – Assistente Pessoal de Agenda e de Contactos [35].

Sistema Embebidos para o Sistema PUC. Consiste na implementação de interfaces homem–máquina para serviços baseados em equipamentos móveis (PDAs, telemóveis) e também para serviços baseados em voz. A aplicação protótipo pretende demonstrar as capacidades do conceito PUC numa infra–estrutura de redes 3G/NGN. Os serviços propostos dividem-se em duas partes: assistente pessoal de comunicações (*"myComs"*); assistente pessoal de agenda e contactos (*"myAgenda"* e *"myContacts"*). A nível tecnológico recorre-se à utilização do J2ME para interfaces móveis e a *VoiceXML* para interface voz.

Este texto foi transcrito do documento [15]. Este trabalho final de curso foi realizado pelos alunos Nuno Domingos e Nuno César.

Assistente Pessoal de Agenda e de Contactos. O objectivo deste projecto de TFC é a concepção, desenvolvimento e avaliação de serviços de demonstração do conceito *Personal Unified Communication* (PUC). Foi desenvolvida uma aplicação protótipo que visa incluir as funcionalidades típicas de um Assistente Pessoal de Comunicações (*WebMail* e *Instant Messaging*) e de um Assistente Pessoal de Agenda e Calendário [35].

Este texto foi transcrito do documento [35]. Este trabalho final de curso foi realizado pelos alunos Marco Guimarães e Rui Maia.

O PUC3, como já referido, encontra-se ainda em desenvolvimento e consiste numa integração dos vários serviços desenvolvidos até então. Tendo sido iniciado em 2003 prevê-se que esteja concluído durante o ano 2004. Pretende com esta integração proporcionar aos serviços uma interface comum e consistente tanto a nível de componentes como a nível de interface do utilizador. Está a ser realizado pelos alunos David Martins e Eurico Frade. Como a escrita desta tese demora mais do que o espectável também esta terceira fase do projecto foi concluída em Julho de 2004.

Este projecto ainda proporcionou a realização de um artigo científico apresentado e publicado no III Congresso Iberoamericano de Telemática que aconteceu no Uruguai no ano 2003 [11].

1.3 Objectivos da tese

A próxima lista reúne tópicos de objectivos expectáveis no desenvolvimento deste trabalho.

- criação de uma plataforma base de serviços que sirva como base de trabalho para outros trabalhos;
- integração do conceito PUC com o conceito PIM;
- exploração de novos serviços (plataforma de serviços) e sua viabilidade sobre novas tecnologias;
- como carácter tecnológico a abstracção das tecnologias de rede, ubiquidade de tipo de acesso e de tipo de media, elevado número de utilizadores e de transacções, gestão e subscrição flexível e dinâmica de serviços, tecnologia Java e *Open–Source*. Estes objectivos voltam a ser referidos no capítulo 3, na secção 3.2;
- validação da utilização das novas tecnologias em problemas reais, concretos;

Como consequência do trabalho realizado, espera-se:

- aprendizagem de novas tecnologias e modelos associados a aplicações *server-side*;
- aprendizagem de outras tecnologias: aplicações *Web*; *cascade style sheet*; protocolo Jabber; Ldap;
- o desenvolvimento de aplicações distribuídas;

1.4 Organização da tese

1.4.1 Conteúdo da tese

Esta tese é composta por cinco capítulos e um apêndice. Os próximos parágrafos descrevem sucintamente o conteúdo de cada capítulo.

Capítulo 1 – Introdução. Neste capítulo é introduzido de uma forma genérica o problema em questão, dificuldades e características. Em sequência é relatada a motivação que levou a realização deste projecto, enquadramento temporal e grupos de trabalho, uma vez que o projecto no seu todo tem sido constituído por vários trabalhos realizados ao longo de 3 anos. São descritos ainda os objectivos desta tese.

Capítulo 2 – Estado da Arte. Este capítulo resulta da investigação de outras aplicações cujas funcionalidades são semelhantes às que se pretendem para este projecto. As aplicações analisadas representam o estado actual disponível no contexto pretendido. Neste capítulo são também analisadas as tecnologias de suporte à realização de um sistema deste tipo, nomeadamente baseadas em software livre e em Java, assim como também é justificada a sua utilização.

Capítulo 3 – Análise do Sistema. Este capítulo e o próximo são os dois capítulos que em conjunto formam o corpo desta tese. Neste primeiro é realizada uma abordagem teórica ao problema, o sistema é caracterizado de uma forma funcional e não funcional. Entenda-se por forma funcional a identificação de serviços que um sistema neste contexto deverá disponibilizar; na forma não funcional são identificados requisitos que o sistema deverá conseguir responder.

Capítulo 4 – Protótipo Desenvolvido. Conforme foi dito anteriormente este capítulo e o anterior dominam o corpo da tese. Este capítulo descreve a implementação da solução proposta. A sua organização baseia-se nos módulos do sistema, nomeadamente na componente *Web* e na componente que representa a lógica de negócio. Para além destas componentes é ainda relatado o processo de implementação de cada um dos componentes identificando problemas e soluções adoptadas. Na última secção deste capítulo é descrita a interface homem–máquina criada para este protótipo. Para melhor compreensão do sistema e dos módulos que o constituem recorre-se frequentemente a diagramas de classes e diagramas de sequências de acções para representarem modelos de dados e interacções entre módulos, respectivamente.

Capítulo 5 – Conclusões. Resume-se resultados obtidos, se corresponderam aos expectáveis, conclusões obtidas da utilização da tecnologia, ... no fundo se os objectivos propostos foram atingidos. No final integra ainda uma secção onde se discute trabalho a realizar no futuro e aspectos em aberto.

Apêndice A – Instalação e Configuração. Este apêndice corresponde a um guia de instalação, onde são identificados recursos e tecnologias necessários para executar o sistema, onde é descrito a organização da aplicação de distribuição, e onde é descrito a ferramenta de compilação e distribuição da aplicação.

Apêndice B – Glossário. Este apêndice colecciona palavras chave e seus significados encontradas ao longo do documento. A grande maioria corresponde à identificação de siglas com uma descrição resumida do seu significado.

Bibliografia. A tese termina com um conjunto de referências bibliográficas, umas cujos documentos serviram de apoio à investigação, outras serviram apenas para exemplos. Algumas referências cingem-se apenas a um endereço electrónico.

1.4.2 Convenções

No relatório, palavras que não sejam da língua portuguesa, estrangeirismos, e que estejam inseridas em texto corrido estão com fonte *itálico*. Tecnologias e produtos estão igualmente com fonte *"itálico mas com aspas"*. Blocos de código, referência a ficheiros ou a directorias assim como pacotes ou identificação de padrões de programação estão com fonte de `tamanho fixo`.

Algumas figuras foram copiadas de outros documentos, outras, devido à fraca qualidade do original, foram adaptadas do mesmo. Sempre que tal acontecer é feito referência na própria legenda da figura. As figuras criadas de raiz são identificadas por não terem nenhuma referência.

Capítulo 2

Estado da Arte

Este capítulo é constituído por duas componentes principais: uma componente que descreve as principais tecnologias que deram suporte à realização do sistema PUC; uma segunda parte onde são enumeradas e descritas algumas iniciativas de aplicações e sistemas relacionados com gestão de informação pessoal e gestão de comunicações pessoais. No final do capítulo é apresentada uma tabela onde figuram os vários sistemas investigados identificando características de gestão de informação pessoal e características de gestão de comunicações. Assim, promove-se discussão do que existe actualmente assim como também aquilo que os utilizadores estão habituados a usufruir e a esperar de um sistema com estas características.

2.1 Tecnologias de suporte

De seguida são apresentadas algumas tecnologias que deram suporte à realização do sistema PUC; suporte desde a concepção até à implementação. Algumas destas tecnologias foram utilizadas por mim directamente e também por outros grupos de trabalho no mesmo projecto (*J2EE*, *JBoss*, *JWMA*, *Jabber*, *JWMA*); outras foram utilizadas exclusivamente por outros elementos na concepção de outros serviços que integram o PUC (*J2ME*, *VoiceXML*).

O J2EE[1] e o J2ME[2] são especificações definidas pela *Sun MicroSystems*. *Jabber* é uma especificação para troca de mensagens assíncronas. O *JBoss*, e *JWMA*, são implementações Java cada qual com um objectivo concreto: o *JBoss* é uma implementação da especificação *J2EE*; o *JWMA* é uma implementação de um cliente de correio electrónico com interface *Web*. *VoiceXML* é mais uma especificação que permite a comunicação por voz via Internet.

[1] *Java 2 Enterprise Edition*
[2] *Java 2 Micro Edition*

Um denominador comum a todas as tecnologias referidas, nomeadamente as que correspondem às implementações, é que são todas desenvolvidas e distribuídas segundo o paradigma do código livre [70, 77].

2.1.1 Java 2 Enterprise Edition

O *"J2EE"* [31] é a especificação da SUN para plataformas de suporte de aplicações empresariais desenvolvidas em Java. A plataforma J2EE é baseada num modelo de aplicações distribuídas multi-camada oferecendo entre outras as seguintes características: (1) independência do sistema operativo e da arquitectura computacional; (2) a possibilidade de reutilização de componentes de software; (3) suporte transparente à persistência e ao processamento transaccional; e (4) modelo unificado de segurança.

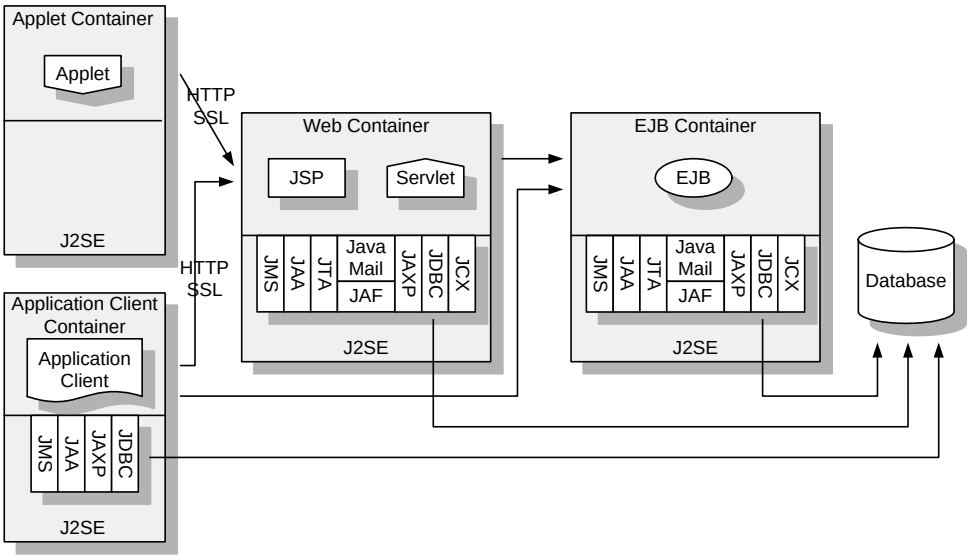

Figura 2.1: Arquitectura simplificada do J2EE (adaptado do original [31])

A figura 2.1 ilustra os principais módulos de uma aplicação desenhada (e desenvolvida) segundo a arquitectura J2EE. Em particular são evidentes a existência de distintos domínios de execução, designados geralmente por "contentores". Em geral, uma aplicação empresarial J2EE encontra-se distribuída por vários componentes executados nos seguintes domínios: contentor de *Applets* (tipicamente *Applets* executados no contexto de *Web browsers*); contentor *Web* (tipicamente *Servlets* e *JSPs*); e contentor EJB (tipicamente EJBs, *Enterprise Java Beans*, também designados por "componentes empresariais") [22].

2.1.2 JBoss

O *"JBoss"* [49] é um servidor aplicacional J2EE desenvolvido e promovido pelo *"JBoss Group"* de acordo com o paradigma do código livre, sendo distribuído segundo licença GLGPL[3]. O *"JBoss"* é um servidor muito popular [9] e usado em inúmeras soluções comerciais. O *"JBoss"* é um servidor de aplicações e enquadra-se entre o sistema de base de dados da empresa e o servidor *Web*, conforme figura 2.2. Normalmente é disponibilizado ao público integrado com o servidor *Web Tomcat*[4] [78] ou com o servidor *Web Jetty*[5] [66] que corresponde a uma solução mais simplificada.

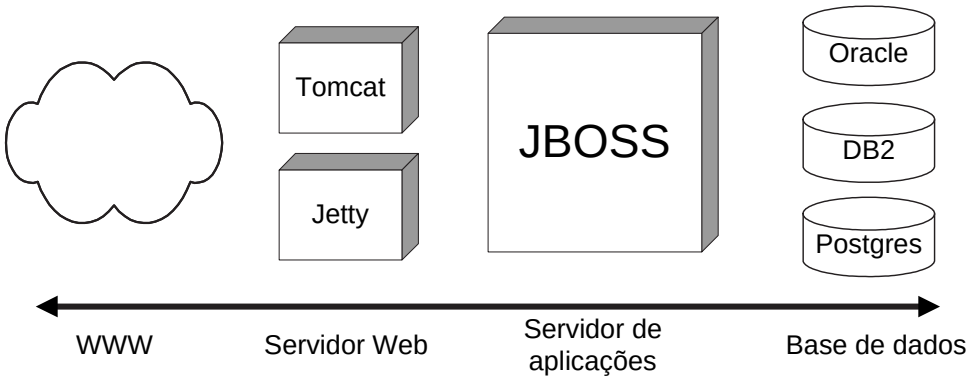

Figura 2.2: Arquitectura de um sistema com JBoss (adaptado do original [49])

O *"JBoss"* é constituído por diferentes componentes, nomeadamente: *JBossServer*, como contentor básico de EJB; *JBossMQ* para as mensagens JMS; *JBossMX* para correio electrónico; *JBossTX* para transacções JTA/JTS; *JBossSX* para segurança baseada em JAAS; *JBossCX* para conectividade JCA; e *JBossCMP* para gestão de persistência *CMP persistence*. O *"JBoss"* possibilita a integração modular de todos estes componentes através da arquitectura JMX[6] [63], conforme mostra a figura 2.3. Esta arquitectura permite substituir qualquer dos referidos componentes por um outro, desde que seja compatível com a arquitectura JMX e implemente as mesmas APIs. Tal modularidade possibilita o uso de outros componentes que não sejam fornecidos necessariamente pela empresa *"JBoss Group"*.

Uma das características mais apreciadas no *"JBoss"* é o designado *hot deployment*: a instalação de um componente de aplicação é tão simples como copiar pacotes JAR para

[3] *GNU Lesser General Public License*

[4] No âmbito do *Apache Jakarta Project*, o *Tomcat* é um contentor de *Servlets* que é usado na implementação de referência oficial para as tecnologias *Java Servlet* e *Java Server Pages*.

[5] *Jetty* é um servidor HTTP e um contentor de *Servlets* 100% Java. Significa que não é necessário configurar e correr separadamente o servidor *Web* (como o Apache) para se conseguir usar Java, *Servlets* e JSPs para gerar conteúdo dinâmico. Por ser em Java o servidor *Web* e o servidor aplicacional correm no mesmo processo aumentando a eficiência nas interligações entre contentores.

[6] *Java Management eXtension*

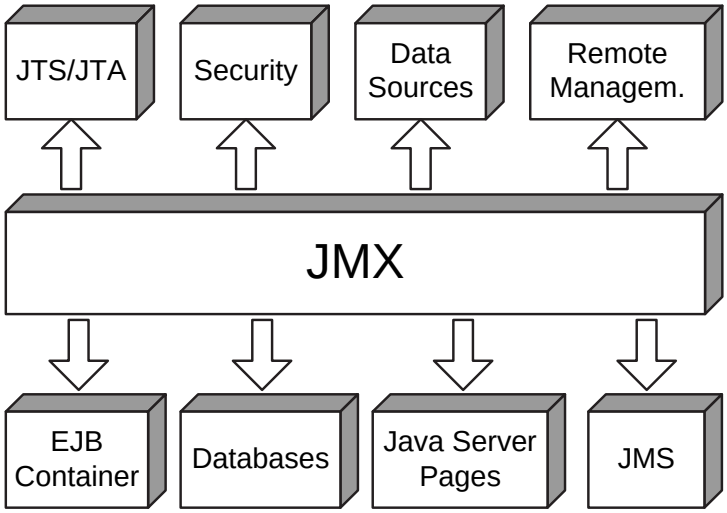

Figura 2.3: Arquitectura por módulos do JBoss (adaptado do original [49])

o directório de instalação. Se esta operação for efectuada quando uma versão anterior da aplicação estiver carregada, a nova versão substitui a mais antiga sem conflitos e sem ser necessário parar o servidor, i.e., não terá qualquer impacto noutras aplicações que estejam a correr no mesmo servidor.

Encontra-se já em fase de desenvolvimento um novo servidor de aplicações que, em vez de ter como base a arquitectura JMX, tem como base uma *framework* implementada sobre um conceito diferente que é o paradigma *Aspect Oriented Programming* [52]. Vai ser utilizado na versão JBoss 4.0 [5]. Não é um conceito novo; na realidade este conceito foi lançado em 1996 no seio da Xerox PARC [52]. Foi um paradigma que nunca teve grande investimento ao contrário do que aconteceu com outros paradigmas, nomeadamente o paradigma *Object Oriented Programming*. Actualmente está-se a verificar uma crescente utilização de paradigma AOP porque começa a ter uma maior utilização prática como por exemplo em servidores aplicacionais. AOP e OOP não são conceitos similares, antes, complementares. OOP é adequado para modelar comportamento comum numa hierarquia de objectos. A sua lacuna é conseguir aplicar comportamento comum num conjunto de objectos que não têm relações entre si. É sugestivo imaginar OOP como um desenvolvimento de software *top-down*, enquanto que AOP um desenvolvimento de software *left-right*, ou seja, horizontal. São, portanto, conceitos e paradigmas complementares. Os conceitos que estão por trás do OOP são herança, encapsulamento, polimorfismo; os conceitos que estão por trás do AOP são avisos/intercepções, introduções, *metadata*, e pontos de intercepção [5]. Os exemplos clássicos de utilização de AOP são mecanismos de *logging*, de segurança, e de monitorização. Actualmente, e no contexto de um servidor de aplicações, poderá ser utilizado para implementar serviços de segurança, autenticação,

mecanismos de transacções, entre outros.

2.1.3 Java 2 Micro Edition

O *"J2ME"* [33] é a plataforma Java adaptada aos dispositivos embebidos tais como telemóveis, PDA's, *set-top boxes* de televisão digital, computadores de bordo de automóveis.

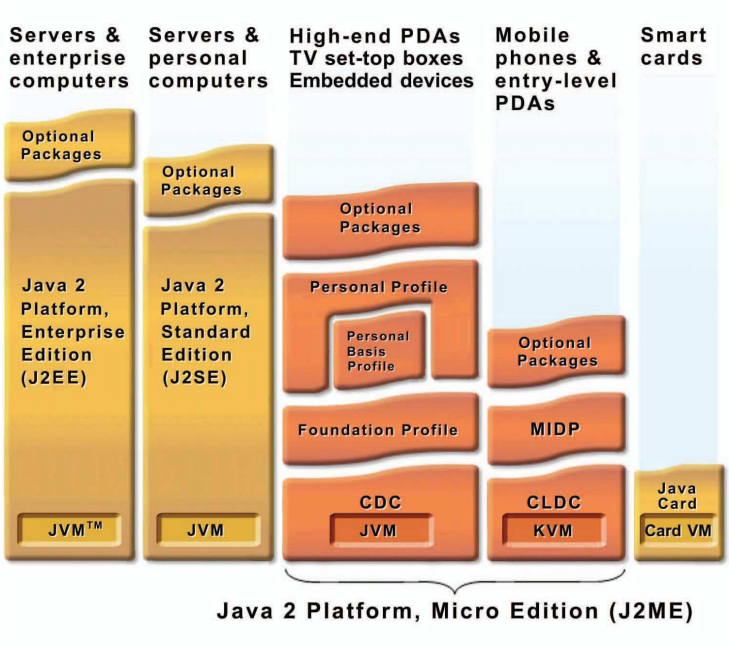

Figura 2.4: Comparação entre as plataformas J2EE, J2SE e J2ME (copiado do original [33])

A compatibilidade com um conjunto tão distinto de dispositivos é conseguida através da definição de configurações e perfis.

Configurações

Actualmente o J2ME possui duas configurações, conforme sugerido na figura 2.4:

CDC[7] [28][25] - esta configuração foi desenvolvida para dispositivos com maior capacidade de memória, processadores mais rápidos e ligações com maior largura de banda, tais como *set-top boxes* de televisão interactiva, computadores de bordo de automóveis ou PDA topo de gama.

CLDC[8] [27][21] - é a mais limitada das duas configurações, implementada tendo em vista os dispositivos com ligação intermitente à rede, processadores mais lentos e capacidade de memória limitada, como sejam telemóveis ou PDA. Estes dispositivos tipicamente

[7] *Connected Device Configuration*
[8] *Connected Limited Device Configuration*

dispõem de CPU de 16 ou 32 bits e um mínimo de 128 KB até 512 KB de memória disponível.

Perfis

De maneira a proporcionar um ambiente de *runtime* direccionado para uma categoria específica de dispositivos, as configurações têm de ser combinadas com um conjunto de API de alto nível, designados por "perfis", que definem o modelo do ciclo de vida da aplicação, a interface de utilizador e o acesso a propriedades específicas de determinado dispositivo. Existem quatro "perfis". Uma para a configuração CLDC e três para a CDC. São as seguintes:

MIDP [29][19] O MIDP[9] foi desenvolvido para telemóveis e PDAs de gama baixa. Ele proporciona as funcionalidades essenciais para aplicações móveis, incluindo a interface utilizador, conectividade à rede, armazenamento de dados e gestão da aplicação. Combinado com o CLDC, o MIDP providencia um completo ambiente JAVA que permite aproveitar ao máximo as capacidades dos dispositivos móveis e minimiza tanto a utilização de memória como de energia. Foi este o perfil utilizado no desenvolvimento do cliente J2ME no trabalho realizado no contexto do PUC2 [15].

***Foundation Profile* [30][26]** Os perfis do CDC estão dispostos em camadas de maneira a permitir a adição de perfis à medida que são necessários para fornecer as funcionalidades necessárias aos diferentes tipos de dispositivos. O *Foundation Profile* é o perfil de mais baixo nível para CDC. Ele fornece uma implementação do CDC com capacidade de ligação a rede que pode ser usado para implementações bastantes embebidas sem necessidade de interface utilizador, podendo ser combinado com o *Personal Basis Profile* e o *Personal Profile* para dispositivos que necessitam de uma interface de utilizador gráfica.

***Personal Profile* [32][23]** O *Personal Profile* é o perfil CDC direccionado para dispositivos que necessitam de uma completa interface gráfica para o utilizador ou suporte para *Applets* Internet, tais como PDA topo de gama e consolas de jogos. Inclui bibliotecas AWT[10] e proporciona *Applets* de fácil utilização baseados na *Web* desenvolvidos para utilização num ambiente *desktop*. O *Personal Profile* substitui a tecnologia *Personal Java* (utilizada por exemplo no *Nokia Communicator*) e proporciona uma fácil migração para a plataforma J2ME.

[9] *Mobile Information Device Profile*
[10] *JAVA Abstract Window Toolkit*

Personal Basis Profile **[20][24]** O *Personal Basis Profile*, é um subconjunto do *Personal Profile* que proporciona um ambiente aplicacional para dispositivos com ligação à rede que necessitam de um nível básico de apresentação gráfica ou que necessitam da utilização de ferramentas gráficas especializadas para aplicações específicas. Exemplos de dispositivos são *set-top boxes* de TV, computadores de bordo de automóveis e quiosques de informação.

2.1.4 JWMA

O *"JWMA"* (*Java Web Mail Architecture*) [89] é um projecto Java, com código livre, que oferece uma solução de correio electrónico com interface *Web*.

O *"JWMA"* suporta-se nas seguintes tecnologias/API: Java Mail API, para acesso ao servidor de correio electrónico; e *Java Server Pages* e *Java Servlet API*, para gestão e produção de interfaces homem–máquina *Web*.

A figura 2.5 ilustra a arquitectura do *"JWMA"*, a qual segue o padrão MVC (*Model-View-Control*) [53, 18], padrão arquitectural bastante conhecido que distingue claramente componentes de modelo, de componentes de controlo, e de componentes de apresentação.

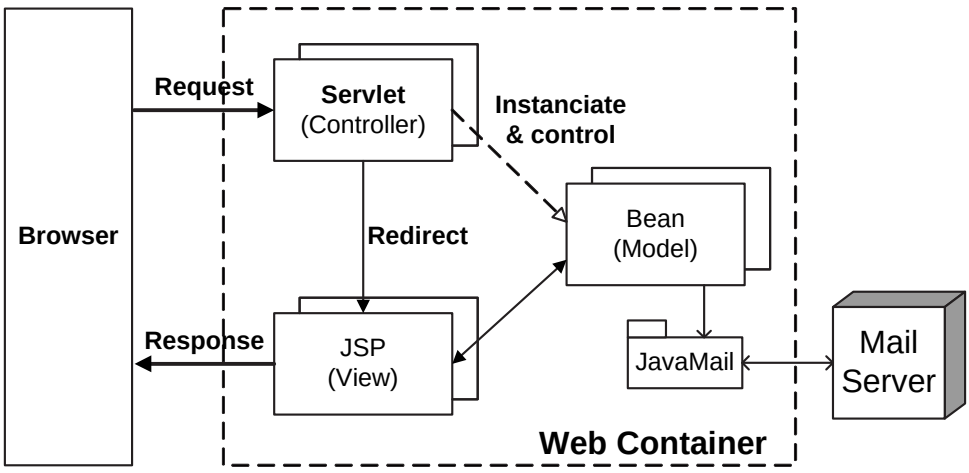

Figura 2.5: Arquitectura de software do JWMA (adaptado do original [89])

Neste caso concreto os *Servlets* desempenham o papel de componentes de controlo; são o ponto de entrada da aplicação, ou seja, recebem os pedidos efectuados pelo utilizador. Os *Beans* desempenham o papel de componentes do modelo; é através deles que é realizado o acesso ao servidor de correio electrónico. As JSPs (*Java Server Pages*) desempenham o papel de componentes de apresentação; através delas são criadas as páginas HTML com que o utilizador vai interagir.

2.1.5 Jabber

O *"Jabber"* [48] é um protocolo de comunicação para troca de mensagens na Internet, definido em XML[11] conforme se pode verificar na figura 2.6, e que disponibiliza um conjunto de funcionalidades muito potentes e expansíveis. Recentemente, foi aprovada a criação de um grupo de trabalho do IETF[12] para a normalização deste protocolo definido, neste contexto, como XMPP[13] [46].

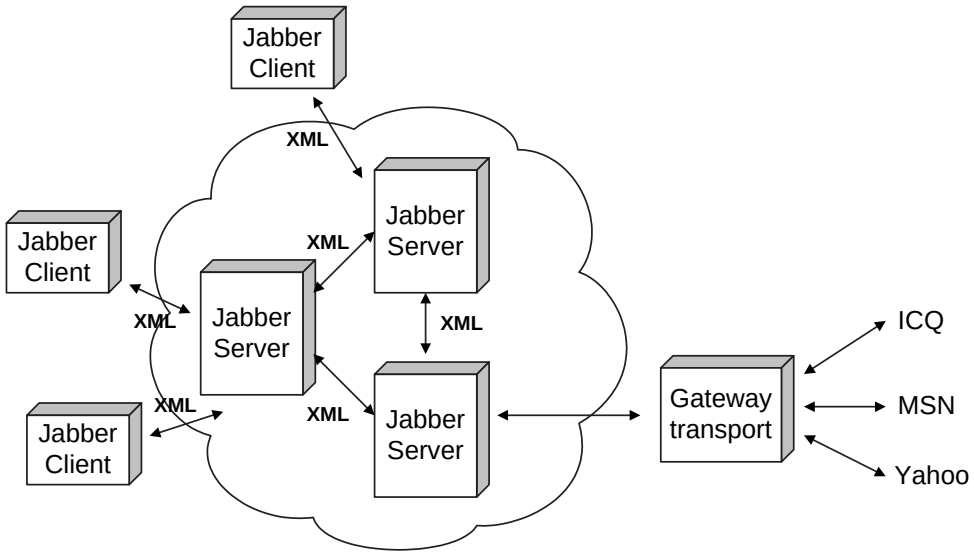

Figura 2.6: Arquitectura Jabber (adaptado do original [48])

Uma das suas aplicações práticas é na implementação de sistemas para troca de mensagens instantâneas e notificações de presença. O *"Jabber"* é um protocolo definido e desenvolvido de acordo com o paradigma especificação livre. Esta tecnologia é suportada por uma grande comunidade de utilizadores e de programadores. Apesar de ser uma tecnologia que assenta no paradigma especificação livre o *"Jabber"* tem um forte apoio da indústria, sendo o "Jabber.com" o seu principal representante comercial (e.g., o serviço "Mensageiro" do Portal SAPO foi fornecido através desta organização).

O protocolo *"Jabber"* permite efectuar operações adicionais ao nível da aplicação, e.g., pesquisas de bases de dados (perfil de utilizadores, serviços de directório), gestão de acesso.

2.1.6 VoiceXML

O *"VoiceXML"* [85] é a especificação de uma linguagem XML usada para criar interfaces homem–máquina baseadas em voz, especialmente usadas em terminais do tipo telefone.

[11] *eXtensible Markup Language*
[12] *The Internet Engineering Task Force*
[13] *eXtensible Messaging and Presence Protocol*

Usa reconhecimento de voz assim como as teclas do telefone (DTMF[14]) como input; e áudio previamente gravado e síntese de texto para voz (TTS[15]) como *output*.

O *"VoiceXML"* foi desenhado para permitir um controlo total na definição de interacções ou de diálogos (falados) entre os utilizadores e as aplicações. O *VoiceXML* oferece várias capacidades, nomeadamente formas de controlar o output áudio, input áudio, lógica de apresentação e fluxo de controlo, manuseamento de eventos e ligações telefónicas básicas.

O *"VoiceXML"* facilita a rápida criação de novas aplicações e esconde dos programadores alguns detalhes de implementação. Por exemplo, separa a camada da interacção com o utilizador da camada lógica do serviço.

Segundo a figura 2.7 uma aplicação *"VoiceXML"* consiste genericamente:

- Num Servidor de Media *"VoiceXML"*, que corre um interpretador *"VoiceXML"*, o qual actua como cliente para o servidor de aplicações. Este interpretador compreende diálogos *"VoiceXML"* e controla recursos de voz e de telefonia. Estes recursos incluem ASR[16], TTS, funções de toque e gravação, assim como uma interface para a rede telefónica.
- Num Servidor de Aplicações que gera o *"VoiceXML"* de acordo com a execução duma lógica da aplicação, que pode fazer uso duma base de dados ou servidor de transacções.

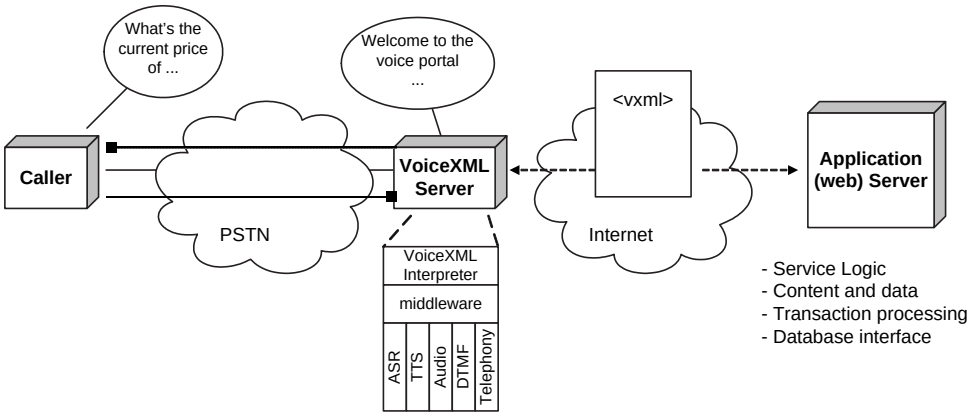

Figura 2.7: Componentes duma aplicação VoiceXML (adaptado do original [85])

2.1.7 Resumo

Esta secção resume as várias tecnologias que deram suporte à realização do sistema.

[14] *Dual Tone Multi-Frequency*: usado por telefones de teclado. DTMF atribui um par de frequências específicas, ou tom, para cada tecla.

[15] *text-to-speech*

[16] *Automatic Speech Recognition*

Segundo a tabela 2.1, uma das constatações que se pode verificar é que todas estas tecnologias têm subjacente a plataforma Java. A utilização da plataforma Java correspondeu a uma das primeiras decisões aquando a análise do estado da arte. Um responsável preponderante por esta decisão foi um dos participantes do projecto, a PT Inovação, que tinha como premissa a utilização desta plataforma.

Uma vez analisada a tecnologia, concordo igualmente que seja utilizada esta plataforma. Nomeadamente, por se tratar de uma plataforma já com alguma maturidade, estar estável, já ter mostrado ser capaz de dar resposta a problemas com igual ou superior complexidade, haver vários exemplos da sua utilização em produtos comerciais [10], existir já muito trabalho sobre esta plataforma, existir uma forte comunidade na concepção e implementação de componentes código livre que se podem integrar posteriormente num sistema. Aliás, verifica-se que para todas as tecnologias analisadas, ou correspondem a especificações públicas, ou correspondem a implementações código livre.

Tecnologia	Descrição	Tipo
J2EE	Especificação da Sun para plataformas de suporte de aplicações empresariais.	Especificação
J2ME	Especificação da Sun para sistemas embebidos e com recursos limitados.	Especificação
Jabber	Protocolo de comunicação em XML para troca de mensagens assíncronas.	Especificação
VoiceXML	Linguagem baseada em XML usada para criar interfaces homem-máquina baseadas em voz.	Especificação
JWMA	Implementação de um cliente Web para acesso a um servidor de correio electrónico.	Implementação
Jboss	Implementação da especificação J2EE, mais precisamente do servidor de componentes distribuídos.	Implementação

Tabela 2.1: Resumo das tecnologias analisadas

2.2 Aplicações similares

De seguida são enumeradas e descritas algumas iniciativas onde se identificam, sempre que possível, requisitos e funcionalidades das mesmas. Para cada aplicação são identificadas igualmente semelhanças e diferenças com o sistema desenvolvido no âmbito desta tese. No final desta secção é apresentada uma tabela onde estão identificadas e reunidas características PUC[17] e PIM[18] de cada uma das aplicações analisadas.

[17] *Personal Unified Communication*
[18] *Personal Information Management*

2.2.1 Microsoft Outlook

O *"Microsoft Outlook"* [57] é a referência no que diz respeito a aplicações de gestão de informação pessoal e de gestão de comunicações pessoais. Todas as outras iniciativas, incluindo o sistema PUC, têm como objectivo disponibilizar serviços semelhantes ou próximos aos disponibilizados pelo *"Microsoft Outlook"*. Sendo essa a referência não significa que outras iniciativas não possam oferecer outros serviços, ou mesmo os mesmos mas com outras características; serão sempre mais valias.

De entre as várias funcionalidades do *"Microsoft Outlook"* destacam-se: agenda; contactos; correio electrónico; tarefas; notas. Algumas destas funcionalidades estão bastante correlacionadas e integradas entre si como por exemplo a agenda e as tarefas. Para cada um destes serviços é permitido realizar um sem número de acções com vista a gerir a informação do utilizador. Verifica-se que a maior parte dos serviços identificados relacionam-se com serviços de gestão de informação pessoal. O único que pertence à categoria de gestão de comunicações pessoais é o serviço de correio electrónico.

Os serviços disponibilizados não são completamente independentes uns dos outros. Há determinadas operações que estão presentes em todos os serviços. Por exemplo, todas as operações que permitem a parametrização integrada dos próprios serviços.

Uma funcionalidade bastante útil e que é comum a todos os serviços, ou pelo menos é uma colecção de informação dos vários serviços, é o *"Outlook Today"* que permite ao utilizador verificar de uma forma rápida e eficiente toda a informação respeitante ao dia corrente, ou aos próximos dias.

Uma outra funcionalidade, também ela interessante, é a possibilidade de notificação assíncrona da ocorrência de um evento (*reminder*). Estas notificações são comuns no serviço de agenda e no serviço de tarefas.

No serviço de agenda existe o conceito de evento, marcação, e de reunião (encontro).

"Marcações" são actividades que se podem marcar na agenda sem que envolvam terceiros ou recursos. Associado às marcações estão as notificações assíncronas. Um conceito inovador é uma marcação poder ser configurada para ser partilhada entre diferentes utilizadores. Esta funcionalidade revela a existência de colaboração.

Um "encontro" é uma marcação onde se convida um ou mais utilizadores ou onde se reserva recursos. As respostas aos pedidos de encontro são recebidas via correio electrónico onde se verifica a interacção de serviços.

Um "evento" é uma actividade cuja intenção é durar no mínimo 24 horas. Normalmente significa uma conferência ou férias. Pode haver eventos pontuais (seminário) ou anuais (aniversário). Por omissão um evento é mostrado como tempo livre do utilizador enquanto que uma marcação é mostrada como ocupado.

Uma outra funcionalidade também ela presente em todos os serviços é a possibilidade de organização da informação; a estratégia de organização da informação não é imposta pelo sistema. Por exemplo, notas e mensagens no *inbox* podem ser organizadas por pastas; itens de agenda, contactos, ou tarefas podem ser organizados por categoria.

O serviço de correio electrónico é semelhante aos serviços de correio electrónico existentes noutras aplicações. Ao nível de ligações, para envio de mensagens permite o estabelecimento de uma ligação a um servidor SMTP[19] e para recepção de mensagens permite o estabelecimento de ligações a servidores POP[20] e IMAP[21].

A grande desvantagem do *"Microsoft Outlook"* é que se trata de uma aplicação com instalação local com todas as desvantagens que isso acarreta: é dependente da plataforma Windows; obriga o utilizador à efectuação de todo um processo de instalação; obriga a que o acesso ao serviço seja sempre através do dispositivo onde existe a instalação; na maior parte das utilizações a informação gerada ou recebida está guardada num repositório local (sistema de ficheiros). Esse facto obriga a que o acesso seja sempre local. Esta será a pior desvantagem, porque impossibilita a mobilidade; impossibilita um sem número de características proporcionadas pela mobilidade. Por exemplo, impossibilita que, de qualquer lugar, de qualquer dispositivo com acesso à Internet, se possa usufruir do serviço. É importante não esquecer que se trata de uma desvantagem apenas enquanto os dados estiverem apenas num repositório local.

A partir da versão *"Outlook 2002"* é possível definir repositórios remotos onde a informação passa a estar disponível remotamente. Na versão *"Outlook 2003"* existe o conceito de serviços remotos, como por exemplo, o serviço *Share Point* (ponto de partilha) ou o serviço *Microsoft Office Internet Free/Busy*, onde o utilizador poderá guardar e publicar informação do tipo contactos, disponibilidade, reuniões. Acontece que a utilização destes serviços não é natural; obriga uma série de configurações por parte do utilizador. Mais, segundo a minha opinião, o *"Outlook"* vive ainda ensombrado pela tradição de uma aplicação local onde, sem que haja uma forte divulgação, continuarão a ser serviços pouco utilizados ou apenas utilizados por quem tenha uma forte motivação.

Aplicações comerciais

O *"Microsoft Outlook"* é um produto final disponibilizado pela empresa Microsoft, e como tal não serve de base à implementação de outros sistemas. Sendo o sistema operativo Windows aquele que domina o mercado dos PCs pessoais é natural que a aplicação *"Microsoft Outlook"* também o seja no que diz respeito às aplicações de comunicação e de gestão

[19] *Simple Message Transport Protocol*
[20] *Post Office Protocol*
[21] *Internet Message Access Protocol*

de informação pessoal. Existem vários estudos onde identificam que cerca de 90% dos utilizadores da Internet fazem-no utilizando um sistema operativo *Windows* [45, 80].

Tendo sido já identificado como uma desvantagem o acesso local, e sendo ainda assim a aplicação mais usada, houve necessidade de muitas aplicações móveis disponibilizarem mecanismos de sincronização de informação com o *"Microsoft Outlook"*. A própria Microsoft disponibiliza esta funcionalidade em aplicações instaladas em dispositivos com o sistema operativo *Windows CE*.

2.2.2 Weblicon

A empresa *"Weblicon Technologies AG"* [86] desenvolveu e vende dois serviços que no contexto desta tese são importantes referir: um "organizador virtual" e um "servidor SyncML" para sincronização.

O organizador virtual oferece funcionalidades de calendário, gestão de contactos, correio electrónico, disco virtual, tarefas, portanto todo um conjunto de serviços que formam uma solução do tipo PIM. Pode ser facilmente integrado com uma rede de servidores, aplicações ou base de dados.

Weblicon PIM

O *"Weblicon PIM"* é um organizador virtual que inclui serviço de mensagens e gestão pessoal de informação. A componente de gestão de comunicações pessoais unificadas, para simplificação de escrita PUC, inclui correio electrónico, mensagens SMS e MMS assim como a componente de gestão de informação pessoal, PIM, inclui contactos, calendário, tarefas, e sincronização via *SyncML* [69]. Tem uma arquitectura que suporta o acesso ao serviço por diferentes meios físicos: *web*, móvel e voz. Dados os diferentes meios físicos vão existir diferentes clientes para o acesso ao serviço: HTML[22] para *web*; WAP[23], i–mode[24] [56], J2ME, para acesso móvel; interface *VoiceXML* para acesso voz.

Segundo a figura 2.8 verifica-se que tem igualmente a capacidade de se sincronizar com diferentes aplicações *desktop*, nomeadamente o *Microsoft Outlook*, *Lotus Notes*, *Palm PDAs* e *PocketPCs* graças à utilização do standard da indústria *SyncML*. Através desta tecnologia permite a sincronização OTA[25] com dispositivos móveis que tenham igualmente esta capacidade.

A aplicação *"Weblicon PIM"* é baseada em *standards* da indústria. É uma aplicação escrita na linguagem Java, mais precisamente executa-se sobre um servidor de aplicações

[22] *HyperText Markup Language*

[23] *Wireless Application Protocol*

[24] Sistema de acesso à Internet móvel utilizado pela companhia NTT DoCoMo's, líder do mercado móvel Japonês.

[25] *Over The Air*

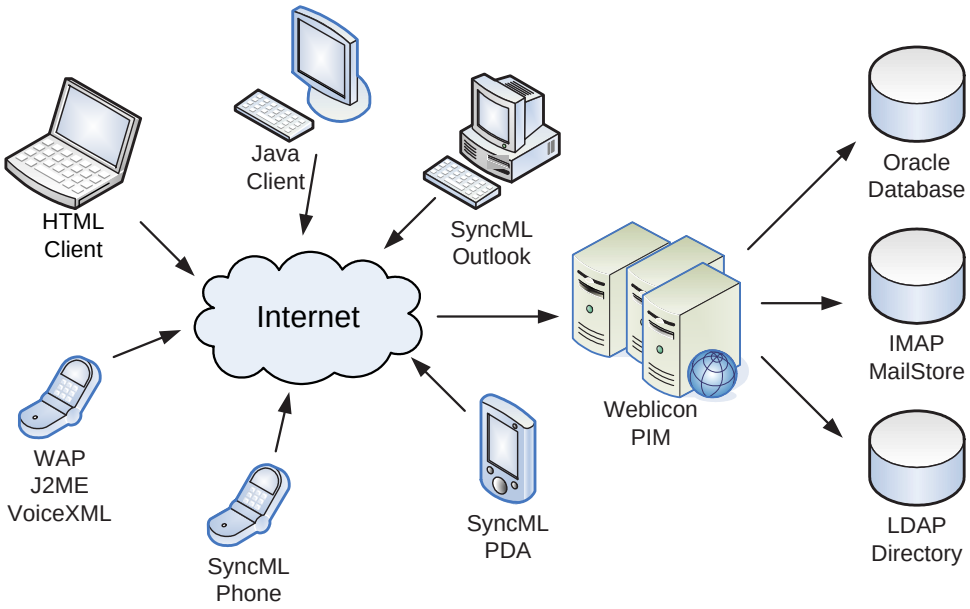

Figura 2.8: Arquitectura multi–acesso do Weblicon (adaptado do original [86])

J2EE o que a torna independente da plataforma. É também independente da base de dados, podendo ser integrada com diferentes servidores LDAP[26] e servidores IMAP[27] e disponibiliza uma API XML para integração com serviços externos através de *Web Services*.

A arquitectura foi realizada tendo em conta as características que um serviço deste tipo necessita providenciar, nomeadamente, escalabilidade, confiança, e inter–operabilidade. Todo o processo de desenho e implementação foi realizado em estreita colaboração com a indústria das telecomunicações obtendo uma arquitectura robusta e flexível que responda às necessidades de uma indústria deste tipo.

Weblicon SyncML Server

O *"Weblicon SyncML Server"* suporta sincronização dos dados centralizados ligados ao PIM (lista de contactos, de eventos e de tarefas) com *Microsoft Outlook*, *Outlook Express*, *Lotus Notes/Domino*, *Palm* e *PocketPC PDAs* assim como sincronização sem fios com dispositivos móveis com capacidade *SyncML*. Tal sincronização pode ser observada na figura 2.9. Usando esta arquitectura, o *"Weblicon SyncML server"* pode ser integrado com serviços do tipo PIM já existentes, como por exemplo o *Sun ONE Calendar server* e o *LDAP Directory server*. O *"Weblicon SyncML server"* foi oficialmente certificado na *"Sync Fest"* em *Boston* em Abril de 2002 garantindo compatibilidade com outros dispositivos com capacidade *SyncML*. Existe suporte para que a sincronização dos dados

[26] *Ligthweight Directory Access Protocol*
[27] *Internet Message Access Protocol*

seja bidireccional. A sincronização é realizada sobre HTTP[28].

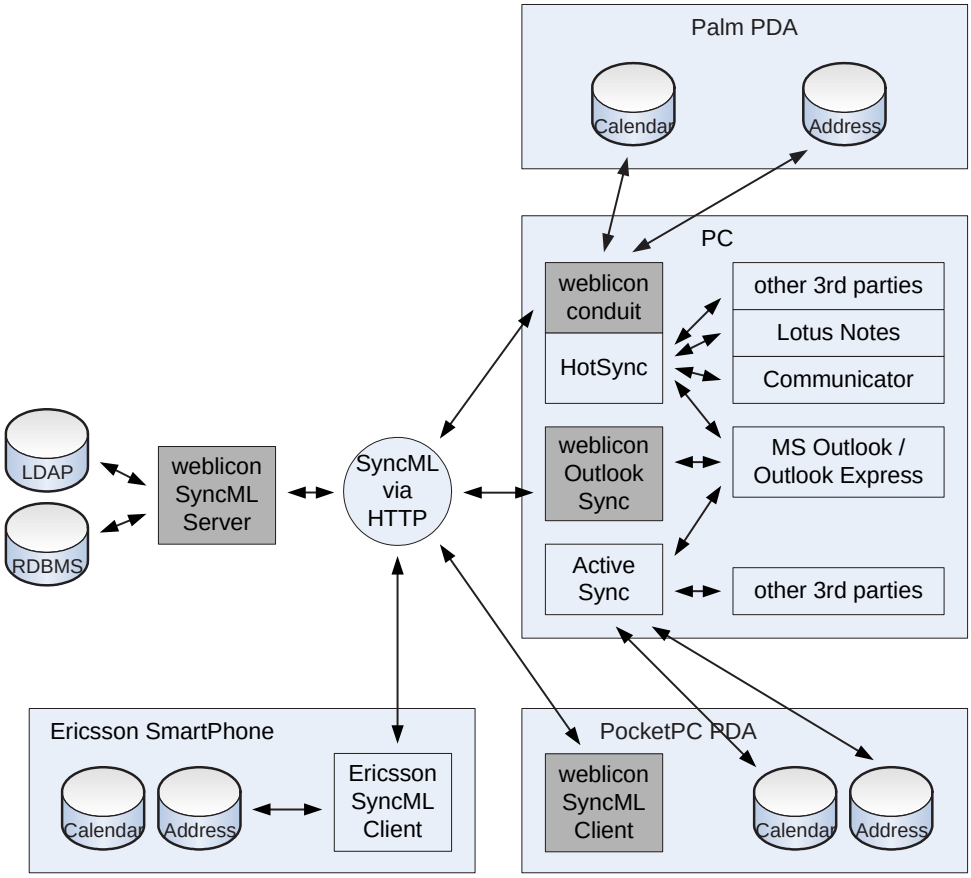

Figura 2.9: Arquitectura SyncML no Weblicon (adaptado do original [86])

Aplicações comerciais

No *site* da empresa que desenvolveu este sistema é possível encontrar um número considerável de aplicações que se suportam sobre esta plataforma [87].

- **WESTEL/T-Mobile.** Operadora na Hungria utiliza o *"Weblincon SyncML Server"* como suporte do sistema e todos os seus clientes utilizam *SyncML* para uso móvel.
- **Macgate.** Primeiro portal *Apple* que oferece um organizador *online* com ambiente *Apple*. Tem como principal alvo a comunidade *Apple* Alemã.
- **Debitel.** Oferece o *"Weblicon Organizer"* como "Mein Büro" e "Mein Mail". O serviço foi oficialmente introduzido no *CeBit* 2004.
- **The Strato Communicator.** Oferece acesso via Internet ao correio electrónico a cerca de 1.2 milhões de clientes, em combinação com o primeiro serviço *SyncML* disponível na Alemanha.

[28] *HyperText Transport Protocol*

- **Siemens AG.** Licenciou o *"weblicon SyncML server"* e todos os clientes *"weblicon SyncML"*. A tecnologia *"weblicon SyncML"* tornou-se de-facto um standard em sincronização para fabricantes de dispositivos móveis.
- **Mobilkom Austria.** Mais de 3 milhões de Austríacos fazem a suas chamadas com a A1. Corresponde a um valor acima de 44% de utilizadores de móveis do mercado Austríaco. Foi a primeira operadora a oferecer a tecnologia *SyncML* aos seus subscritores.
- **Sun microSystems.** A infraestrutura da *SunOne* utiliza a tecnologia *"Weblicon SyncML server"*. Por sua vez o sistema de mensagens da *SunOne*, calendário e servidores de directorias são utilizados em diversas instalações de operadoras.
- **KPN Mobile**. Tradução de uma citação de Sander den Herder, gestor de produtos na operadora: *"O "organizador weblicon" oferece uma personalização fácil e uma excelente interface com o utilizador. Facilita à equipa a adopção de requisitos ambiciosos assim como integração atempada; é a razão pela qual a "weblicon" foi escolhida entre seis dos principais competidores no US."*
- **E-Plus Mobilfunk.** Providencia esta operadora alemã com uma lista de contactos pessoal e um diário *online* que podem ser acedidos via *Web* e *WAP*.
- **Midray.** Juntamente com os seus clientes e parceiros é capaz de desenvolver soluções à medida com características móveis.
- **BMW.** Em parceria com a *"weblicon"* desenvolvem vários projectos relacionados com a gestão de informação pessoal.

2.2.3 Hambo

"Hambo" [36] é um sistema que potencia a oferta de serviços móveis. À semelhança de outros projectos, o *"Hambo"* é livre e é código aberto.

"Hambo" é independente dos dispositivos, permitindo que os utilizadores possam aceder aos serviços pessoais, através de qualquer dispositivo ligado à Internet. Acenta numa arquitectura onde existe o conceito de *plug–in* para novos componentes, facilitando a expansão e criação de novos serviços.

Toda a arquitectura baseia-se em standards da indústria incluindo Java, XML, WML[29], HTML, *palm7 web–clipping* e *i–mode* [56]. O facto de ser código aberto potencia o suporte de novos formatos como por exemplo *voiceXML* e XHTML[30], como também facilita a integração com sistemas já existentes. Está já preparado para GPRS[31], UMTS[32], PCS[33]

[29] *Wireless Markup Language*
[30] *eXtensible HyperText Markup Language*
[31] *Gena eral Packet Radio Service*
[32] *Universal Mobile Telecommunications System*
[33] *Public and Commercial Services Union*

e GSM[34].

O projecto *"Hambo"* consiste num conjunto de módulos de aplicações e servidores. Os diferentes serviços oferecidos pelo sistema são:

- **Informação** - notícias em diferentes categorias; boletim meteorológico; horóscopo.
- **Gestão Pessoal de Informação** - serviço de mensagens (via SMS ou correio electrónico, calendários, tarefas (notificações via SMS e correio electrónico) e contactos (funções de procura, grupos de contactos, notificações de aniversários, ...).
- **Entretenimento** - toques, imagens.
- **Portal Personalizado** - personalização da página inicial e lista de *bookmarks*. Podem desenhar o seu próprio *site* de Internet móvel pessoal.
- **Directórios para dispositivos móveis** - consiste na categorização de *links* para serviços móveis disponíveis por outras entidades.
- **Conversação e comunidade** - permite aos utilizadores criar e gerir comunidades virtuais móveis no seu portal. Um motor de busca por interesse facilita o utilizador a iniciar o seu círculo de amigos *online*. Permite ver quem está *online*, conversar com amigos.

Aplicações comerciais

Na investigação realizada não foi encontrada a utilização desta plataforma em aplicações comerciais. Este projecto está catalogado no portal de código livre *sourceforge* [36]. O seu estado encontra-se como concluído não se verificando alterações desde Maio de 2002, data da última actualização no portal.

Este projecto não é apenas uma plataforma de serviços; inclui igualmente aplicações móveis finais e prontas a usar em quatro áreas: (1) serviços de informação (notícias, meteorologia, ...); (2) serviços de comunicação (correio electrónico, SMS, ...); (3) serviços de comunidade (chat, ...); (4) e serviços de entretenimento (toques, imagens, ...).

2.2.4 Chandler

O *"Chandler"* [6] é mais um PIM com o objectivo de concentrar e facilitar a gestão de informação e comunicações de um utilizador. Estamos a falar essencialmente de serviços de envio e recepção de correio electrónico, gestão de calendário e agenda, e gestão da lista de contactos. Actualmente, ainda se encontra em fase de desenvolvimento.

A principal particularidade deste sistema, e que o distingue de outros, é permitir a partilha de informação entre utilizadores. Por esta particularidade é também referido como um *Interpersonal Information Manager*.

[34] *Global System for Mobile Communications*

É intenção ser código aberto na gestão pessoal de informação, como também ser uma plataforma de desenvolvimento de aplicações de gestão de informação, é independente da plataforma, usando a linguagem *Python* [81].

Gestão de informação pessoal

O conceito do *"Chandler"* é permitir ao utilizador organizar a sua informação segundo os seus critérios e não segundo critérios do sistema. Permite relacionar itens de tipos diferentes e agrupá-los num mesmo lugar segundo um mesmo contexto. E estamos a falar de tipos de informação tão díspares como por exemplo, mensagens de correio electrónico, lista de mensagens de correio electrónico, mensagens instantâneas, notas, contactos, tarefas, eventos, encontros, documentos, páginas *web*, *bookmarks*, fotos, MP3's, *blogs*, ... Os dados podem ser guardados numa máquina local, noutras máquinas, ou recursos partilhados como por exemplo servidores. Providenciará mecanismos de procura avançada que permitirá procurar informação em repositórios locais e em outros repositórios em rede. Os resultados dessa busca poderão ser guardados para posterior utilização.

A grande vantagem do *"Chandler"*, e é o que está na base do seu desenho, é conseguir extrair informação relevante a partir de recursos tão diferentes.

Calendário

Irá ter com certeza as funcionalidades normais de um calendário. Mas acima de tudo, a principal característica que irá distinguir o *"Chandler"* é a possibilidade de partilha de informação de um utilizador directamente com outro utilizador sem que tenha que haver recursos centralizados com uma gestão central.

Partilha e colaboração

A partilha não se resume ao calendário; a partilha pode ser de qualquer informação. Permite maior interacção, menos esforço, e coordenação de actividades.

Existe facilidade na sincronização e actualização de informação entre repositórios.

O ambiente de colaboração providenciado pelo *"Chandler"* facilita a discussão, organização e coordenação de projectos, criação e revisão de documentos, gestão do fluxo de informação e de tarefas.

Chandler como plataforma

O *"Chandler"* é uma plataforma aberta e não está limitado às funcionalidades previstas inicialmente. Em qualquer momento com o agrupamento de diferentes serviços pode-se

criar novas aplicações ou novos serviços, mudar a aparência de uma aplicação, acrescentar interfaces, configurar serviços. Ao nível programático existe o conceito de parcela que é a unidade básica da organização do código do *"Chandler"*. Logo é permitido aos programadores criarem novas parcelas ou alterarem existentes para obterem novas funcionalidades. As parcelas estão escritas em *wxPython* (*Python + wxWindows*) [73].

Aplicações comerciais

Como referido, o *"Chandler"* encontra-se em fase de desenvolvimento não dando, por isso, e ainda, suporte a nenhuma aplicação comercial. No entanto, existe uma estratégia bem definida para que haja uma forte adesão ao produto. A esta estratégia eles descrevem como sendo a "Teoria da Adopção" [7].

O mercado objectivo que o *"Chandler"* serve, gestão de informação pessoal, é extremamente amplo e diversificado. Acreditam que existem melhores formas que as actuais de servirem este tipo de mercado. Nomeadamente, estará criada a oportunidade de como as mensagens, eventos, tarefas e outros tipos de informação poderão ser organizados e apresentados às pessoas segundo o contexto correcto e por forma a tirarem o melhor partido desta informação. Acreditam igualmente que a partilha e a colaboração de tal informação é uma área com potenciais inovações (ainda pouco explorada).

Definiram etapas distintas para o desenvolvimento do produto.

Canoga→Westwood→Pasadena→Futuras versões

A estes nomes fazem corresponder versões do *"Chandler"* conforme se pode verificar na tabela 2.2. Nesta tabela está ainda identificado o público alvo para cada uma das etapas, assim como as funcionalidades a disponibilizar e também uma escala temporal para cada uma das etapas. Esta tabela é um resumo da informação encontrada em [7].

2.2.5 Elefante

Tal como o *Microsoft Outlook* o *"Elefante"* não é uma plataforma de serviços; é sim já uma instanciação de um produto. Ou seja, não é utilizado para criar outros produtos, é antes uma solução final e comercial.

O *"Elefante"* [17] é um assistente pessoal virtual criado no Brasil, Rio de Janeiro, em Agosto de 1997. É considerado o número um do mercado ibero–americano conseguindo atingir actualmente os 1,3 milhões de utilizadores no Brasil; recebeu uma série de prémios pelos serviços *on-line* que oferecia tendo sido eleito em 2002 o número um na Internet móvel. A imagem de marca é um elefante rosa; a justificação para o logotipo é a sua capacidade de memorização: "memória de elefante". A intenção do sistema é libertar

Etapa	Versão	Público Alvo	Focus	Escala temporal
Canoga Early Developer Releases	0.2-0.5	Programadores	Desenho e construção (plataforma e infraestrutura, arquitectura modular e extensível, partilha e colaboração, funcionalidades básicas de email, agenda e calendário).	Início - Abril de 2003 Previsão para versão 0.4 - Octubro de 2004
Canoga End User Releases	0.5-1.0	Utilizador final	Serviços (funcionalidades de gestão de informação, interface com o utilizador, robustez e performance).	Não está definido
Westwood	2.0	Universidades	Facilidades de distribuição (servidores centrais para facilitar a administração e acesso nómado, funcionalidades de calendário com capacidade de interoperabilidade com outras infraestruturas da universidade, segurança e privacidade).	Não está definido
Pasadena	3.0	Pequenas e médias empresas	Suporte a terceiros e costumização (interoperabilidade, migração de outros produtos, framework extensível, distribuição, suporte).	Não está definido

Tabela 2.2: Mapeamento entre nomes de etapas e versões do Chandler

o utilizador da preocupação de organizar a sua informação pessoal. Portanto trata-se, essencialmente, de um sistema gestor de informação pessoal.

É composto por um serviço de agenda virtual onde o utilizador efectua a marcação de eventos e tarefas, ou a marcação de datas especiais em que queira ser notificado. O serviço encarrega-se de lembrar diariamente o utilizador dos compromissos, aniversários, contas para pagar que ele tenha agendado.

Disponibiliza ainda serviços de notificação de informação. Essa informação é seleccionada pelo utilizador e nela podem constar notícias especializadas, propaganda, oportunidades de emprego, horóscopo, cotações de bolsa, previsões meteorológicas, entre outros. O utilizador é notificado via correio electrónico uma única vez por dia com toda a informação seleccionada.

O serviço de agenda pode ser acedido em qualquer lugar onde exista Internet através de um *web browser*, ou via telemóvel através de um WAP *browser*, ou mesmo através de um PDA[35] com sincronização via *HotSync*.

Aplicações comerciais

O *"Elefante"*, uma vez que não é uma plataforma, mas sim um produto comercial, não é propriamente utilizado em outros sistemas. Desde Dezembro de 2001, o *"Elefante"* está a ser gerido pela Invent, consórcio gestora de empresas de Internet do Fundo IP.Com,

[35] *Personal Digital Assistent*

que controla e administra as páginas electrónicas Central de Desejos, Via Global, Olé, O Carteiro, MundoMedia, Comunique-se, Orelha Digital e O Gênio [17].

2.2.6 OutSystems

A empresa *"OutSystems"* [74] é uma empresa portuguesa e representa o produto *"OutSystems Hub Edition"*.

Por se tratar mais de uma *framework* e não de um produto não é tão comparável com os outros sistemas analisados, no entanto, dada as suas características pareceu-me razoável manter a sua análise nesta secção colocando-a em último e fazendo com certeza referência a esse facto.

Este produto suporta todo o processo de desenvolvimento, distribuição e gestão de aplicações para Internet, Intranet ou mesmo móveis e que ligam utilizadores móveis ou fixos a sistemas empresariais existentes e base de dados.

Com esta infra–estrutura as empresas fornecedoras de serviços, conseguem reduzir para metade o tempo de distribuição de uma aplicação e reduzem 30% o tempo de desenvolvimento comparado com soluções tradicionais alternativas, segundo números da empresa.

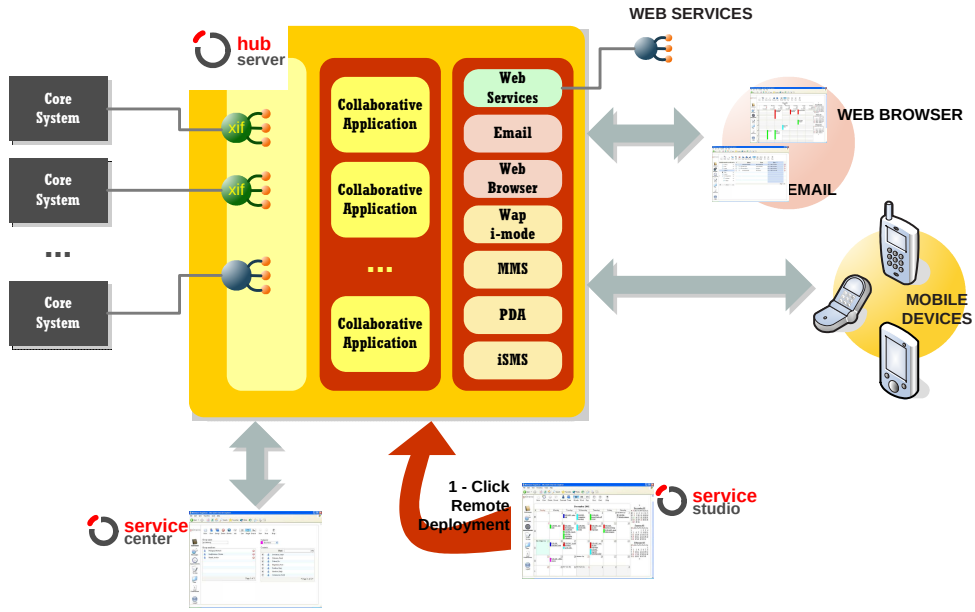

Figura 2.10: Arquitectura do OutSystems Hub Edition (adaptado do original [74])

OutSystems Hub Edition

Segundo a figura 2.10, o *"OutSystems Hub Edition"* junta uma ferramenta de desenvolvimento visual, o *"Service Studio"*, e uma plataforma escalável, o *"Servidor Hub"*.

Este servidor permite ligação com *firewalls*, servidores de correio electrónico, fornecedores SMS[36], servidores de autenticação, WAP *gateways*, entre outros sistemas. Tem a capacidade de servir ainda várias aplicações com contextos diferentes. Disponibiliza para gestão das aplicações uma consola de administração, denominada por *Service Center*.

A plataforma *"OutSystems Hub Server"* está desenvolvida sobre a *framework .NET*, fazendo uso intensivo de standards como por exemplo XML, SOAP[37], e *Web services*. Dentro da tecnologia *wireless* suporta ou faz uso de SMS, MMS[38], WAP, WAP Push, GSM, CDMA[39] e GPRS.

O *"OutSystems Hub Edition"* não é um sistema fechado; permite que programadores possam criar e acrescentar novas livrarias ao sistema já existente. Disponibiliza ferramentas para controlar visualmente estas novas aplicações.

Integração rápida e fácil com sistemas existentes

Sistemas existentes podem recombinar-se com novos serviços que herdam as características da plataforma *Hub Edition*: disponibilidade, eficiência, mecanismos de *caching*, mecanismos de segurança centralizada. O *Hub Edition* importa de uma forma nativa e publica *Web Services* permitindo um modelo de integração rápido e eficiente.

Criação de aplicações Intranet, Internet e Móveis

Uma aplicação pode ser sempre integrada com novos serviços de uma forma simples e eficiente através da ferramenta *Service Studio*. Rapidamente se consegue compor visualmente interface *Web*, de correio electrónico ou móveis, um modelo de dados, lógica de negócio, e políticas de segurança, tudo integrado com sistemas externos.

Conforme já referido, constata-se que o *"OutSystems Hub Edition"* não é um produto final mas sim uma *framework* de desenvolvimento. Ainda assim, verifica-se que com o tipo de serviços que disponibiliza permite criar sistemas com as mesmas características que o que pretendemos criar.

Aplicações comerciais

Sendo um produto português é natural que a maior parte das suas aplicações seja em empresas portuguesas. Esse facto é visível observando a lista de empresas que utilizam este

[36] *Short Message Service*

[37] *Simple Object Access Protocol*

[38] *Multimedia Message Service*

[39] *Code-Division Multiple Access*, tecnologia celular digital que usa técnicas *spread-spectrum*; GSM usa TDMA (*Time-Division Multiple Access*).

produto [75]. Ainda assim não se confina a empresas nacionais sendo também utilizado em produtos de empresas de outras nacionalidades.

- **ANA - Sistema de Gestão de Requisições.** Esta companhia que gere nove dos principais aeroportos de Portugal, servindo mais de 20 milhões de passageiros por ano, usou o *"Outsystems Hub Edition"* para gerar o sistema de gestão de requisições, totalmente integrado com SAP-MM[40] e disponível a toda a companhia através de interface *Web* e correio electrónico. Tudo isto em quatro semanas. Outro sistema também desenvolvido para a mesma empresa com suporte na mesma tecnologia foi o sistema de actualizações de alertas *online*.
- **Brisa - Extensão dos seus sistemas.** É uma das principais gestoras de auto-estradas europeias. Esta empresa constrói e opera mais de 1.000 km de auto-estradas e áreas de serviço em Portugal, e ainda mantém participações em companhias relacionadas no Brasil, Espanha e Itália. No seguimento da estratégia de digitalização de uma colecção de processos, a BRISA seleccionou a *"OutSystems"* como a tecnologia a utilizar para desenvolver aplicações Internet, extranet e móveis.
- **Galp Transgás - Sistema de aprovação e seguimento de facturação e controlo de custos.** Concessionária para a exploração de Gás Natural em Portugal que distribui 10% de energia base no país seleccionou o *"OutSystems"* para implementar o sistema de aprovação de seguimento de facturação e controlo de custos.
- **Grupo EDP - Facilidades de Gestão.** Principal produtor e distribuidor de electricidade em Portugal e um dos maiores operadores de electricidade na Europa. A EDP gere e mantém um conjunto de facilidades de grupo, servindo perto de 10.000 empregados como principais clientes para os seus serviços de manutenção, usando uma aplicação *Web* central totalmente desenvolvida usando o *"OutSystems Hub Edition"*.
- **Telefónica Móviles Corporation - Centro de mensagens.** Um dos produtos do *"OutSystems"*, o Centro de Mensagens *"OutSystems"*, foi seleccionado como a plataforma recomendada para as operadoras sem fios do grupo Telefónica Móviles. Um exemplo foi em Espanha cuja companhia passou a suportar todo um conjunto de tecnologias (MMS, WAP Push, etc) no topo das quais serviços inovadores são facilmente configurados. Outras operadoras actuando em mercados emergentes têm à sua disposição um conjunto completo de serviços de mensagens SMS, EMS[41], MMS, Wap Push, totalmente configuráveis e parametrizáveis à medida das necessidades locais.
- **Optimus - Introdução de novos serviços.** Operadora sem fios em Portugal com

[40] *Systems, Applications, and Products in Data Processing - Materials Management*
[41] *Enhanced Messaging Service*

2.5 milhões de subscritores desenvolveu o seu portal WAP Optimus Zone usando o *"Outsystems Hub Edition"*. Desde a adopção desta plataforma, muitas aplicações foram desenvolvidas disponibilizando serviços de valor acrescentado aos seus clientes Optimus. Geo-SMS e SMS-Pro são apenas alguns exemplos de serviços.

- **Golden Bytes - Suporte ao crescimento num ambiente de rápidas mudanças.** Esta companhia é especializada no desenvolvimento de serviços de dados móveis com um forte foco em serviços Premium SMS. Com 60% do mercado holandês, a Golden Byte está a expandir-se para outras regiões da Europa para consolidar a sua posição de liderança em serviços SMS e serviços media de valor acrescentado (e.g. MMS, WAP Push, e video). Alinhada com esta estratégia de expansão, a Golden Byte decidiu adoptar o *"OutSystems Hub Edition"* como a sua plataforma de desenvolvimento para serviços baseados em SMS, Internet móvel (WAP ou i-mode), MMS e WAP Push.

2.2.7 Análise comparativa

A tabela 2.3 reúne as características encontradas em cada uma das aplicações analisadas. No eixo das abcissas estão enumeradas as aplicações analisadas; no eixo das ordenadas estão identificados tópicos para cada uma das características principais PUC e PIM; ainda estão identificadas algumas características que, por não se enquadrarem no conceito PUC ou PIM, foram reunidas em tópicos genéricos.

			Outlook	Elefante	OutSystems	Weblicon	Hambo	Chandler
Funcionalidades	**PIM**	Agendas	x	x		x	x	x
		Contactos	x			x	x	x
		Tarefas	x	x		x	x	x
		Notas	x					
	PUC	Email	x		x	x	x	x
		IM					x	
		SMS			x	x	x	
		MMS			x	x		
		VoIP				x		
Outros Aspectos		Interfaces	PC	browser, WAP	browser, WAP	browser, WAP, j2me, i-mode, voiceXML	browser, WAP, i-mode, voiceXML	Não está claro
		Web services			x	x		
		Colaboração	x					x
		Notificação	local	email, SMS			email, SMS	
		Plataforma	Windows		.Net	j2ee	Java	Python, wxWindows
		Código livre					x	x
		Sincronização	x	HotSync		SyncML	x	x

Tabela 2.3: Características das aplicações

Pode-se dividir a comparação das diferentes aplicações em quatro categorias: (1) segundo o conceito de gestão de informação pessoal (PIM); (2) segundo o conceito de comunicações pessoais (PUC); (3) segundo características diversas; e ainda (4) comparar segundo uma perspectiva global. É de ter em conta que se está a comparar sistemas que nem sempre servem os mesmos objectivos, ou seja, enquanto umas servem de suporte a outros sistemas, nomeadamente o *"OutSystems"*, *"Weblincon"*, *"Hambo"* e *"Chandler"*; outros são produtos finais como é o caso do sistema *"Elefante"* e do *"Outlook"* da Microsoft. O *"Hambo"* tem a particularidade de apresentar serviços prontos a serem utilizados por utilizadores finais; na minha perspectiva, é mais um conjunto de serviços pronto a ser integrado num sistema global, do que um sistema independente. Numa perspectiva de serviços podemos comparar todos de forma igual porque todos são prestadores de serviços.

Observando quanto à componente de gestão de informação verificamos que praticamente todos, excepto o *"OutSystems"*, oferecem o mesmo tipo de serviços. O mais completo é o *"Outlook"* por apresentar ainda o serviço "notas"; no entanto, e segundo a minha perspectiva, nos moldes actuais não me parece de grande utilidade (por exemplo não é dada a hipótese de configuração para a nota ficar sempre no topo proporcionando como possível consequência o esquecimento do utilizador). O *"OutSystems"* está sem dúvida mais vocacionado para as comunicações, uma vez que não apresenta de base qualquer serviço no contexto de gestão de informação.

Analisando na perspectiva de comunicações pessoais verificamos que o sistema mais completo é o *"Weblicon"*. O único tipo de comunicação que não disponibiliza, dentro dos analisados, é a comunicação por mensagens instantâneas (na realidade quase nenhum sistema apresenta esta funcionalidade; a única excepção é o sistema *"Hambo"*). A razão para esta situação, até porque se trata de um serviço com bastante aceitação actualmente, é a difícil implementação em aplicações com interface *Web*. O sistema que não oferece funcionalidades de comunicações pessoais é o *"Elefante"*.

Quanto às características genéricas existem diferentes tópicos que se podem discutir individualmente. Em relação às interfaces disponibilizadas para interacção do utilizador com o sistema, no que diz respeito ao *"Chandler"*, o facto de aparecer na tabela que "não está claro" não quer dizer que não disponibilize; acontece que como ainda está em fase de desenvolvimento, e segundo as suas etapas (ver secção 2.2.4), ainda não foi atingida aquela onde pretendem disponibilizar interfaces com o utilizador e por consequência não apareceu na documentação investigada qualquer referência ao tipo de interface a desenvolver. As aplicações que são mais ricas nas interfaces são a *"Weblicon"* e a *"Hambo"*. Quanto à característica *Web services* verifica-se que está presente nos sistemas mais vocacionados

para dar suporte à criação de outros sistemas. Na notificação, o único que teria capacidade para realizar notificação instantânea é o sistema *"Hambo"* por disponibilizar o serviço de mensagens instantâneas. No entanto, pela documentação investigada, a notificação é realizada por correio electrónico ou por SMS. Perderam uma boa oportunidade de apresentar alguma inovação na área das notificações. No sistema *"Elefante"* a notificação é realizada de igual forma através do correio electrónico ou SMS. No *"Outlook"* a acção de notificar é local onde se realizou o pedido de notificação. Uma das características que poderá trazer maiores inovações a estes tipos de serviços, a colaboração, os únicos a fazerem referência são o *"Outlook"* e o *"Chandler"*. Com alguma infelicidade (porque acredito na iniciativa código livre), verificamos que apenas apostam em código livre o *"Hambo"* e o *"Chandler"*. Praticamente todos disponibilizam mecanismos de sincronização.

Fazendo algumas considerações numa perspectiva global, considero que o *"Hambo"* foi descontinuado, ou pelo menos não tem havido grande investimento neste produto. Digo isto porque no portal onde é disponibilizado [36], uma vez que é código livre, a última actualização corresponde a 2002, e o facto de também não estarem divulgadas parcerias com outras empresas ajuda a validar esta observação. Das restantes aplicações, o *"Outlook"* peca por corresponder a uma instalação local obrigando a uma série de configurações para guardar a informação num servidor remoto. Todas estas acções não são naturais ao utilizador. Dada a sua massiva utilização, todos os dispositivos têm uma instalação local do *"Outlook"*, ou uma versão mínima (dependendo do dispositivo), ou simplesmente uma aplicação onde todos sabem sincronizar a informação entre os diferentes repositórios. Dos vários sistemas analisados, o mais completo é o disponibilizado pela *"Weblicon"*. Este será o sistema mais próximo que pretendemos atingir no que diz respeito à disponibilização de serviços e de interfaces com o utilizador. No entanto a forma e a estratégia adoptada no desenvolvimento e distribuição utilizado no *"Chandler"* está mais próximo da forma como o nosso sistema vai ser (ou foi, dada a altura da escrita desta tese) desenvolvido.

Ao nível das utilizações comerciais, há umas aplicações que não se podem comparar com outras, uma vez, como já referido, umas são aplicações finais disponibilizadas ao público, e outras são sistemas de suporte à criação de outros sistemas. Nesta última vertente verificámos que tanto a *"Weblicon"* como o *"OutSystems"* são bastantes utilizados no mercado actual. O primeiro com uma grande utilização na Europa, com principal incidência na Alemanha (com alguma naturalidade sendo o país onde nasceu a *"Weblicon"*). É também utilizado no Estados Unidos; o segundo, por ser uma empresa nacional, evidentemente tem uma utilização com maior incidência em Portugal. Contudo, não se limita ao nosso país.

Constata-se pelas empresas envolvidas que potenciais clientes de uma plataforma deste

tipo são operadoras de telecomunicações, uma vez que estamos a falar de serviços relacionados com comunicações e estamos a falar de serviços potencialmente apelativos, inovadores que às operadoras interessará no sentido de manter satisfeitos os seus clientes e, melhor ainda, conseguir angariar novos clientes. No entanto, um sistema deste tipo não quer dizer que seja de uso exclusivo para operadoras. Há possibilidade de existir parcerias: por exemplo, uma empresa tem o sistema e repositório de informação e tem parceria com uma operadora de telecomunicações, por exemplo para garantir os serviços de comunicação.

No capítulo 5, enquadrado em trabalho futuro, faz-se referência novamente à tabela 2.3, onde se inclui na comparação das aplicações analisadas, o sistema concebido e implementado no âmbito deste nosso projecto (ver tabela 5.1).

Capítulo 3

Análise do Sistema

3.1 Visão geral

O objectivo principal deste trabalho é potenciar a criação de um sistema que reúna as qualidades encontradas de forma dispersa nos sistemas analisados no capítulo 2, secção 2.2. Quando se fala em "reunir qualidades" corresponde a reunir funcionalidades, formas de disponibilizar os serviços, formas de criar e desenvolver o próprio sistema, a arquitectura e plataforma de suporte do mesmo. Ao referir sistema não é pretensão estar a referir a um produto acabado, pronto a ser usado. Quando se refere sistema está-se a falar mais de uma plataforma de serviços, onde estes possam ser facilmente integrados de forma a criar novos serviços. Por exemplo na notificação de um encontro marcado na agenda: essa notificação pode ser realizada por mensagem instantânea, se o utilizador estiver *online*, ou por recepção de SMS, se *offline*.

Conforme já se teve oportunidade de verificar nos capítulos anteriores quando se refere "serviços" estamos a pensar em duas grandes áreas: serviços relacionados com a gestão de informação pessoal (PIM) e serviços de comunicações pessoais (PUC). Por serviços de gestão de informação pessoal falamos por exemplo de serviços do tipo de gestão de agenda e ou de contactos (estes são aqueles que se encontram mais divulgados). Por serviços de comunicações pessoais entendamos serviços do tipo correio electrónico, mensagens instantâneas, SMS, etc. Segundo a secção 2.2 verifica-se que existe actualmente já uma série de aplicações que disponibilizam este tipo de serviços. Umas aplicações estão mais vocacionados para uma área; outras estão mais vocacionados para outra; outras, porém, disponibilizam serviços em ambas as áreas (e.g., *"Weblicon"*). Podemos então constatar que os conceitos PUC e PIM, por si só, e vistos de forma independente, não são novidade. A novidade poderá advir da criação de novos serviços apelativos e que satisfaçam a necessidade e exigência dos utilizadores através da integração destes dois

grandes conceitos.

Para além da criação de novos serviços expectáveis já pelo utilizador deve-se introduzir serviços conceptualmente novos. Por exemplo, o sistema *"Chandler"* [6] é um exemplo interessante a seguir. No que diz respeito a gestão de informação pessoal, propõem organizar a informação através de critérios do utilizador e não através de critérios do sistema. Desta forma conseguem apresentar ao utilizador toda a informação relevante segundo o critério por ele definido, mesmo que essa informação seja oriunda de diferentes tipos (e.g., mensagem de correio electrónico, vídeo, MP3, foto, etc) (ver secção 2.2.4). Um outro conceito também referido, que no meu entender potencia a introdução de novos serviços conceptuais, são serviços baseados na partilha de informação e colaboração entre diferentes intervenientes. Estes dois conceitos facilitam trabalho em equipa, fundamental nos dias actuais.

Uma outra característica que se pretende presente no sistema a desenvolver é a introdução de **contexto** no nosso sistema. Este conceito foi introduzido no capítulo 1. Não é um conceito recente mas ainda gera muita discussão e expectativas [13, 79, 76]. Aplicações que tenham interacção com o utilizador e que disponibilizem funcionalidades baseadas em contexto são aplicações potencialmente mais apelativas do que se não o fizerem. Todavia, contexto é suficientemente vago para poder ser tudo! Por exemplo duas pessoas em diálogo. Esse diálogo pode ser influenciado por diversos e inúmeros factores. Por exemplo, se as duas pessoas já se conhecem, onde é que se encontram, a disposição emocional ou física de cada uma delas, enfim, um sem número de condições que podem alterar o resultado da conversa. De alguma forma, é importante introduzir e ou simular estes aspectos num diálogo entre o utilizador e o computador. Para facilitar esta tarefa, foram definidas e aceites ao longo do tempo várias categorias de contexto [13], as quais são conhecidas pelos 5 *W's*: *Where, Who, When, What,* e *HoW*. Ou seja, se uma aplicação responder a cada uma das cinco categorias mencionadas então poder-se-á dizer que a aplicação é baseada em contexto (*"context awareness"*). Basta responder a uma das categorias para se poder dizer que é baseada em contexto dentro dessa categoria.

No nosso sistema podemos conceber as características que deverão existir para se poder dizer que é baseado em contexto. À partida será suficiente arranjar serviços que respondam às cinco categorias que caracterizam contexto. A tabela 3.1 exemplifica serviços e ou funcionalidades possíveis para cada uma das categorias.

Um outro problema será como introduzir contexto na aplicação. Várias possibilidades foram referidas em [13]. De forma resumida o contexto poderá ser dado pelo utilizador de uma forma manual (menos agradável) ou de uma forma automatizada se a aplicação tiver capacidade para isso (e.g., captar das acções do utilizador e as suas preferências).

Categoria de contexto	Exemplos de serviços
Localização (Where)	A aplicação notifica o utilizador através de SMS, por exemplo, da proximidade de um restaurante com determinada especialidade gastronómica, ou da proximidade de uma farmácia em serviço (não foi considerado o problema da determinação da localização do utilizador)
Quem (Who)	A aplicação pode ceder mais ou menos privilégios função da identificação do próprio utilizador
Quando (When)	O utilizador marca na sua agenda um apontamento com notificação. O processo de notificação corresponde a um serviço com a característica "quando"
Quê (What)	O utilizador está a preencher um formulário. A aplicação pode validar durante o preenchimento os dados inseridos pelo utilizador
Como (hoW)	Na recepção de alertas: se estiver *online* recebe via mensagem instantânea; se estiver *offline* recebe via correio electrónico ou via SMS

Tabela 3.1: Exemplos de serviços baseados em contexto

Este é sem dúvida um grande problema que corresponde à captação e filtração do que é contexto. Existem muitos trabalhos nesta área, nomeadamente no desenvolvimento de *frameworks* e técnicas que facilitem a captação de contexto sem intervenção directa do utilizador [44, 41, 40, 43, 39, 38, 42].

Falando no nosso sistema de uma forma mais concreta, a figura 3.1 ilustra a visão geral do mesmo. O nome dado ao projecto foi PUC (*Personal Unified Communication*), porque a sua primeira versão focava-se na disponibilização de serviços de comunicação (que corresponde ao âmbito prático desta tese). Posteriormente aconteceram outros trabalhos relacionados e que integraram componentes de gestão de informação pessoal. Nessa altura poder-se-ía ter arranjado um novo nome para o sistema mais coerente com o que representava (integração de serviços de comunicações com informação pessoal), mas por uma questão histórica manteve-se o nome PUC (o enquadramento deste trabalho e dos restantes está descrito na secção 1.2). O sistema é constituído conceptualmente por um nó central, designado por "Servidor Aplicacional PUC" (que integra diferentes servidores de recursos incluindo Servidor LDAP, Servidor Correio Electrónico, Servidor Jabber, Servidor Media VoiceXML) e por vários nós, de diferentes tipos, que desempenham o papel de terminal de acesso. Entre outros, podemos identificar os seguintes tipos de terminais e tipos de interface homem–máquina: PC com interface *Web*; telemóveis com interface WAP ou interface Windows (e.g., baseado no J2ME); ou telefones inteligentes.

O nó "Servidor PUC" executa os componentes de software fundamentais com perfil servidor, designadamente os componentes correspondentes aos serviços disponibilizados (e.g., Serviço-1, Serviço-2, ... Serviço-N). Note-se que o "Servidor PUC" representado apenas por um nó, por motivos de simplicidade, é na realidade concretizado por vários nós distribuídos de modo a garantir a escalabilidade e desempenho do sistema correspondente ao princípio de suporte de um "elevado número de utilizadores e de transacções".

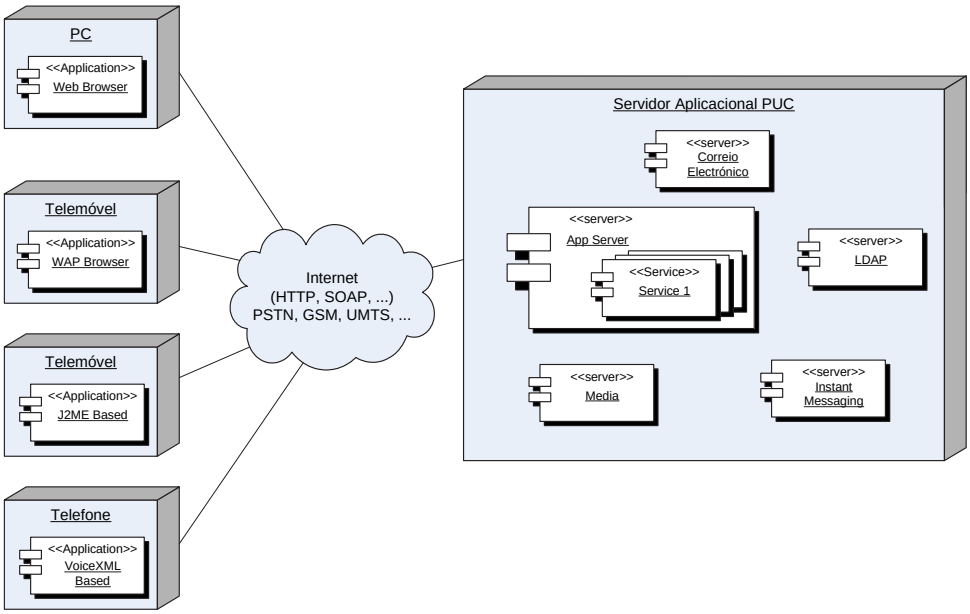

Figura 3.1: Visão geral do sistema PUC (adaptado do original [11])

Por outro lado, de forma a satisfazer o princípio da "ubiquidade de tipo de acesso e de tipo de media", deve permitir-se o acesso ao sistema a partir de vários terminais, os quais podem executar componentes pré-instaladas e específicas do sistema PUC (e.g., componente J2ME instalada nos telemóveis) ou apenas componentes gerais e não específicas (e.g., um *Web/Wap browser* instalado num PC ou num telemóvel).

Num sistema deste tipo é exigido que um utilizador tenha que realizar um registo antes de poder usufruir dos serviços disponíveis. Uma vez registado, o utilizador só precisa autenticar-se perante o sistema sempre que pretenda ter acesso ao mesmo.

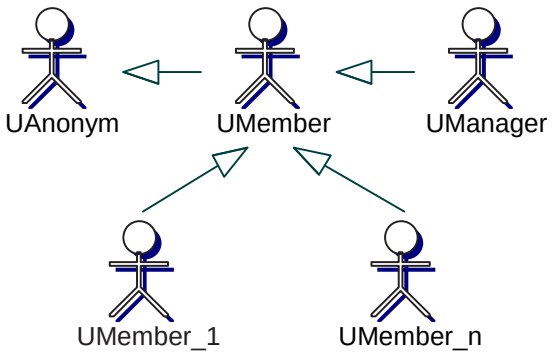

Figura 3.2: Hierarquia de utilizadores

A figura 3.2 mostra a hierarquia de actores: anónimo, membro e gestor. Um utilizador "anónimo" tem com certeza um raio de acções limitado no sistema não podendo usufruir

dos serviços disponíveis. Um utilizador "membro" teve que passar previamente pela fase de registo. Nesse estado já tem conta no sistema e poderá utilizar os serviços disponíveis. Reparar na figura que existe a noção de vários tipos de utilizadores "membro". Isto significa que poderão existir diferentes tipos de privilégios, função do registo efectuado. Em qualquer altura deverá ser dada a possibilidade de configurar e alterar os privilégios de acesso ao sistema. O utilizador "gestor" é aquele que tem privilégios de administração do sistema. Deverá ser ele o responsável pela gestão do registo de novos utilizadores, novos serviços, facturação, etc.

3.2 Princípios

A concepção e o desenvolvimento do sistema PUC são orientados por um conjunto restrito de princípios, os quais se referem de seguida.

3.2.1 Abstracção das tecnologias de rede

O PUC procura seguir os princípios conceptuais das redes de próxima geração, em particular a utilização de interfaces abertas de programação que abstraiam o programador das tecnologias de rede subjacentes. Em termos de arquitectura funcional estas interfaces são oferecidas por elementos de rede a que se designa "Servidor de Aplicações de Telecomunicações".

3.2.2 Ubiquidade de tipo de acesso e de tipo de media

O sistema deve poder ser acedido através de distintos tipos de terminais (e.g., telefone, telefone inteligente, PDA, PC) e segundo diferentes tipos de interacção homem–máquina (e.g., interacções baseadas em interface voz, Web, WAP, menus e ecrãs de telemóveis). Note-se no entanto, que nem todos os casos de utilização poderão ser suportados integralmente ao nível dos vários tipos de terminais e de acessos. Por exemplo num telefone onde a única interface física é o teclado dificilmente o utilizador conseguirá ter acesso ao mesmo número de serviços e funcionalidades que conseguirá ter com uma interface mais rica como é, por exemplo, a interface de um PC. Na realidade, até poderá não ser impossível; o que acontece é que a interface poderá ser tão complicada que deixará de ter sentido existir.

3.2.3 Elevado número de utilizadores e de transacções

O sistema deve suportar a subscrição e utilização de milhares ou milhões de utilizadores. Ou seja, o sistema deverá ser escalável, em termos do número de utilizadores e de

transacções, mantendo tempos de resposta adequados aos parâmetros de qualidade exigidos. Adicionalmente, tendo em conta a dimensão de utilizadores envolvida, o sistema deverá apresentar disponibilidade e tolerância a faltas próxima de 100%.

3.2.4 Gestão e subscrição flexível e dinâmica de serviços

O sistema deve permitir a gestão (e.g., introdução, suspensão, eliminação) de serviços pessoais de forma fácil e dinâmica. Deve ainda suportar a gestão de versões de forma a garantir e controlar a evolução graciosa dos serviços, i.e., sem forçar paragens do sistema.

Por outro lado, o sistema deve permitir aos seus utilizadores a gestão flexível e também dinâmica da subscrição dos serviços disponíveis. Neste âmbito deve ser suportado o conceito de "conta de utilizador global" no sistema (de forma que o utilizador não tenha de guardar e manter várias *passwords* conforme o número de serviços subscritos) bem como "conta de cliente global" (de forma que o sistema de *billing* emita apenas uma única factura independentemente dos serviços subscritos).

3.2.5 Tecnologia Java e Open-Source

O sistema deve ser desenvolvido em Java, sobre as suas várias tecnologias emergentes, particularmente agregadas às especificações J2EE [31] e J2ME [33]. Adicionalmente, de forma a minimizar a "reinvenção da roda" tanto quanto possível e a acelerar o desenvolvimento do projecto, poderão ser usados e integrados projectos Java, desde que desenvolvidos por terceiras partes segundo a filosofia do software livre (i.e., *open-source*) [70, 77].

3.3 Visão dos principais serviços

Tendo em conta o princípio de "gestão e subscrição flexível e dinâmica de serviços" o sistema deverá suportar com flexibilidade quer a gestão de serviços (i.e. registo de novo serviço, suspensão, evolução para nova versão, remoção) quer a sua subscrição.

A operação de gestão de serviços, sendo crítica, deverá ser realizada apenas e exclusivamente pelos administradores e gestores do PUC, i.e., por empregados da empresa que operem o sistema. Por outro lado, a subscrição de serviços é realizada pelos utilizadores/-subscritores, na medida que estes poderão alterar de forma fácil e flexível os serviços que pretendem e posteriormente usá-los de forma integrada com todos os restantes já subscritos. Entre outros aspectos, deve ser suportado de forma independente dos serviços subscritos, os requisitos de "conta de utilizador (*login* e *password*) única" e de "factura (*billing*) única".

3.3.1 myComs - Serviço Pessoal de Comunicações

O serviço *"myComs"* (serviço pessoal de comunicações) tem como objectivo oferecer aos seus utilizadores um leque alargado e integrado de recursos e serviços de comunicação. Conforme ilustrado na figura 3.3, os serviços variam desde o serviço de correio electrónico (*email*), até ao de mensagens instantâneas e de presença (IM, *Instant Messaging*) passando pelo serviço de comunicação tradicional de telefonia, baseado em voz, ou pelo suporte de espaços *Web* pessoais e possibilidade de gestão de *bookmarks* e preferências.

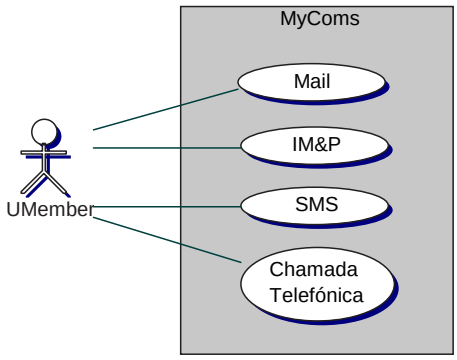

Figura 3.3: Caso de uso de comunicações

3.3.2 myContacts - Serviço Pessoal de Contactos

O serviço *"myContacts"* (serviço pessoal de contactos) tem como objectivo permitir aos seus utilizadores a gestão de contactos, os quais podem corresponder a contactos individuais e ou institucionais/empresariais. Existirão mecanismos avançados de referência dinâmica a contactos públicos mantidos por outros utilizadores ou pela própria empresa operadora do PUC. Desta forma, podem-se estabelecer ligações virtuais a contactos mantidos por entidades terceiras, com a garantia de que quando a informação correspondente é alterada (e.g., alteração de um número de telefone) todos os utilizadores com essas referências virtuais têm acesso à informação mais actualizada. Adicionalmente, existirão mecanismos de sincronização entre os contactos mantidos centralmente no servidor PUC e os mantidos localmente num telemóvel pessoal.

A figura 3.4 ilustra os casos de utilização básicos relacionados com os contactos, nomeadamente a componente de gestão de contactos e a componente de gestão de grupos de contactos. Existindo a noção de grupo de contactos facilita a alteração e configuração dos privilégios atribuídos a determinado grupo.

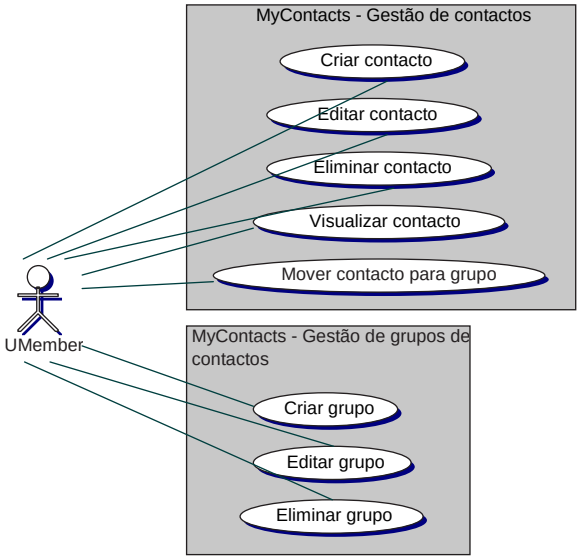

Figura 3.4: Caso de uso de contactos (adaptado do original [35])

3.3.3 myAgenda - Serviço Pessoal de Agenda

O serviço *"myAgenda"* (serviço pessoal de agenda) tem como objectivo permitir aos seus utilizadores a gestão de agenda, com as operações clássicas de marcação de reuniões, compromissos, eventos, etc. De forma a suportar a realização de algumas operações de forma colaborativa (e.g., marcação de reunião), o dono de cada agenda poderá conceder privilégios de acesso e mesmo alteração da sua agenda a outros utilizadores definidos em grupos especiais para o efeito (por exemplo, poderá permitir que os utilizadores definidos no seu grupo "Secretárias" possam ter acesso à sua agenda e fazer quaisquer marcações no período livre dos dias úteis, entre as 9h e as 17h.). À semelhança com o serviço *"myContacts"*, existirão mecanismos de sincronização entre a agenda mantida centralmente no servidor PUC e a mantida localmente num telemóvel pessoal.

A noção de agenda implica inevitavelmente a existência de calendário e funcionalidades relacionadas, conforme se pode verificar na figura 3.5. Na componente do calendário verifica-se ainda a ponte entre o serviço *"myAgenda"* e o serviço *"myContacts"* através, por exemplo, do caso de utilização de "associar contacto a evento".

3.3.4 Tipos de utilizadores

Uma vez apresentado os principais serviços, e o facto de existir uma hierarquia de utilizadores, deve-se mostrar, função do tipo de utilizador, quais os privilégios que tem e quais as acções que pode realizar no sistema.

O sistema PUC é utilizado por diferentes utilizadores, cada qual associado a um determinado perfil de utilização, designados por actores. Estão previstos três tipos de actores:

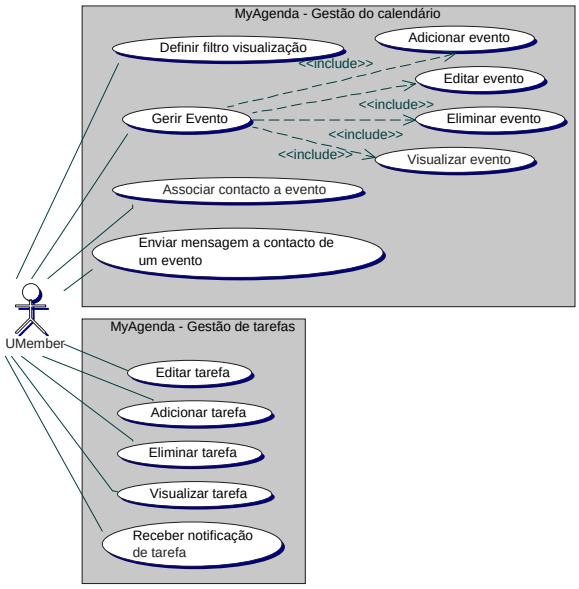

Figura 3.5: Caso de uso de agenda (adaptado do original [35])

anónimo, membro, gestor, conforme se pode ver na figura 3.2.

O utilizador "anónimo" tem um acesso limitado, normalmente podendo consultar informação geral do sistema e/ou solicitar a subscrição do serviço, conforme ilustra a figura 3.6.

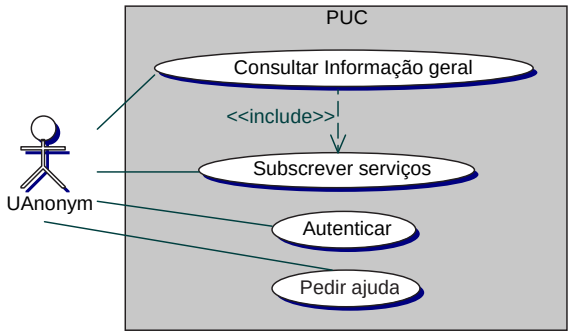

Figura 3.6: Caso de uso do utilizador anónimo

Conforme se pode verificar na figura 3.7, o utilizador "membro", após autenticação, tem um acesso completo às funcionalidades disponíveis. Neste âmbito pode ainda existir algum controle de acesso mediante o tipo de subscrição de serviços que fez. A figura 3.2, na secção 3.1, reflecte exactamente esta característica com a noção de hierarquia de utilizadores onde está previsto a existência de vários tipos de membros. Para cada tipo de membro poderão existir diferentes privilégios no acesso aos serviços do sistema. No protótipo implementado um utilizador autenticado tem acesso a todos os serviços.

Por fim o utilizador "gestor" tem acesso completo à gestão de contas dos membros

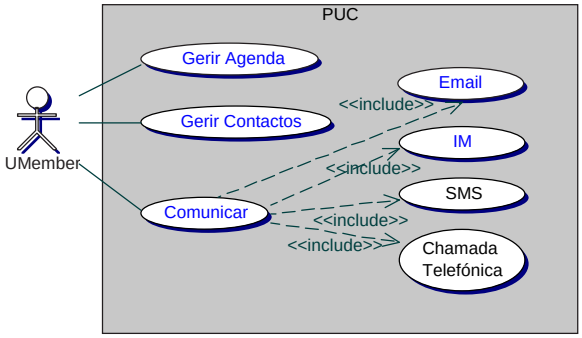

Figura 3.7: Caso de uso do utilizador membro

do sistema; em caso algum, o gestor poderá ter acesso directo às contas dos membros de modo a garantir a privacidade do serviço. Neste protótipo não existe implementado o perfil de gestor. Ainda assim a figura 3.8 reflecte as acções possíveis de realizar para um utilizador com perfil de gestor.

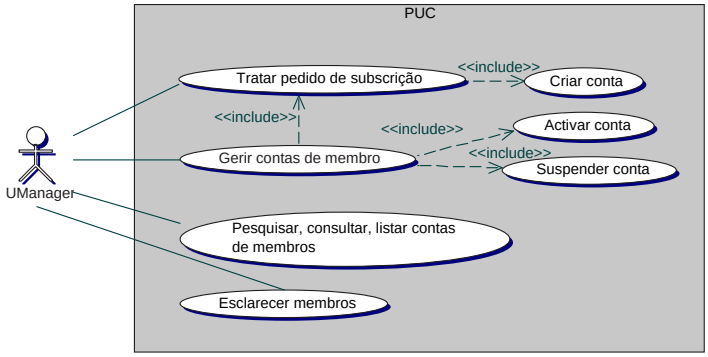

Figura 3.8: Caso de uso do utilizador gestor (adaptado do original [35])

3.4 Serviço pessoal de comunicações myComs

No âmbito do trabalho prático desta investigação, focou-se no desenho e implementação do serviço *"myComs"* (ver capítulo 4). Todavia, neste capítulo descreve-se com maior abrangência as funcionalidades desejáveis para o sistema; algumas das quais foram implementadas e outras a merecer maior aprofundamento e trabalho futuro.

Dentro das comunicações pessoais foram eleitos alguns serviços considerados de maior utilização na actualidade: (1) serviço de correio electrónico; (2) serviço de mensagens instantâneas e notificação de presença; (3) considerado também um serviço, fundamentado nas próximas secções, o serviço de subscrição e autenticação.

Já referido, o *"myComs"* é um serviço pessoal de comunicações. Nesse sentido a ideia é integrar um leque variado de sub serviços na área das telecomunicações, nomeadamente

serviço de correio electrónico, serviço de mensagens instantâneas e notificações de presença, serviços relacionados com voz sobre IP, entre outros. Num cenário actual, para cada um destes serviços o utilizador necessita de instalar localmente uma aplicação para permitir o acesso ao respectivo serviço, precisa configurar essa aplicação, precisa registar-se perante o serviço, precisa manter contas para cada serviço, enfim... uma série de actividades que o utilizador precisa de realizar, quando o seu objectivo inicial era apenas aceder um determinado serviço. A integração de serviços vem permitir que o utilizador se autentique uma única vez para conseguir aceder a todos os serviços a que tem direito.

Outra característica do *"myComs"* é que os serviços são disponibilizados via *Web*. Esta particularidade permite minimizar qualquer tipo de intervenção no lado do utilizador. Em última instância o utilizador precisará apenas de um *browser* para aceder aos serviços. Tem igualmente a vantagem das actualizações dos serviços serem totalmente transparentes para o utilizador final.

3.4.1 Serviço subscrição

Um utilizador para se tornar membro do *"myComs"*, e com isso usufruir de todos os serviços disponibilizados precisa de se registar no sistema, ou seja, de realizar a subscrição. Neste sentido a subscrição foi entendida como mais um serviço do *"myComs"*. Isso reflecte-se logo na arquitectura deste serviço que segue o modelo adoptado para todos os restantes serviços. Para pormenores sobre a arquitectura do serviço ou implementação remeter à secção 4.2.1 e 4.3.1. Segundo a figura 3.9 a subscrição, a nível funcional, não é mais do que recolher os dados do utilizador através de um conjunto de formulários, validar esses dados, e criar uma nova entrada no servidor de registos do utilizador (embora não implementado, terá também o objectivo de criar as demais contas necessárias nos diferentes servidores para que os demais serviços possam ser disponibilizados).

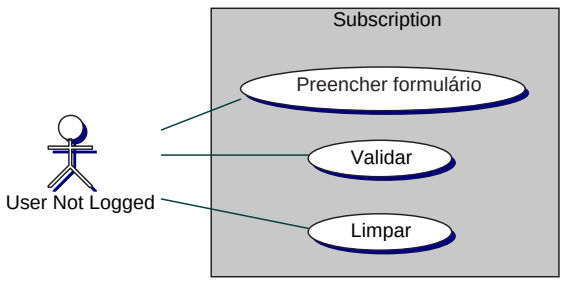

Figura 3.9: Caso de estudo do serviço de subscrição

3.4.2 Serviço correio electrónico

Um serviço de *webmail* não é mais do que um serviço que permite ao utilizador aceder ao seu correio electrónico através de um *browser*. O serviço de correio electrónico deverá permitir aos seus utilizadores várias funcionalidades, de entre as quais se destacam: a possibilidade de envio e recepção de mensagens de correio electrónico com ou sem *attachement*; a gestão de caixas de correio (e.g., segundo o paradigma de pastas hierárquicas) - com esta funcionalidade o utilizador consegue criar novas pastas, remover pastas, apagar mensagens, mover mensagens entre pastas; e a gestão de contactos e endereços electrónicos. Esta última funcionalidade sugere as fortes dependências entre os vários serviços do PUC, nomeadamente entre o *"myComs"* e o *"myContacts"*.

A figura 3.10 reflecte as acções possíveis de um utilizador depois de se autenticar no sistema, no que diz respeito ao serviço de correio electrónico. Pode-se desde já verificar que em relação ao serviço de correio electrónico o utilizador tem um leque alargado de acções que pode realizar: desde a parte de visualização até à parte de gestão passando pela parte de envio.

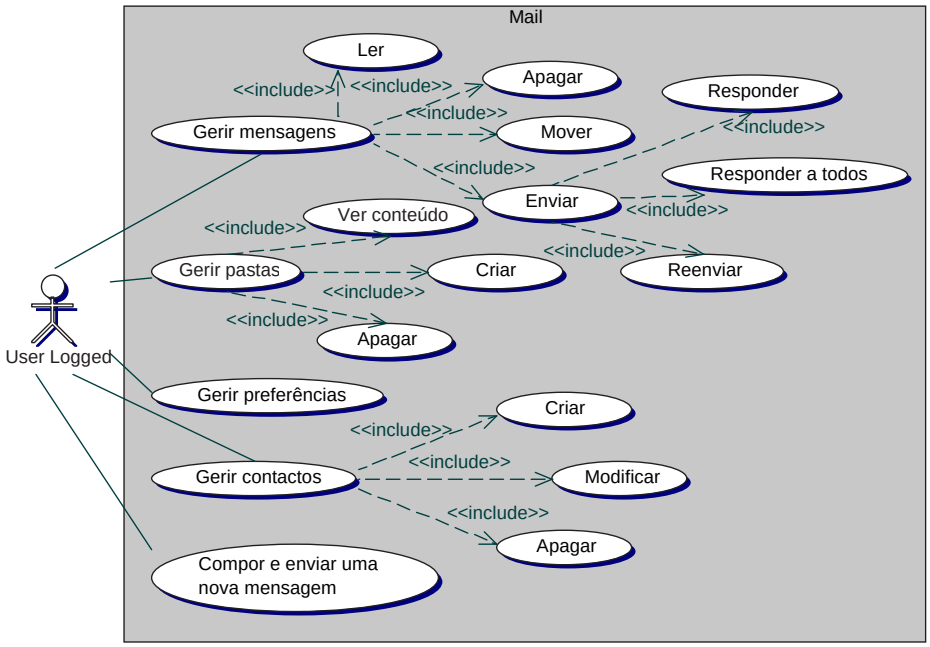

Figura 3.10: Caso de estudo do serviço de correio electrónico

3.4.3 Serviço de mensagens instantâneas e de presença

O serviço de mensagens instantâneas e presença é um serviço em tudo semelhante aos serviços proprietários conhecidos da Microsoft (Messenger) [58], I Seek You (ICQ) [37], da AOL (AIM) [3], entre outros. Entre as várias funcionalidades permite: alterar o estado de

presença do utilizador; manter uma lista de amigos com a possibilidade de estar organizada por grupos e com o estado de presença de cada um; enviar mensagens instantâneas para um amigo cujo estado de presença seja presente. Uma das mais valias deste serviço é ser mais um serviço disponibilizado numa interface *Web*, requerendo ao utilizador, no máximo, um *browser*. É uma mais valia devido à dificuldade da sua implementação porque, ao contrário de uma aplicação *Web* normal em que por norma a aplicação se baseia em pedido/resposta, aqui, neste serviço, as notificações são assíncronas. As soluções possíveis e adoptadas estão detalhadas na secção 4.4.3.

A figura 3.11 ilustra mais uma vez as acções possíveis no serviço de IM de um utilizador que já se autenticou perante o sistema. Em relação ao serviço de correio electrónico tem um número de acções limitado disponibilizando apenas o essencial: receber mensagens e notificações de presença; enviar mensagens e alterar estado de presença. Tal facto deve-se a ter sido um serviço desenvolvido de raiz; ao contrário do serviço de correio electrónico que tinha já previsto todas as funcionalidades referidas. Outras funcionalidades que ficariam bem neste serviço seriam a nível de gestão da lista de amigos, nomeadamente: inserir/remover/modificar/mover "amigos"; inserir/remover/modificar/mover grupo; bloquear notificações de determinados "amigos"; entre outras funcionalidades. Dado o suporte a este serviço ser o protocolo *"Jabber"*, no fundo seria observar as extensões ao protocolo e implementar as mais razoáveis. A implementação destas funcionalidades terá que ser encarada como trabalho futuro.

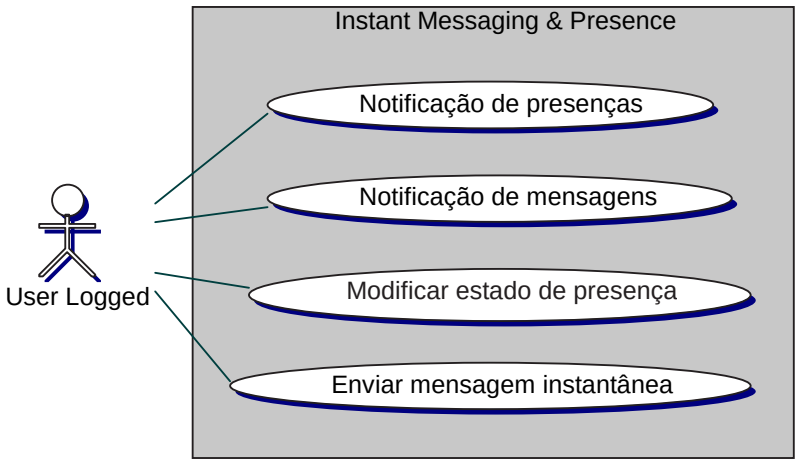

Figura 3.11: Caso de estudo do serviço de mensagens instantâneas e notificação de presença

3.5 Visão de outros serviços

A secção 3.3 identifica os principais serviços disponibilizados no sistema PUC. Esses serviços (comunicações, agenda e contactos), por si só, não são novidade; vimos vários exemplos desses serviços na secção 2.2. No entanto têm que existir porque são os alicerces para a criação de outros serviços, em que esses sim, são potencialmente inovadores.

Um exemplo para permitir a criação de novos serviços é a integração no sistema do conceito de **sensores**. Um "sensor" pode ser entendido como um recurso hardware ou mecânico capaz de captar informação, dados; pode ser entendido também como um serviço que encapsula informação e coloca-a disponível. É neste segundo sentido que nos interessa a utilização de sensores. Com sensores tem-se a capacidade de caracterizar o ambiente que nos rodeia que pode ser aproveitado no sentido de permitir o sistema tomar decisões baseadas em contexto.

Para obter informação que seja utilizada como contexto pode-se pensar em sensores do tipo GPS para determinação da localização; sensores de temperatura, pressão ou humidade para determinação do meio ambiente onde o utilizador se encontra. Enfim, praticamente qualquer sensor pode produzir informação que pode ser utilizada para determinação do meio envolvente do utilizador. Mais, extrapolando para um futuro não muito longínquo, e conforme sugerido em [54], vários objectos físicos poderão ter **identificadores únicos** (através por exemplo de RFIDs[1]), podendo-se estabelecer novos padrões de interacção. Ou seja, toda a informação estará disponível para poder criar serviços tão diferentes quanto inimagináveis nos dias actuais. Claro que toda esta informação acarretará problemas ao nível da segurança, da privacidade, etc. (No entanto, estes aspectos não são alvo deste trabalho.)

Outro conceito a integrar no sistema é a **gestão de informação**. A este nível pode-se pensar em características tão diversas como por exemplo, a partilha de informação, a gestão de grupos, gestão de privilégios, ou outra característica tão premente como é a colaboração. Estas características estão de alguma forma relacionadas entre si. A partilha de informação significa que várias entidades têm acesso a um mesmo repositório de informação. Tem que existir igualmente a noção de **gestão de grupos**, com a necessidade de os criar, gerir, atribuir-lhes diferentes tipos de privilégios, etc. Por sua vez, a partilha exige um forte controle de acessos, ter a capacidade de atribuir diferentes tipos de acesso e privilégios função da entidade. Por último, para existir colaboração tem que existir igualmente a capacidade de definir **grupos de colaboração**, de especificar em que colaborar, ter a capacidade de gestão e definir critérios de decisão. Alguns destes

[1] *Radio Frequency IDentifiers*

conceitos estão presentes no sistema *Chandler* [6], estudado na secção 2.2.4. Eles apostam bastante nestes conceitos como fontes inovadoras para a criação de novos serviços. É essencial para a implementação de serviços deste tipo que o sistema suporte a noção de entidade (pessoa), grupos, contactos, e suporte também comunicação.

Outro conceito importante e onde se pode apostar é na criação de um sistema activo. Um **sistema activo** é um sistema que tenha iniciativa, que reaja a acontecimentos independentes das acções do utilizador. Um sistema com esta capacidade proporcionará serviços que agradará com certeza a qualquer utilizador. O sistema pode, inclusive ter a capacidade de aprendizagem e com isso adaptar-se a situações novas. Uma dificuldade na implementação de um sistema com estas características prende-se, por exemplo, com razões tecnológicas. Aplicações com interface *Web* normalmente reagem apenas a acções despoletadas pelo utilizador, ou seja, são aplicações passivas que baseiam-se a responder a pedidos do utilizador; raramente tomam a iniciativa de iniciarem elas a comunicação. De qualquer forma não quer dizer que não seja uma dificuldade que não possa ser ultrapassada. Se aliarmos a este conceito a existência de sensores que dêem ao sistema informação do ambiente onde o utilizador se encontra mais facilmente se imaginam serviços onde o sistema tenha a iniciativa de notificar o utilizador, ou mesmo sem notificar o utilizador, que reaja a essas alterações. Tendo o nosso sistema recursos de comunicação tão diversos como correio electrónico ou mensagens instantâneas facilmente o sistema pode ter a iniciativa de notificar o utilizador que algo aconteceu. Mais, tendo o sistema a noção de presença do utilizador pode haver critérios na forma de comunicar. Por exemplo, se o sistema tem conhecimento que o utilizador se encontra presente (ligado ao sistema) então o canal de comunicação preferencial poderá ser por mensagem instantânea. Por outro lado, se o utilizador se encontra desconectado e está com telemóvel pode-lhe ser enviado um SMS com uma notificação mais ou menos detalhada.

Para dar ideia da potencialidade dos serviços que um dia podemos vir a usufruir fica aqui o registo de alguns exemplos de serviços que actualmente residem apenas no nosso imaginário ou simplesmente em literatura de ficção científica.

Um exemplo gastronómico. Um dia os restaurantes vão ter a capacidade de publicar as suas especialidades. Essa informação vai estar disponível algures num servidor onde o nosso sistema poderá aceder. Função da capacidade de percepção do sistema pelo gosto gastronómico, por exemplo, do senhor Manuel, pode alertá-lo para a localização próxima de um restaurante específico com determinada especialidade. A característica "alertar" não é talvez uma novidade (encontra-se serviços semelhantes na operadora de telecomunicações TMN [83], por exemplo); no entanto, a capacidade de percepção do

gosto gastronómico do senhor Manuel já o é. A captação dessa informação pode ser obtida à custa do conhecimento dos hábitos alimentares do senhor Manuel.

Um outro exemplo na área do retalho. Se imaginarmos que um dia todos os produtos alimentares e/ou equipamentos eléctricos terão identificadores únicos (RFIDs), e que todos poderão interagir entre si incluindo com o nosso sistema, então é possível, por exemplo, o sistema despoletar automaticamente o processo de encomenda num supermercado dos produtos que vão acabando. Neste processo de negócio, a senhora Ana por exemplo, precisa apenas de ser notificada para confirmação da encomenda e recepção da mesma.

Um exemplo no ramo automóvel. Hoje em dia existem já carros que alertam o seu dono que o deve levar à oficina para revisão (função da detecção de anomalias ou quilómetros percorridos). Este serviço pode ainda ir mais longe permitindo que o sistema marque automaticamente uma data para a revisão tendo em conta a disponibilidade da oficina e a agenda pessoal da senhora Beatriz, por exemplo. Mais uma vez precisa de notificar a senhora Beatriz para lhe dar conhecimento e para permitir uma validação final.

Por último, um cenário simples mas que no futuro pode ser tão importante quanto o é hoje o telemóvel. O sistema vai buscar informação meteorológica a uma base de dados pública fornecida pelo Instituto Nacional de Meteorologia, por exemplo. Consegue inferir, através da informação facultada, que vai chover. O sistema detectando que a senhora Carla não leva consigo o seu guarda-chuva (através de RFIDs, por exemplo) alerta-a para o facto. Quem diz guarda-chuva diz qualquer peça de roupa ou calçado que não se dê bem com água. Estes "pormenores", que para nós actualmente considero insignificantes, poderão em dias futuros fazer parte da nossas vidas como o é, por exemplo, hoje em dia o telemóvel.

Os exemplos apresentados reflectem a necessidade duma capacidade computacional elevada (não só de execução mas também na capacidade de armazenamento e transmissão de informação), uma vez que praticamente todos sugerem a existência de componentes independentes para cada utilizador (pode-se chegar ao extremo de imaginar um sistema independente para cada utilizador). Para além da forte capacidade computacional exigida verifica-se que a criação de serviços inovadores e apelativos depende de imaginação.

A partir do momento que exista uma base sólida de serviços, a integração e criação de novos serviços fica facilitada. O importante é que o sistema seja construído segundo uma arquitectura expansível e suporte a integração de novos serviços sem danos para o

sistema conforme um dos princípios enunciados na secção 3.2, "gestão e subscrição flexível e dinâmica de serviços".

Não é portanto objectivo deste projecto a implementação de todos os serviços referidos. Acima de tudo, o que é importante é mostrar que é possível disponibilizar tais serviços, mediante mais ou menos trabalho. Aliás, no momento da escrita desta tese ainda está a decorrer a execução de um terceiro trabalho (referido na secção 1.2), enquadrado no PUC3, onde eventualmente poderá ser abordada a implementação de alguns serviços de mais alto nível. Assim sendo, para se conseguir abordar este tipo de serviços é necessário ter um conjunto de serviços base robusto o suficiente para proporcionar de uma forma simples a geração de novos serviços.

Capítulo 4

Protótipo Desenvolvido

4.1 Arquitectura de componentes

No estado actual do *"myComs"* existem apenas três componentes distintos ou serviços: (1)*webmail*; (2)IM&P; (3)Subscrição. Estes três componentes, ainda que distintos, podem interagir entre si e aparecem ao utilizador final integrados numa mesma apresentação. Ao nível da instalação, cada componente pode existir num nó físico independente, ou podem coabitar todos no mesmo nó físico. Apenas ao nível da interface *Web* as componentes que partilhem estado têm que executar no mesmo contexto, ou seja, partilhar o mesmo espaço físico e como tal não podem correr em nós distintos. Esse facto é representado na figura 4.1 pelo componente *root*. A instalação, tendo como preocupação a escalabilidade e tolerância a falhas, está detalhada no documento com os requisitos não funcionais [8].

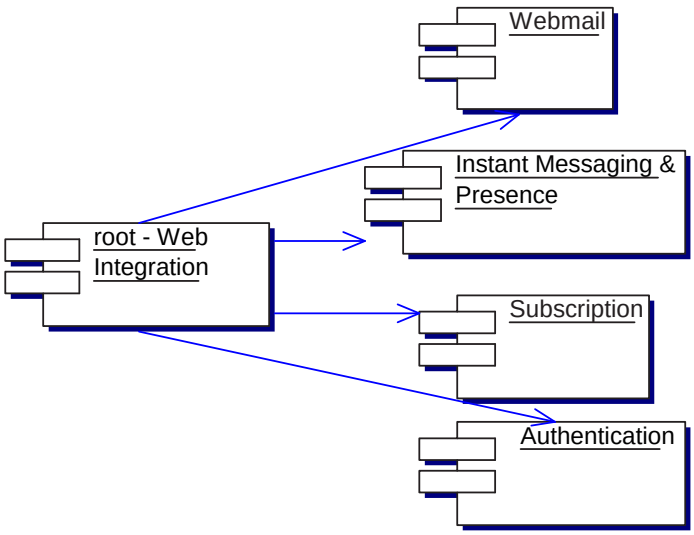

Figura 4.1: Modelo de componentes do myComs

Todos os módulos seguem a mesma estrutura interna representada na figura 4.2. Neste

caso concreto, está representado o serviço *webmail*, mas poderia estar igualmente representado o serviço *"subscrição"* ou *"IM&P"*.

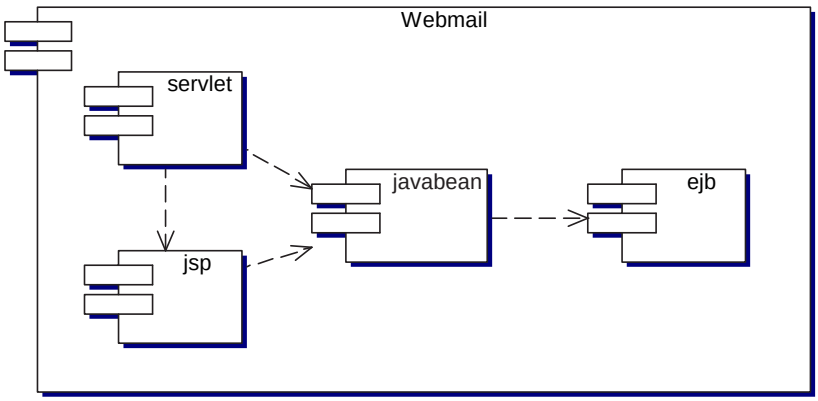

Figura 4.2: Modelo de componentes de qualquer módulo

Essa mesma estrutura aparece na figura 4.3 de acordo com os contentores onde se executam.

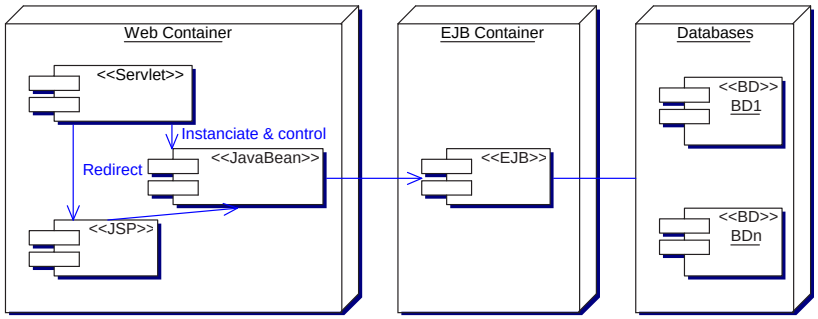

Figura 4.3: Estrutura de um módulo da aplicação myComs

A arquitectura mostrada baseia-se na especificação J2EE. J2EE (Java to Enterprise Edition) [31] é uma arquitectura standard, que utiliza a linguagem de programação Java, vocacionada para o desenvolvimento e distribuição de aplicações empresariais orientadas para a *Web*.

Esta arquitectura pode ser igualmente utilizada para desenvolver e distribuir aplicações intranet, substituindo os modelos de duas e três camadas. Significa que diferentes componentes podem correr num único nó.

Tem todas as vantagens inerentes a um standard, nomeadamente, com muito poucas alterações (sem alterações de fundo), uma mesma aplicação pode executar em diferentes implementações J2EE. Os contentores onde a aplicação irá executar já fornecem uma série de serviços de sistema que de outra forma seriam da responsabilidade do programador,

como por exemplo, serviços a nível de segurança, transacções, gestão do ciclo de vida dos componentes, entre outros.

4.2 Modelo de objectos na componente Web

Todos os módulos seguem a estrutura da figura 4.4. Trata-se do padrão de programação bastante conhecido denominado por MVC (*Model-View-Control*) [53, 18]. Neste padrão distinguem-se responsabilidades ao nível do modelo, do controlo e da apresentação. Neste caso o modelo é representado pelos *JavaBeans* [60], o controlo pelas *Servlets* [61] e a apresentação pelas JSPs [64]. As *Servlets* são o ponto de entrada da aplicação, aceitam os pedidos do utilizador, interagem com os *JavaBeans* e redireccionam para as JSPs; os *JavaBeans* representam a informação a ser mostrada e são eles que fazem de intermediários com os componentes empresariais; por último, as JSPs são o output da aplicação, são elas que geram as páginas HTML mostrando a informação recorrendo aos *JavaBeans*.

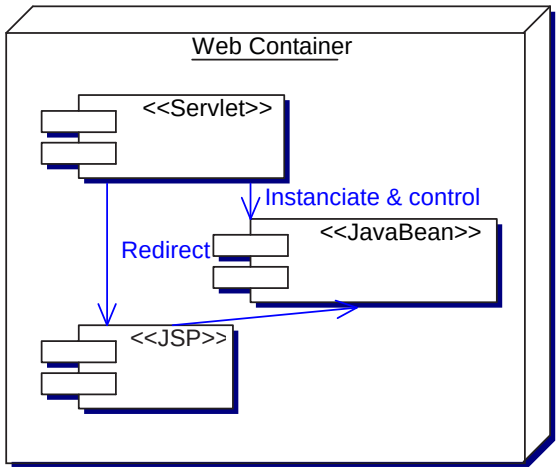

Figura 4.4: Estrutura da aplicação myComs na componente Web

De seguida são apresentados diagramas de classes das componentes *Web* dos vários serviços.

4.2.1 Subscrição e autenticação

A figura 4.5 revela exactamente o padrão adoptado: as *Servlets* a fazerem de controlo; os *JavaBeans* de modelo; e por último as JSPs a fazerem de apresentação. De realçar ainda nesta figura a classe `MemberTypeBean` que representa a informação a transitar entre os componentes empresariais e a componente *Web*. Ao nível das JSPs está referenciada apenas uma a título de exemplo não querendo com isso induzir que seja única.

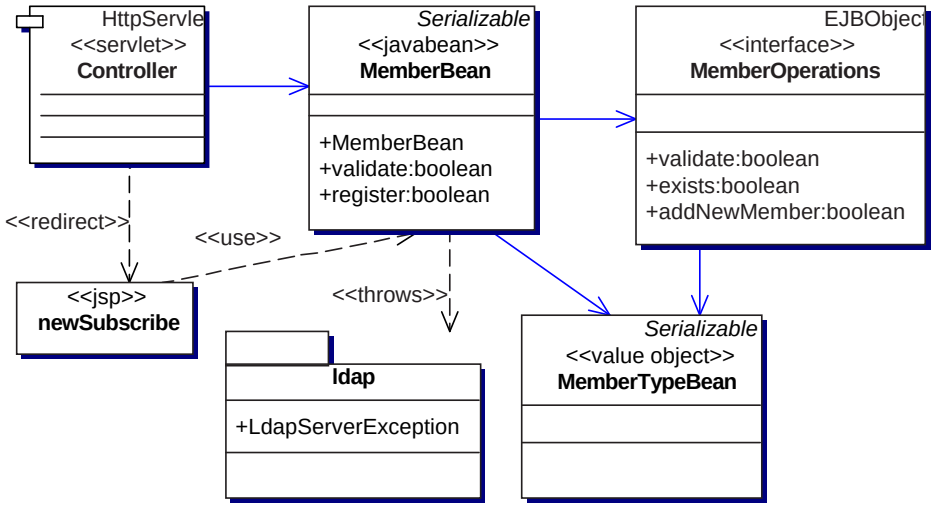

Figura 4.5: Diagrama de classes da componente Web do serviço subscrição

A autenticação respeita a mesma arquitectura. É utilizada uma autenticação do tipo *Simple*, correspondendo apenas à inserção de um *Login* e de uma *Password* que são passados em claro. Pode passar por trabalho futuro a utilização de outros tipos de autenticação, nomeadamente utilizando a API JAAS (*Java Authentication and Authorization Service*) [12], a utilização de um canal seguro recorrendo ao protocolo HTTP/SSL [34] (versão segura do protocolo HTTP), ou mesmo numa primeira fase recorrer a processos de autenticação fornecidos pela plataforma (esta autenticação não dispensa uma autenticação interna uma vez que está dependente da plataforma). A autenticação tem em comum com a subscrição o componente remoto que concentra as operações para ambos os contextos. Este facto é visível na figura 4.6. A descrição deste componente está detalhada na secção 4.3.1. Mais uma vez não estão presentes todas as JSPs que dizem respeito à autenticação. De salientar a classe `LoginVO` que representa a informação a transitar entre componentes. Neste módulo foi utilizada uma biblioteca `serviceLocator` que corresponde à implementação do padrão que localiza serviços, nomeadamente serviços remotos. Como trabalho futuro, todos os módulos deveriam utilizar esta biblioteca.

4.2.2 Correio electrónico

A diferença desta arquitectura para as anteriores corresponde na existência de interfaces que são utilizadas por JSPs. Ou seja, as JSPs não acedem directamente aos *JavaBeans* mas acedem através de uma interface que eles disponibilizam que pode, e na maioria dos casos acontece, não corresponder à interface total dos seus métodos. Neste sentido as JSPs estão limitadas em relação ao `JwmaController`, por exemplo, uma vez que tendo uma referência directa para a classe que representa a JSP passa a ter acesso a todos os

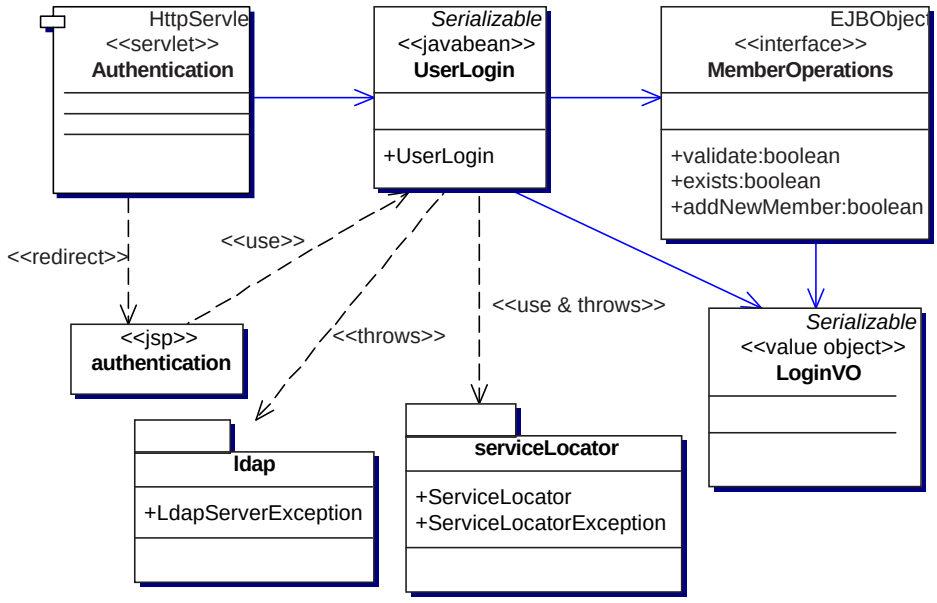

Figura 4.6: Diagrama de classes da componente Web do serviço autenticação

métodos públicos dessa classe. Esta solução sendo herdada do JWMA [89] não deixa de ter sentido uma vez que as JSPs, correspondendo à componente apresentação, apenas devem ter acesso aos métodos de `get` dos *JavaBeans*. Mais uma vez está apenas ilustrado na figura 4.7 um exemplo dos muitos que poderiam ser mostrados para este serviço.

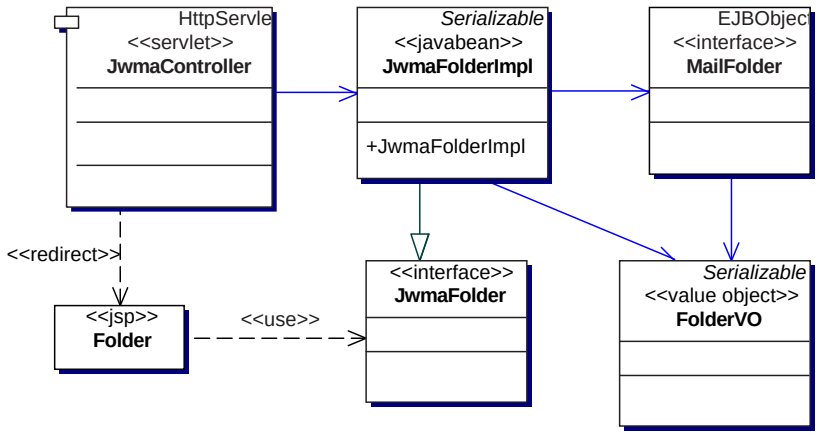

Figura 4.7: Diagrama de classes da componente Web do serviço correio electrónico

Embora não ilustrado na figura 4.7 de salientar que as classes `JwmaError` e `JwmaException` são excepções de aplicação geradas pelos componentes remotos e capturadas pelos componentes *Web*.

4.2.3 Mensagens instantâneas

A componente *Web* deste serviço foi aquela que implicou e envolveu maiores detalhes e comportamentos mais específicos. Tratando-se do serviço de mensagens instantâneas e notificações de presença estamos a falar de um serviço com características assíncronas e dessas notificações assíncronas terem que ser mostradas ao utilizador com um intervalo de tempo próximo de zero. É importante que não nos esqueçamos do conceito: "instantâneo". Estas características constituem assunto de discussão dado o suporte tecnológico utilizado na interacção com o utilizador. Estamos a falar de um acesso *Web* em que a aplicação do lado do cliente é um *browser*. Acontece que esta tecnologia assenta no conceito pedido/resposta permitindo que o *browser*, num cenário normal, apenas tome conhecimento de actualizações quando efectuar novo pedido ao servidor, pedido esse que por sua vez está condicionado por acções do utilizador. Para a implementação deste serviço era premente a existência de um suporte de comunicações ponto a ponto[1]. A solução, de uma forma geral, passou por acrescentar um outro canal de comunicação, para além do canal HTTP, por onde o servidor pudesse notificar o utilizador de novas actualizações. Não foi a única solução; outras foram investigadas e inclusive implementadas, mas a que ficou no final foi então a solução que envolveu mais um canal de comunicação. O conceito, vantagens, desvantagens das várias soluções está detalhado na secção 4.4.3.

Quanto à solução implementada foi necessário código do lado do utilizador para que pudesse ser criada uma ligação com o servidor. Esse código está na forma de uma *Applet* com o nome `Notifier`. Foi necessário do lado do servidor alguém que se responsabilizasse por aceitar ligações num porto pré estabelecido e que depois soubesse associar o novo canal criado ao objecto de sessão que o requereu. Essa associação é importante uma vez que a dada altura podem existir muitos objectos a requererem um novo canal (tem relação directa com o número de utilizadores a acederem no mesmo instante à aplicação). A classe responsável por aceitar ligações tem o nome `MyServerSocket`. Por último é necessário quem faça despoletar todo o processo, quem fique responsável pela ligação criada e quem realize as notificações de actualizações ao cliente. A classe com estas características é o `XmppBean` que é ao mesmo tempo o `SocketAcceptedListener` (uma vez que a primeira implementa a interface da segunda). A figura 4.8 ilustra a existência destas classes assim como as suas relações e inclusive as suas interfaces mais relevantes.

A figura 4.9 ilustra a sequência de acções na criação deste novo canal de comunicação assim como todos os intervenientes no processo. Conforme se pode verificar na mesma figura e na listagem 4.1 todo o processo é iniciado com a criação do `XmppBean`. Durante o processo da sua criação adiciona-se como interessado em receber a notificação de que o seu canal de comunicação foi criado. Daí esta classe implementar a interface

[1] usualmente conhecido por *peer-to-peer*

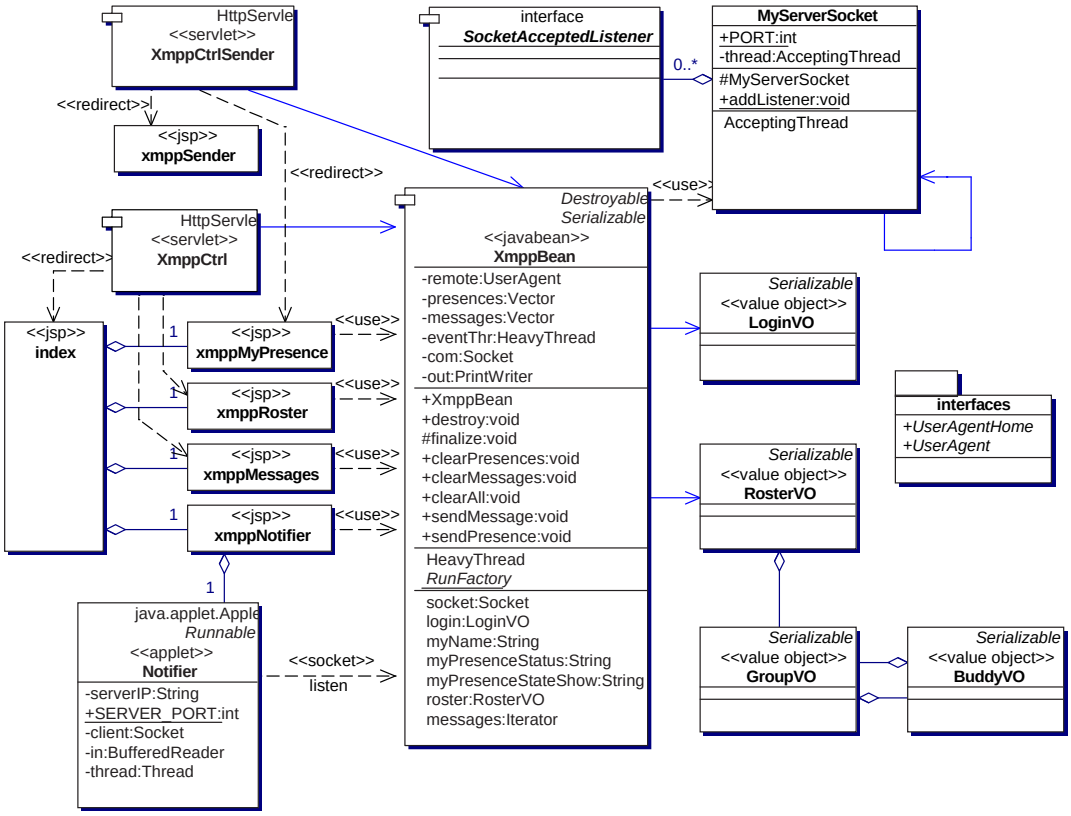

Figura 4.8: Diagrama de classes da componente Web do serviço IM&P

`SocketAcceptedListener`.

Listagem 4.1: Estabelecimento de um canal de comunicação

inocrea.web.userAgent.model.XmppBean.java

```
...
public class XmppBean
implements Destroyable, SocketAcceptedListener, Serializable {
  ...
  public XmppBean(InetAddress clientAddr) {
    initThreads(clientAddr);
    ...
  }
  ...
  private void initThreads(InetAddress clientAddr) {
    ...
    try { MyServerSocket.addListener(clientAddr, this); }
  }
  ...
  /**
  * Implementacao da interface SocketAcceptedListener.
  */
  public void setSocket(Socket socket) {
    ...
  }
  ...
}
```

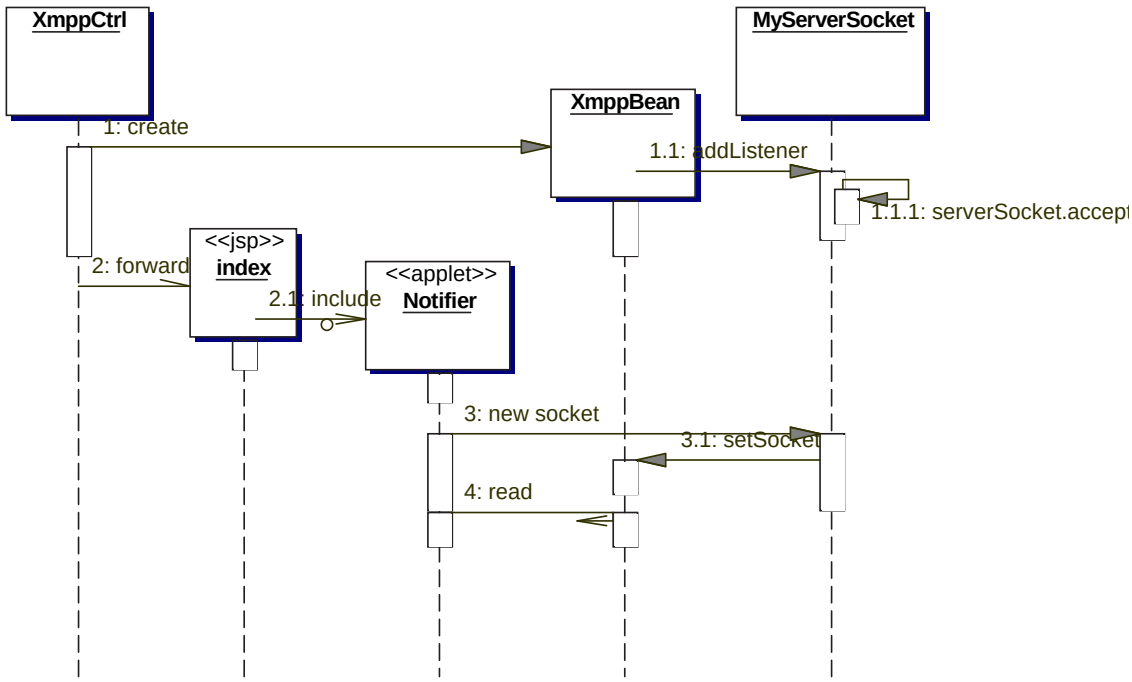

Figura 4.9: Diagrama de sequência de acções na criação do canal de comunicação entre cliente servidor

A classe `MyServerSocket` funciona como elemento centralizador de aceitação de ligações. Esta classe segue o padrão *Singleton* porque só pode haver uma única entidade pendurada num porto a aceitar ligações (mais do que uma gera excepção). O problema que aqui se colocava era então associar o interessado na notificação com a nova ligação uma vez que esta última é totalmente assíncrona e que num contexto de grande escala pode haver muitos interessados. A solução para este problema passou por utilizar o endereço do utilizador que vai fazer o pedido de criação da ligação, ou seja, no momento do `XmppBean` se adicionar ao `MyServerSocket` passar como parâmetro não apenas o interessado, que é ele próprio, mas também o endereço do utilizador. O `MyServerSocket`, por sua vez, mantém uma *Hashtable* com estas associações onde o endereço do utilizador serve de chave. Assim, no processo de aceitação de novo *socket*, como através deste consegue descobrir o endereço do utilizador, consegue notificar correctamente o interessado entregando-lhe a nova ligação. Todo este processo é visível na listagem 4.2.

Listagem 4.2: Classe centralizadora de pedidos de ligações

```
1  inocrea.web.userAgent.model.MyServerSocket.java
2  ...
3  public class MyServerSocket {
4    private Hashtable listeners = new Hashtable();
5    ...
```

```
6     /** Creates a new instance of MyServerSocket */
7     protected MyServerSocket(
8           InetAddress key, SocketAcceptedListener listener
9     ) throws IOException {
10      ...
11      listeners.put(key, listener);
12      ...
13    }
14    synchronized static public void addListener(InetAddress key,
15    SocketAcceptedListener listener) throws IOException {
16      ...
17      else self.listeners.put(key, listener);
18    }
19    ...
20    synchronized private void processListener(Socket socket) {
21      InetAddress key = socket.getInetAddress();
22      ((SocketAcceptedListener)listeners.get(key))
23        .setSocket(socket);
24      ...
25    }
26  }
```

É interessante ainda comentar a gestão do ciclo de vida do `MyServerSocket`. O ponto de interface para esta classe é apenas o método estático `addListener` (ver listagem 4.3). No seu processamento verifica se já existe instância, se não cria-a juntamente com as inicializações necessárias.

Listagem 4.3: Instanciação da classe centralizadora de pedidos de ligações

```
1   inocrea.web.userAgent.model.MyServerSocket.java
2     ...
3     synchronized static public void addListener{
4           InetAddress key, SocketAcceptedListener listener
5     ) throws IOException {
6       if (self == null) {
7         self = new MyServerSocket(key, listener);
8       }
9       else self.listeners.put(key, listener);
10    }
11    ...
```

Destas inicializações consta a criação do `ServerSocket` e a criação de uma *thread* em que essa sim tem como funções ficar à escuta de novas ligações e o processamento respectivo da nova ligação (ver listagem 4.4). Utilizou-se uma *thread* diferente porque o facto de ficar à espera de novas ligações implica uma espera bloqueante não permitindo reacção a outros acontecimentos, como por exemplo não conseguir atender o pedido de novos interessados de ligações.

Listagem 4.4: Inicialização da classe centralizadora de pedidos de ligações

```
1   inocrea.web.userAgent.model.MyServerSocket.java
2     ...
3     class AcceptingThread extends Thread {
```

```
4       ServerSocket server;
5       boolean toContinue = true;
6       AcceptingThread(ServerSocket server) {
7         this.server = server;
8       }
9       public void run() {
10        try {
11          while(toContinue) {
12            Socket socket = server.accept();
13            MyServerSocket.this.processListener(socket);
14          }
15        } catch(IOException e) {
16        }finally {
17          try { server.close(); } catch(IOException e) {}
18    }}}
19
20    /** Creates a new instance of MyServerSocket */
21    protected MyServerSocket(
22    InetAddress key, SocketAcceptedListener listener
23    ) throws IOException {
24      ServerSocket server = new ServerSocket(PORT);
25      thread = new AcceptingThread(server);
26      listeners.put(key, listener);
27      thread.start();
28    }
29    ...
```

Por cada vez que aceita nova ligação e entrega-a, retira o interessado da sua lista de interessados verificando se a mesma fica vazia: se assim ficar termina o processamento que inclui fechar o `ServerSocket` (foi uma forma de diminuir os recursos gastos uma vez que evita o bloqueio da *thread* à espera de novas ligações sem que haja interessados); se não retoma a espera de novas ligações. Mais uma vez a listagem 4.5 demonstra esta sequência de acções.

Listagem 4.5: Remoção da classe centralizadora de pedidos de ligações

```
1   inocrea.web.userAgent.model.MyServerSocket.java
2     ...
3     class AcceptingThread extends Thread {
4       ...
5       public void run() {
6         try {
7           while(toContinue) {
8             Socket socket = server.accept();
9             MyServerSocket.this.processListener(socket);
10          }
11        } catch(IOException e) {
12        }finally {
13          try { server.close(); } catch(IOException e) {}
14    }}}
15    ...
16    synchronized private void removeListener(InetAddress key) {
17      listeners.remove(key);
18      if (listeners.isEmpty()) {
19        thread.toContinue = false;
```

```
20          self = null;
21        }
22      }
23      synchronized private void processListener(Socket socket) {
24        InetAddress key = socket.getInetAddress();
25        ((SocketAcceptedListener)listeners.get(key)).setSocket(socket);
26        removeListener(key);
27      }
28      ...
```

De notar que os principais métodos do `MyServerSocket` são *synchronized*. Assim garante-se exclusão mútua na manipulação da tabela de interessados não permitindo a inconsistência da mesma.

Para terminar todo este processo de criação falta do lado do utilizador efectuar o pedido de novo canal de comunicação. Pela listagem 4.6 é possível verificar que esse feito é realizado no processo de inicialização da *Applet* `Notifier`. Após obter a ligação fica bloqueado à espera de novas actualizações por parte do servidor. Sendo a espera bloqueante, para não impedir receber outras notificações lançou-se mais uma vez uma nova *thread* dedicada a esta espera.

Listagem 4.6: Pedido do estabelecimento de uma ligação

```
1   inocrea.web.userAgent.control.Notifier.java
2   ...
3   public class Notifier extends java.applet.Applet implements Runnable {
4     ...
5     private Socket client = null;
6     private BufferedReader in = null;
7     private Thread thread = null;
8
9     public void init() {
10      try {
11        serverIP = getCodeBase().getHost();
12        Socket client = new Socket(InetAddress.getByName(serverIP), SERVER_PORT);
13        in = new BufferedReader(new InputStreamReader(client.getInputStream()));
14      } ...
15    }
16
17    public void start() {
18      if (thread == null) (thread = new Thread(this)).start();
19    }
20
21    public void run() {
22      while (thread != null) {
23        try {
24          int ch = in.read();
25          ...
26        } ...
27    } }
28    ...
29  }
```

É importante verificar no diagrama de classes que o padrão MVC (*Model-View-Control*)

se mantém com as *Servlets* a manterem o papel de controlo, os *JavaBeans* de modelo e as JSPs o papel de apresentação. De referir ainda que este serviço alberga duas componentes distintas: a recepção assíncrona de mensagens ou presenças; e o envio de alterações do estado de presença do utilizador assim como o envio de mensagens. Esta segunda componente, ao contrário da primeira, não ofereceu qualquer tipo de dificuldades de implementação uma vez que obedece à sequência natural de uma aplicação *Web* (pedido/resposta).

4.3 Modelo de objectos na componente EJB

Em relação aos EJBs não há uma arquitectura típica adoptada que valha a pena ser mencionada. As próximas subsecções apresentam os diagramas de classes dos respectivos serviços.

4.3.1 Subscrição e autenticação

Para este serviço foi desenvolvido uma biblioteca que permitisse a manipulação de dados num servidor LDAP[2] [71]. As classes que fazem uma manipulação genérica, nomeadamente a gestão de ligações ao servidor, pertencem ao pacote `eis.ldap` (eis[3]). As classes de carácter mais específico e que também têm a ver com a manipulação do servidor de LDAP pertencem ao pacote `user.util`. Para implementação deste serviço utilizou-se o componente empresarial de sessão sem estado `MemberOperations` (*session stateless*) que não apresenta qualquer particularidade que valha a pena ser referida. O diagrama de classes correspondente pode ser visto na figura 4.10.

Como trabalho futuro, em vez de uma biblioteca com classes genéricas a manipularem o servidor LDAP dever-se-ía utilizar um componente *Entity* que representasse uma entrada, ou seja um utilizador, no servidor LDAP [72]. Esse componente podia ser: (1) do tipo BMP (*Bean Management Persistency*) [22] para realizar directamente o acesso ao servidor LDAP; (2) do tipo CMP (*Container Management Persistency*) [22] recorrendo a um adaptador EJB QL (*Enterprise Java Beans Query Language*) [22] para LDAP.

A figura 4.11 descreve de uma forma genérica as acções a realizar no processo de subscrição e no processo de autenticação. Verifica-se que a autenticação é uma parte da subscrição uma vez que ambos precisam de verificar se o utilizador existe definido no servidor de utilizadores representado neste diagrama de sequência de acções por LDAP.

[2] *Lightweight Directory Access Protocol*
[3] *enterprise information system*

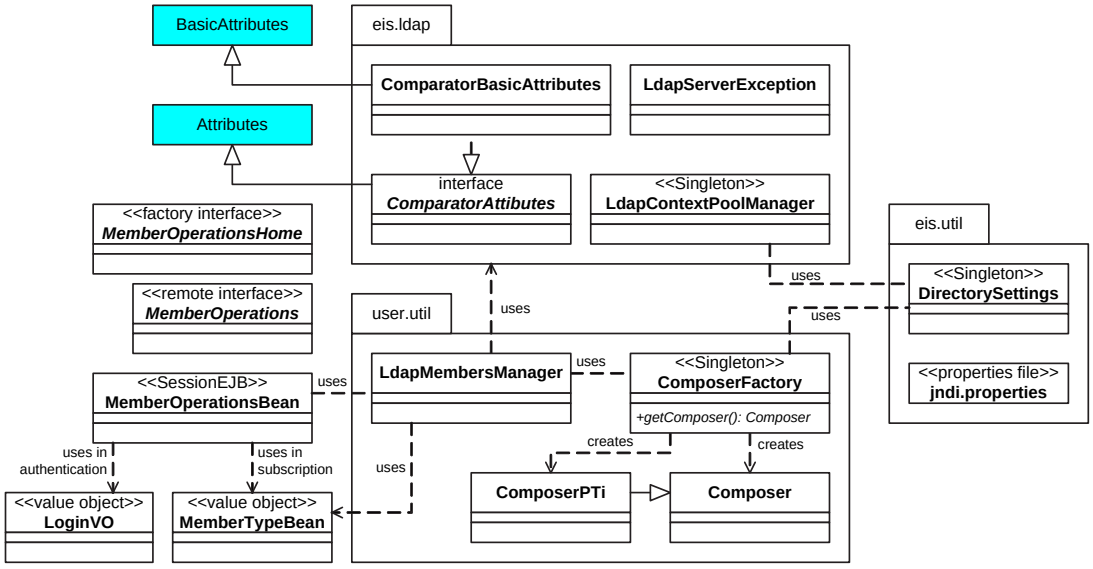

Figura 4.10: Diagrama de classes do módulo subscrição e autenticação

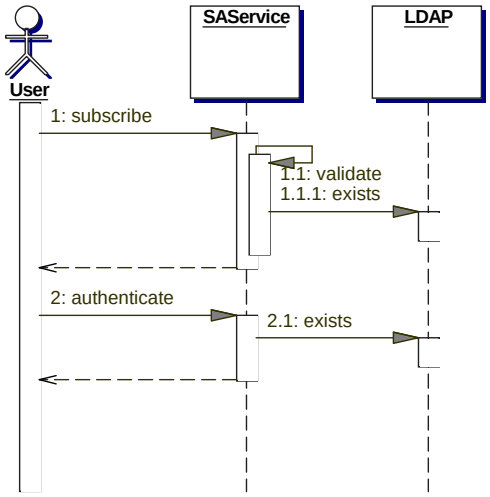

Figura 4.11: Sequência de acções na subscrição e validação de um utilizador

4.3.2 Correio electrónico

A arquitectura deste serviço foi de alguma forma condicionada dado ter sido herdado de uma aplicação fechada dada pelo nome JWMA [89]. Houve uma adaptação da versão original para produzir o modelo em componentes empresariais uma vez que era uma aplicação que se executava apenas no contexto *Web*. Todo esse processo de adaptação está detalhado na secção 4.4.2. No diagrama de classes da figura 4.12 as classes com fundo sombreado representam classes da plataforma. Neste caso, representam classes pertencentes à API `JavaMail` [59]. Depois de alguma maturação pareceu adequado existir

apenas uma única interface remota que representasse todo o serviço de correio electrónico em vez de cada componente ter uma interface remota. Seria mais razoável no seu todo. Não quer isto dizer que não devam existir os outros componentes, não; devem existir mas eventualmente apenas com interfaces locais e no final um componente que implementasse o padrão *session facade*. Mais uma vez, estas alterações deveriam ser encaradas como trabalho futuro.

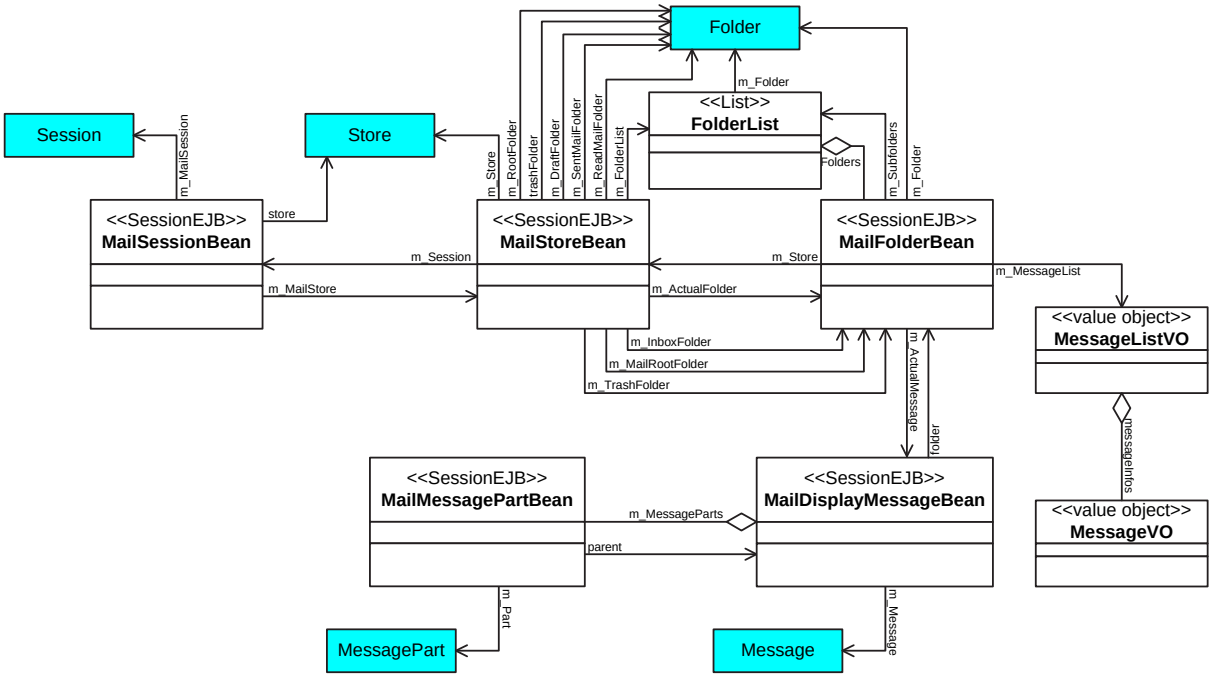

Figura 4.12: Diagrama de classes do módulo correio electrónico

O diagrama de sequência de acções da figura 4.13 ilustra as acções realizadas no âmbito do estabelecimento de uma sessão de correio electrónico, ou seja, as acções realizadas a partir do momento em que o utilizador se autentica no sistema. Verifica-se que numa fase inicial o utilizador se autentica igualmente perante o servidor de correio electrónico e só depois é que estabelece uma sessão de correio electrónico com a chamada a `initMailSession`. As restantes acções correspondem a obter informação em relação ao utilizador, nomeadamente, preferências associadas ao correio electrónico e a sua lista de contactos. A informação de preferências influenciará com certeza o *output* no utilizador. O diagrama ainda dá a informação que esses dados são guardados ao nível do sistema de ficheiros. Esta solução foi herdada da versão original e, embora considerando que não é uma boa solução, não foi alterada, até porque decorrem em paralelo outros trabalhos cujos objectivos respondem exactamente a estas preocupações (*"myAgenda"* e *"myContacts"*).

Por sua vez a acção `initMailSession` mostrada no diagrama anterior corresponde, para além de estabelecer uma nova sessão de correio electrónico, a uma série acções descritas no diagrama da figura 4.14. Ou seja, estabelecer uma nova sessão de correio

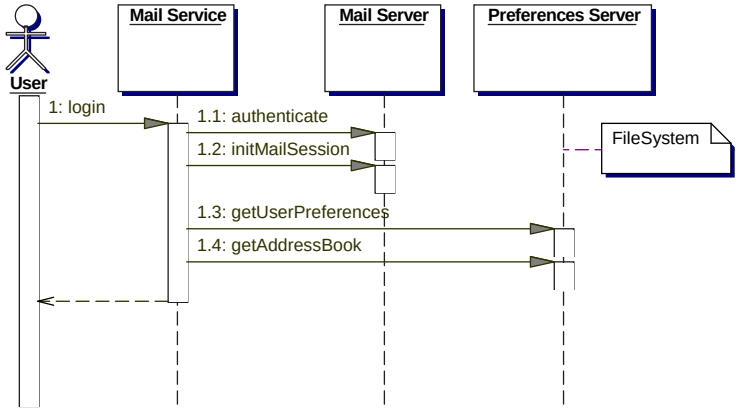

Figura 4.13: Sequência de acções no estabelecimento de uma sessão de correio electrónico

electrónico corresponde a obter uma referência para o repositório de dados, estabelecer a ligação, e obter as directorias respectivas.

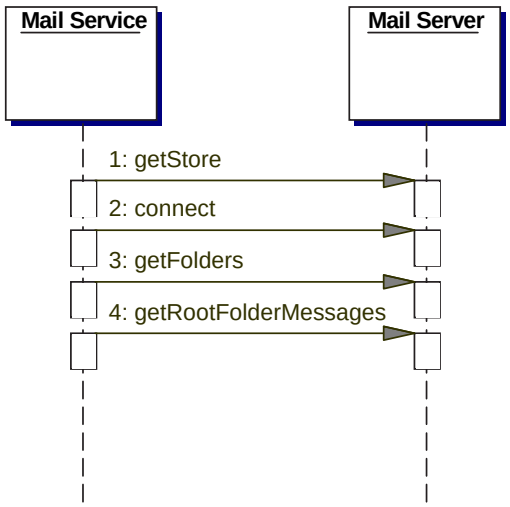

Figura 4.14: Sequência de acções no estabelecimento de uma ligação com o servidor de correio electrónico

4.3.3 Mensagens instantâneas

Para a implementação deste serviço foi utilizado uma biblioteca que abstrai os detalhes da comunicação entre cliente/servidor, que tem o nome *JabberBeans* [88]. Como abstrai apenas a criação de tramas para a comunicação foi desenvolvida uma outra biblioteca

de mais alto nível que providencia serviços igualmente de mais alto nível (figura 4.15). Podemos considerar serviços deste tipo como a ligação e autenticação, o envio de mensagens, presenças, igualmente a recepção de mensagens e presença. Todas as classes desta biblioteca pertencem ao pacote XMPP[4] [46]. Eventualmente, também aqui como trabalho futuro poder-se-ía pensar em utilizar um componente empresarial do tipo MDB[5] [22] ou mesmo utilizar o JCA[6] [62].

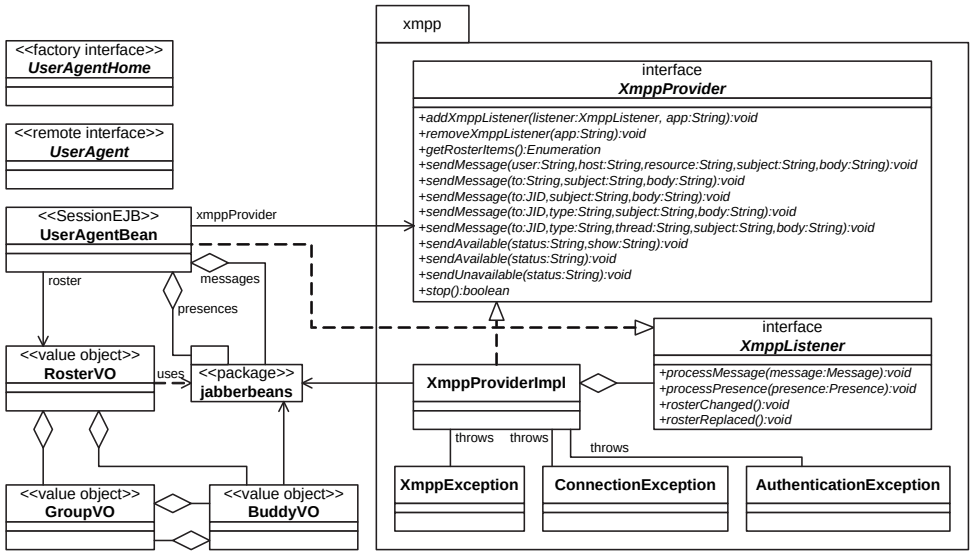

Figura 4.15: Diagrama de classes do módulo mensagens instantâneas e notificações de presença

O diagrama da figura 4.16 ilustra as acções a realizar no estabelecimento de uma nova ligação com o servidor *Jabber* (*jabberd*). Resumindo, quando um utilizador se autentica no sistema PUC, corresponde, no que diz respeito ao serviço de IM, a criar uma nova ligação como servidor, autenticar o utilizador no servidor, pedir a lista de amigos associada ao utilizador, enviar o estado de presença inicial, e por último, adicionar-se como interessado na notificação da chegada de mensagens e alterações do estado de presença dos seus amigos.

4.3.4 Interfaces Remotas

Esta secção apresenta as interfaces remotas dos diferentes componentes envolvidos na aplicação *"myComs"*. Entenda-se por interfaces, interfaces de criação e interfaces de componente, todas elas remotas; interfaces locais dos componentes, a existirem, não são apresentadas nesta secção. Devido à extensibilidade apresentada por estas interfaces

[4] *eXtensible Message Presence Protocol*
[5] *Message Driven Bean*
[6] *J2EE Connector Architecture*

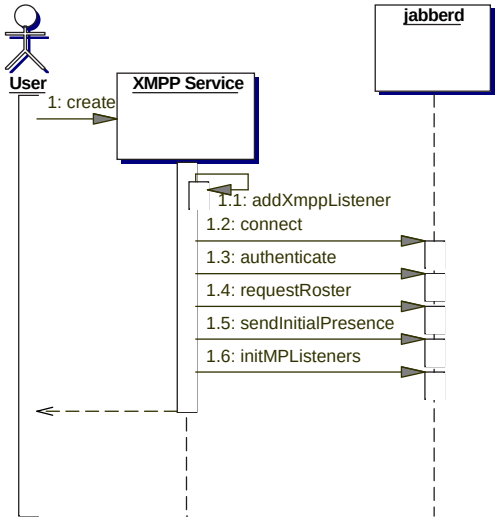

Figura 4.16: Sequência de acções no estabelecimento de uma ligação com o servidor jabberd

são apenas enumerados os pacotes com o nome das interfaces que existem, conforme a figura 4.17.

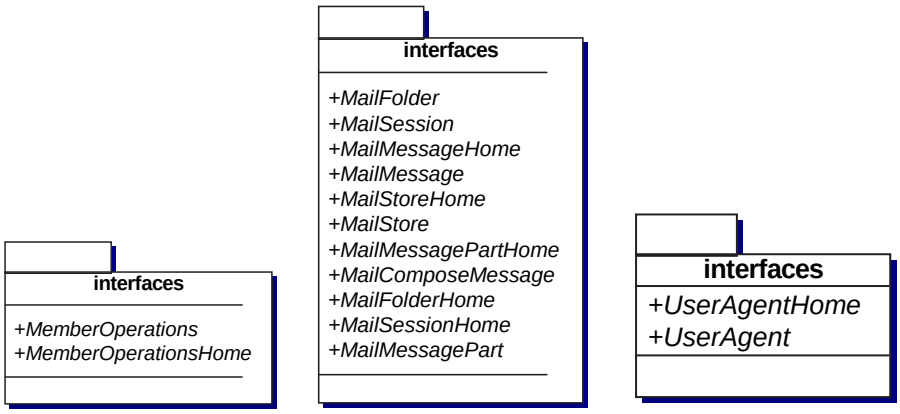

Figura 4.17: Interfaces remotas de criação e de componentes dos serviços

4.4 Detalhes e decisões de implementação

4.4.1 Subscrição e autenticação

À parte da descrição da arquitectura do serviço que se encontra detalhada na secção 4.2.1 e secção 4.3.1 é de salientar que dento do pacote `eis.util` existe um ficheiro de pro-

priedades com o nome `jndi.properties` e uma classe que representa este ficheiro chamada `DirectorySettings`. Estes ficheiros são utilizados para obter informação de configuração, essencialmente, do servidor LDAP. Numa fase inicial o ficheiro de propriedades chamava-se `directory.properties` e na classe que o representa estava o caminho absoluto para este mesmo ficheiro. Ainda que persistindo durante algum tempo sempre se considerou uma má solução uma vez que qualquer alteração do local de instalação corresponderia a correcção na classe, tendo como consequência a necessidade de criar novamente toda a aplicação de distribuição. A solução adequada seria dar um caminho relativo. Acontece que esse caminho relativo é relativo à directoria de arranque da máquina virtual Java que corresponde à directoria `/bin` do *JBoss*. Deixa de ser uma boa solução porque mais uma vez se se mudasse a directoria de instalação do *JBoss* ter-se-ía que alterar código que conduzia novamente a todas as desvantagens a isso associadas. Já no final, durante o processo de integração dos serviços, foi novamente abordado este problema onde se identificaram várias soluções:

- especificar todas as propriedades como entradas de ambiente no ficheiro `ejb-jar.xml` através dos campos `env-entry`.
- especificar um ficheiro de propriedades `jndi.properties`.
- obter um `resource-ref` associado ao ficheiro em causa.

As duas primeiras soluções são boas se a informação em causa estiver na forma chave/-valor (um ficheiro de propriedades tem esse formato). Se for uma informação que diga respeito apenas a um único componente então deverá ser utilizada a primeira solução, se for informação partilhada por vários componentes então deverá ser utilizada a segunda solução. Se a informação não estiver na forma chave/valor, por exemplo se se pretender ler um ficheiro na íntegra, então deverá ser utilizada a terceira solução. Este tema é abordado novamente na secção 4.4.2 onde está descrita com algum detalhe a terceira solução. Neste contexto foi utilizada a segunda solução, o ficheiro `jndi.properties`, uma vez que as propriedades definidas nesse ficheiro poderão ser usadas por diferentes componentes. O excerto de código apresentado na listagem 4.7 revela a facilidade de obter os atributos definidos num ficheiro `jndi.properties`.

Listagem 4.7: Carregamento em memória do jndi.properties

inocrea.eis.util.DirectorySettings.java

```
...
/** Creates a new instance of DirectorySettings */
DirectorySettings() throws Exception {
  if (instance != null) throw new Exception();
  try {
    javax.naming.Context ic = new javax.naming.InitialContext();
    p = ic.getEnvironment();
    instance = this;
```

```
10        }
11        ...
12    }
13    ...
```

Esta facilidade apenas se consegue por o ficheiro se chamar `jndi.properties`. Outro nome qualquer, e já estaríamos perante a terceira solução.

4.4.2 Correio electrónico

De forma a integrar no *"myComs"* o serviço de correio electrónico, foi adoptado e adaptado como projecto de base o sistema JWMA [89], que tem a característica de ser código livre.

No entanto, conforme já referido, o JWMA original baseia-se numa arquitectura *Web* simples, a duas camadas (servidor *Web* + servidor de correio electrónico), i.e., o JWMA é executado apenas no contexto de um contentor *Web*, acedendo directamente ao servidor de correio electrónico. Esta arquitectura pode ser observada na figura 4.18.

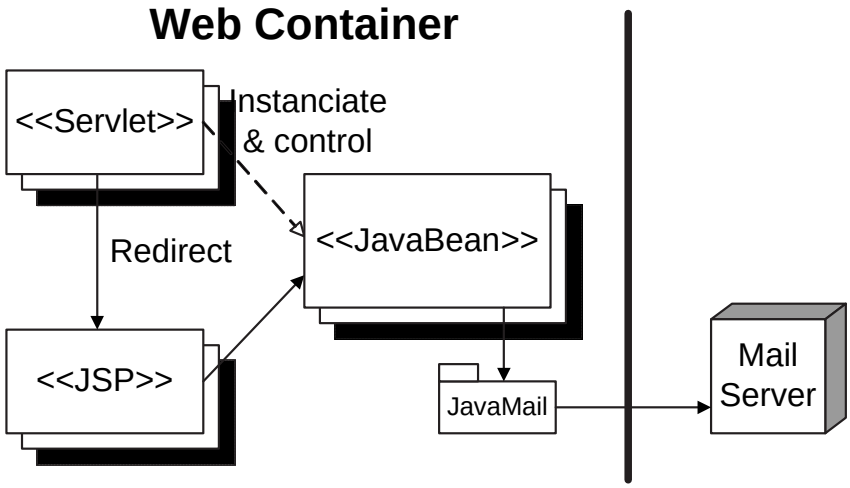

Figura 4.18: Arquitectura original do serviço de correio electrónico

Este facto levantou as subsequentes e inerentes dificuldades: (1) o software não pode ser reutilizado facilmente por outros componentes; (2) dificuldades em separar de facto a camada de apresentação da camada da lógica de negócio; (3) a nível de segurança; (4) dificuldades no suporte a transacções; e (5) no suporte à escalabilidade do serviço. De forma a ultrapassar estas dificuldades foi proposta e implementada uma arquitectura adaptada (da versão original do JWMA) conforme sugerido na figura 4.19.

A adaptação correspondeu a introduzir uma segunda camada entre o contentor Web e o servidor de correio electrónico, designado por contentor de EJBs. O acesso ao servidor de correio electrónico passa a ser da responsabilidade dos componentes aí residentes, designados por EJBs (*Enterprise Java Beans*) [22]. Para além dessa responsabilidade, os

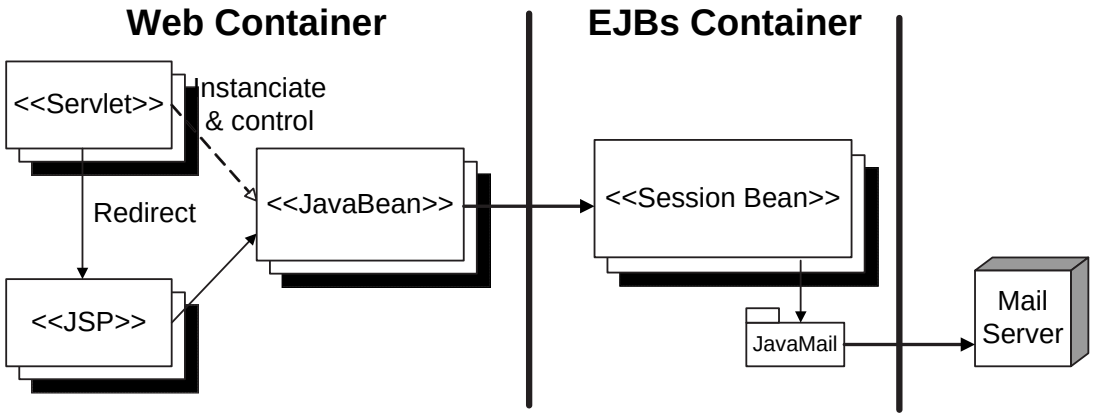

Figura 4.19: Arquitectura final do serviço de correio electrónico

EJBs representam o modelo de negócio. A utilização de EJBs responde às dificuldades encontradas na versão original, nomeadamente: representando apenas o modelo, separam-no de facto da apresentação que pode ir ao nível físico, e como tal, facilmente para um mesmo modelo podem existir diferentes apresentações; os EJBs disponibilizam interfaces remotas permitindo uma fácil utilização por outros componentes; é da responsabilidade do contentor de EJBs dar suporte a transacções e suporte a nível de segurança libertando o programador dessa responsabilidade; para permitir, entre outras características, escalabilidade, o contentor de EJBs dá suporte à criação de um ambiente distribuído com a criação de *clusters* [8].

As figuras 4.20 e 4.21 são um exemplo da adaptação que se fez ao nível dos *JavaBeans*. Na figura correspondente à arquitectura final está ilustrado apenas uma parte da mesma uma vez que para os restantes elementos o processo é idêntico.

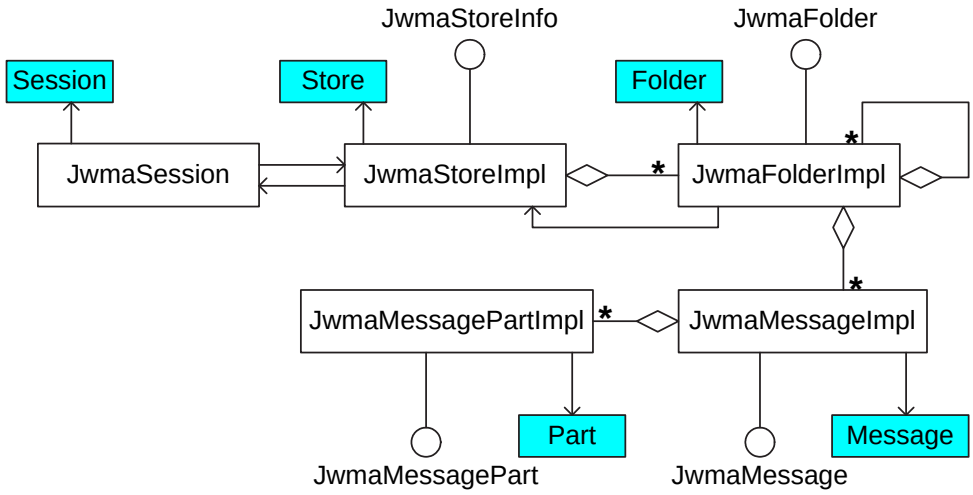

Figura 4.20: Relações entre JavaBeans na versão original

Em relação aos *JavaBeans* foram mantidas as interfaces, o que permitiu não ter que

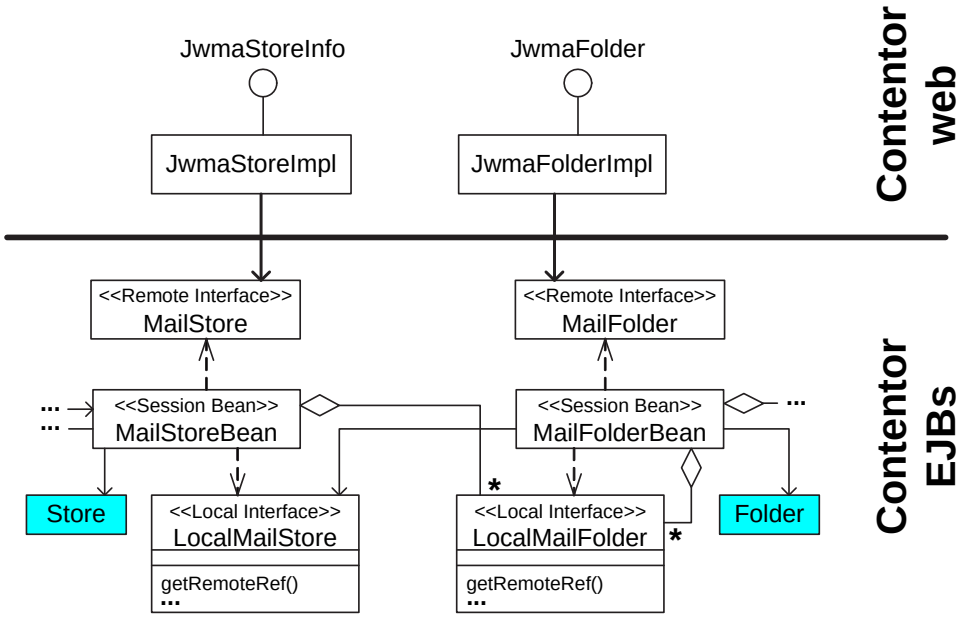

Figura 4.21: Adaptação de JavaBeans a EJBs

se alterar as classes que os usavam (*Servlets* e JSPs), mas ganharam referências remotas para os respectivos EJBs. Cada EJB tem a si associado duas interfaces: interface remota, que permite o acesso por clientes remotos (e.g. *JavaBeans*); interface local, que é utilizada para manter localmente as relações entre EJBs. Foram utilizadas interfaces locais por duas razões: primeiro, e admitindo que os vários EJBs não têm sentido correr em contextos diferentes, ganha-se em eficiência usando interfaces deste tipo; segundo, porque na interacção entre EJBs são passados como parâmetros instâncias pertencentes à API do `JavaMail` [59]. Como estas classes não são serializáveis não podem ser passadas por valor, apenas por referência e como tal apenas num contexto local.

De fazer notar ainda que a interface local disponibiliza um método que retorna uma interface remota em que ambas, a remota e a local, referenciam a mesma instância do EJB. Tal é conseguido através do atributo `sessionContext` que é afectado ao componente durante a sua criação na chamada ao método `setSessionContext`. Este factor não seria importante se tratasse-se de componentes empresariais sem estado (*stateless session* EJB); acontece que estes componentes são componentes *statefull* que mantêm o estado ao longo da sessão, e como tal ambas as referências devem referenciar a mesma instância. Por outro lado, é sempre possível obter uma referência remota de um EJB através de uma referência local desse mesmo EJB, nunca o contrário. Também não tem sentido retornar uma referência de uma interface local para um cliente que está a correr noutro contexto distinto.

Com esta estratégia algumas interfaces locais e remotas partilham a mesma interface, conforme se pode verificar nas listagens 4.8 e 4.9.

Listagem 4.8: Interface local de um componente

```
inocrea.ejb.mail.interfaces.LocalMailFolder.java
...
public interface LocalMailFolder extends EJBLocalObject {
  ...
  public String getName();
  public String getPath();
  public int getType();
  public boolean isType(int type);
  ...
}
```

Listagem 4.9: Interface remota do mesmo componente

```
inocrea.ejb.mail.interfaces.MailFolder.java
...
public interface PucFolder extends EJBObject {
  ...
  public String getName() throws RemoteException;
  public String getPath() throws RemoteException;
  public int getType() throws RemoteException;
  public boolean isType(int type) throws RemoteException;
  ...
}
```

Embora a nível de assinatura das operações difiram nas respectivas excepções lançadas, a sua implementação é única (listagem 4.10).

Listagem 4.10: Implementação do componente

```
inocrea.ejb.mail.session.MailFolderBean.java
...
public class MailFolderBean implements javax.ejb.SessionBean {
  ...
  public String getName() {
    ...
  }//getName
  public String getPath() {
    ...
  }//getPath
  public int getType() {
    ...
  }//getType
  public boolean isType(int type) {
    ...
  }//isType
  ...
}
```

Na versão original existe a classe `JwmaKernel` cujo objectivo é ser uma classe de configuração, ou seja, prepara directorias, mapeia nomes a classes, prepara serviços de correio electrónico, prepara persistência de dados do utilizador, prepara *logs*, entre outros. Para além destas inicializações compete-lhe também fazer a gestão da persistência da informação do utilizador. Como algumas destas inicializações são igualmente necessárias

ao nível dos EJBs, foi necessário criar uma classe homóloga, para correr no contexto dos EJBs, com o nome de `MailKernel`. Conforme se pode verificar na listagem 4.11 o `JwmaKernel`, no seu processo de criação, recebe com parâmetro de entrada o caminho para uma série de ficheiros que são utilizados no estabelecimento de uma sessão.

Listagem 4.11: Passagem do caminho à classe JwmaKernel

inocrea.web.webmail.util.JwmaKernel.java

```
...
public void setup(String path) throws Exception {
  ...
}
...
```

Esse caminho não é absoluto, é relativo e é conseguido à custa da localização da directoria `WEB-INF` (directoria standard que existe sempre no contexto de um componente *Web*). Na listagem 4.12 pode-se verificar a passagem da localização ao `JwmaKernel`.

Listagem 4.12: Obtenção e passagem do caminho à classe JwmaKernel

inocrea.web.webmail.control.JwmaController.java

```
...
public void init(ServletConfig config) throws ServletException {
  ...
  //kernel bootup
  JwmaKernel myKernel = JwmaKernel.getReference();
  //check runtime environment
  String ver=config.getInitParameter("container");
  String loc=config.getInitParameter(“location”);
  ...
  //check container
  if(ver.equals("2.2") || ver.equals("2.3")) {
    loc=config.getServletContext().getRealPath(loc);
  }
  ...
  //setup kernel with location
  myKernel.setup(loc);
  ...
}
...
```

A localização referência é dada como parâmetro de inicialização à *Servlet* `JwmaController` através do ficheiro de configuração `web.xml`, exemplificado na listagem 4.13.

Listagem 4.13: Descriptor XML da componente Web do correio electrónico

descriptors/webmail/web.xml

```
<web-app>
  ...
  <servlet>
    ...
    <servlet-name>jwma</servlet-name>
    <servlet-class>
      inocrea.web.webmail.control.JwmaController
```

```
9       </servlet-class>
10      ...
11      <init-param>
12        <param-name>location</param-name>
13        <param-value>/WEB-INF</param-value>
14      </init-param>
15      ...
16    </servlet>
17    ...
18  </web-app>
```

Todo este processo de inicialização do `JwmaKernel` foi necessário replicar de alguma forma no processo de inicialização do `MailKernel`, com a diferença de deixar de ser no contexto *Web* para ser num contexto de EJBs. Neste contexto pelo que foi dado a perceber não é possível dar uma localização referência através do ficheiro standard `ejb-jar.xml` (o mais semelhante foi o `resource-ref` que ainda assim pareceu não ser possível). Uma forma de se conseguir passar valores de inicialização a um EJB podería ser através do campo `env-entry` ou através do ficheiro de propriedades `jndi.properties`. Ambas as soluções estão já descritas na secção 4.4.1. De qualquer forma em ambas as soluções, esta passagem de valores seria sempre na forma chave/valor, o que deixa de ser viável quando se pretende por exemplo um ficheiro em XML. A solução adoptada para este problema resume-se no método `computePath` do componente empresarial `MailSession` conforme listagem 4.14 e 4.15.

Listagem 4.14: Computação do caminho para o ficheiro jwma.properties

```
1  inocrea.ejb.mail.session.MailSessionBean.java
2  ...
3  private void computePath() throws Exception {
4    String urlPath = getClass().getClassLoader().getResource("etc/jwma.properties")
5          .toString();
6    String urlSubPath = urlPath.substring(0,urlPath.indexOf("etc/jwma.properties"));
7    path = urlSubPath.substring((new String("jar:file:")).length(), urlSubPath.length());
8    java.io.File file = new File(path + "etc");
9    ...
10 }
11 ...
```

Conforme se pode constatar no código acima o algoritmo é completamente diferente da componente *Web* (listagem 4.7). Na componente EJB é necessário calcular a localização da directoria em causa. Se a directoria não existir é necessário criá-la e descompactar do ficheiro `jar` todos os ficheiros necessários à inicialização da sessão de correio electrónico (no caso concreto corresponde a todos os ficheiros presentes na directoria `etc`).

Listagem 4.15: Criação das directorias para estabelecimento de uma sessão de correio electrónico

```
1  inocrea.ejb.mail.session.MailSessionBean.java (continuação)
2  ...
```

```
3  private void computePath() throws Exception {
4    ...
5    java.io.File file = new File(path + "etc");
6    if (!file.exists()) {
7      file.mkdirs();
8      (new File(path + "logs")).mkdir();
9      (new File(path + "data")).mkdir();
10     java.net.URL url = new java.net.URL(urlSubPath);
11     java.net.JarURLConnection jarConnection =
12         (java.net.JarURLConnection)url.openConnection();
13     java.util.jar.JarFile jarFile = jarConnection.getJarFile();
14     Enumeration enum = jarFile.entries();
15     while (enum.hasMoreElements()) {
16       java.util.jar.JarEntry jarEntry = (java.util.jar.JarEntry)enum.nextElement();
17       if (jarEntry.isDirectory()) continue;
18       if (jarEntry.getName().startsWith("etc/")) {
19         java.io.InputStream in = jarFile.getInputStream(jarEntry);
20         java.io.FileOutputStream out =
21           new java.io.FileOutputStream(path+'/'+jarEntry.getName());
22         byte [] buffer = new byte [1024];
23         int i;
24         while ((i = in.read(buffer)) != -1) out.write(buffer, 0, i);
25         in.close();
26         out.close();
27       }
28     }
29   }
30 }
31 ...
```

4.4.3 Mensagens instantâneas

Para implementação deste serviço foi utilizada uma biblioteca Java código livre, *Jabber-Beans* [88], catalogada também no portal *sourceforge* [77], que implementa o protocolo *Jabber*. Essa implementação é de baixo nível já que apenas abstrai a criação de tramas em XML e a comunicação entre cliente e servidor. Devido a este facto foi desenvolvida uma biblioteca Java, XMPP, que disponibiliza um conjunto de interfaces de mais alto nível. Nomeadamente, disponibiliza interfaces para autenticação do utilizador, para envio de mensagens e notificações de presença, bem como interfaces para notificação de mensagens e de presença por parte do servidor.

Conforme sugerido na figura 4.22, constata-se que a arquitectura deste serviço é semelhante à arquitectura adoptada no serviço de correio electrónico.

Durante a implementação deste serviço surgiu a dificuldade de criar a ilusão da recepção instantânea de mensagens e notificações de presença. Esta dificuldade surgiu na medida em que por um lado o servidor *Jabber* pode iniciar sessões de comunicação; mas, por outro lado, os clientes são *browsers Web* e como tal utilizam o protocolo HTTP que se baseia no padrão de comunicação pedido/resposta. Foram analisadas três soluções para

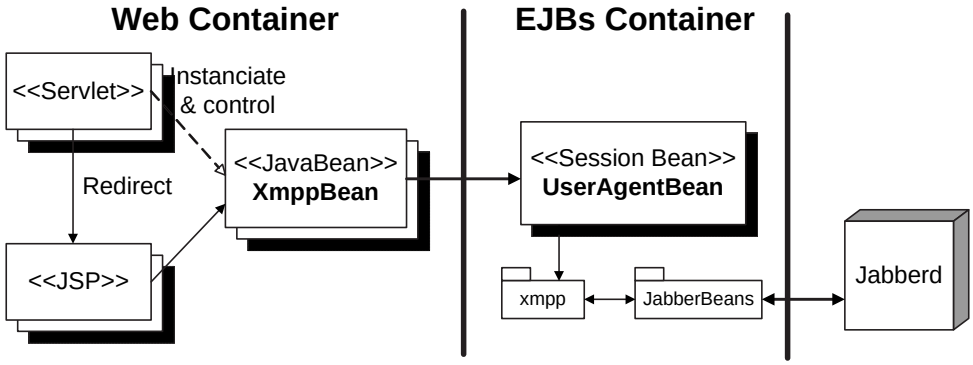

Figura 4.22: Arquitectura do serviço de mensagens instantâneas e de presença

ultrapassar tal dificuldade. Para qualquer uma delas são apresentadas também as suas desvantagens correspondentes.

Polling

Segundo esta estratégia o *browser* faz sistematicamente um refrescamento automático da página onde ocorrem as actualizações. Desta forma quaisquer alterações que ocorram no servidor manifestam-se na página refrescada. Esta solução tem a desvantagem de exigir um compromisso entre a sobrecarga na rede versus o tempo de actualização. Se se quer dar a noção de instantâneo diminui-se o tempo de actualização, contribuindo para o aumento de carga na rede; é uma desvantagem visto a maior parte dos refrescamentos não produzirem de facto quaisquer alterações.

Resposta incompleta

Nesta solução o servidor mantém o canal de resposta aberto. Desta forma assim que tiver actualizações a fazer, realiza-as através desse canal (pode enviar logo as actualizações ou provocar no cliente o refrescamento da página a actualizar). No entanto, para que este canal permaneça válido é necessário manter a *thread* que o criou, o que constitui uma desvantagem pelo facto de exigir um elevado número de recursos do lado do servidor *Web*; com a agravante de serem recursos que não são controlados pelo programador, mas sim pelo servidor, estando portanto dependente da implementação do mesmo.

Applet

Nesta solução, na página HTML produzida vai referenciada um *Applet* que, não tendo apresentação visual, abre um canal de comunicação com o servidor, ficando bloqueada à espera de notificações. Tem desvantagens ao nível da segurança e da administração de rede uma vez que o canal utilizado não é o porto 80. Por sua vez também gasta recursos

do lado do servidor *Web* só que, ao contrário da solução anterior, gasta recursos que o programador controla.

As três técnicas foram implementadas embora apenas a última, *Applet*, e a primeira, *polling*, tenham sido utilizadas na aplicação. A primeira solução é utilizada automaticamente se o *browser* não puder executar o *Applet* Java.

Qualquer umas das três técnicas apresentadas são conhecidas conceptualmente como *client pull* [67] e *server push* [67]. A primeira técnica, *polling*, enquadra-se no *client pull*, ou seja, o servidor provoca que seja o cliente a originar a comunicação (através de *script* ou através do *header* do documento). As duas últimas técnicas, resposta incompleta e *Applet*, enquadram-se no *server push*, onde é o servidor que, mantendo a ligação aberta com o cliente, toma a iniciativa de enviar para este último a informação actual.

4.4.4 Recuperação

Por restrições de tolerância a falhas, escalabilidade e desempenho foi necessário criar um nó lógico constituído pelo agrupamento de várias máquinas (*cluster*) onde o estado da aplicação se encontrasse partilhado. Esta replicação, acompanhada de recuperação de estado, permite que o utilizador não se aperceba, por exemplo, que a máquina que lhe estava a providenciar o serviço foi abaixo, uma vez que há outra com o mesmo estado que automaticamente lhe passa a providenciar o mesmo serviço.

Este assunto, nomeadamente arquitectura do *cluster*, requisitos não funcionais, entre outros, foi tema de um trabalho final de curso [8] que decorreu em paralelo com este trabalho e que, por não ser o tema fulcral deste trabalho, não se irá entrar em grandes detalhes. No entanto, é importante que se refira a intervenção ao nível da aplicação para permitir a recuperação de estado.

Primeiro, é importante perceber que para recuperar estado é necessário que este esteja replicado em máquinas distintas. É da competência do servidor de aplicações gerir esta replicação. A manutenção do estado verifica-se sempre que este se altera: (1) a nível de componentes *Web* sempre que o estado de sessão é alterado, ou seja, no `HTTPSession` sempre que se faça `setAttribute` ou `removeAttribute`. Não promove manutenção de estado a alteração do estado de um objecto afecto à sessão; (2) a nível de componentes empresariais a manutenção é feita sempre que o estado deste se altera.

No caso do *"myComs"* para efeitos de estudo e investigação, a recuperação dos componentes empresariais foi feita apenas no serviço de correio electrónico.

Tratando-se de componentes do tipo *statefull*, a manutenção de estado é realizada serializando o componente. Para tal, o servidor de aplicações fá-lo entrar no modo desa-

ctivação seguido imediatamente de activação. São dois modos que pertencem ao ciclo de vida destes componentes e que ao nível do servidor de aplicações promove a serialização e desserialização, respectivamente, dos mesmos; para além destes efeitos internos, cada uma desta transição de estado promove o respectivo *callback* no componente para que este tenha oportunidade de realizar acções de acordo com o acontecimento, i.e., adquirir ou libertar recursos, respectivamente.

Esta replicação seria quase transparente à aplicação não fosse os atributos voláteis que os componentes têm. Nomeadamente, os objectos pertencentes à API do `JavaMail`, que sendo instâncias de classes não serializáveis, têm que ser declaradas no componente como "transientes" (modificador de um atributo) para que no processo de serialização sejam ignorados. Esta característica obrigou a existir na aplicação código específico para recuperação de estado, uma vez que estes atributos não são recuperados, nomeadamente:

- todos os componentes filhos passaram a ter obrigatoriamente uma referência local para o componente pai;
- todos os componentes tiveram que disponibilizar um método na interface local que permitissem obter o objecto do pacote do `JavaMail` correspondente ao componente filho (ex: o `LocalMailSession` através do método `getStore` disponibiliza o `Store`; o `LocalStoreInfo` através do método `getFolder` disponibiliza `Folder`, entre outros);
- ainda em fase de teste, todos os componentes precisam de ter noção se se precisam de recuperar ou não; para tal, todos os métodos remotos foram interceptados de forma a conseguir-se essa validação. Não é uma solução elegante, antes pelo contrário, mas a solução que parecia razoável (a utilização do *callback* `activate`) fazia a aplicação ficar em *deadlock*.

A estratégia adoptada para recuperação sofreu ao longo do tempo algumas evoluções.

Numa fase inicial pensou-se que a recuperação era apenas feita por necessidade (*on demand*), ou seja, à medida que os atributos "transientes" eram necessários eram recuperados. Com esta solução verificou-se que facilmente iria-se entrar em situações de acesso concorrente, algo que gera excepção naquele que realiza o último acesso. Este acesso concorrente inviabiliza a recuperação (ex: um componente é o *entry point* de uma chamada remota → recupera-se → evoca eventualmente um componente filho → este para recuperar precisa evocar pelo menos um método do componente pai → acesso concorrente).

Numa segunda abordagem, pensou-se numa recuperação local do componente e inserção de um método na interface local que permitisse desencadear a recuperação dos componentes filhos onde era passado como parâmetro desse mesmo método o atributo recuperado do `JavaMail`. Nesta abordagem tirava-se partido da geração de excepções de acesso concorrente para desfazer ciclos infinitos e utilizava-se a notificação de activação do componente (`activate`) gerada pelo servidor de aplicações para iniciar a recuperação. Acontece

que aquando por chamadas a métodos pertencentes às interfaces locais, numa situação de acesso concorrente, é gerada excepção; na utilização destes *callbacks* (`activate`) entra-se em *deadlock*, sem geração de excepção, o que inviabiliza esta solução. No entanto parece-nos que pode ser um defeito do servidor de aplicações porque nunca, em situação alguma, este deveria permitir uma situação de *deadlock* (deveria sempre gerar excepção).

Se a solução anterior não vier a ser viabilizada então será necessário realizar algo do género, ou seja, todos os métodos do componente terão que ser interceptados para verificar se o componente em causa precisa de ser recuperado. Nestas últimas estratégias é importante que um componente não evoque outro sem que valide se precisa ser recuperado, caso contrário vamos cair numa situação de acesso concorrente semelhante à primeira solução.

4.5 Integração de serviços

A integração dos serviços pode ser encarada de diferentes perspectivas: (1) integração ao nível da lógica de negócio, ou seja, ao nível da informação e funcionalidades disponibilizadas entre componentes; e (2) integração ao nível da apresentação. Ao nível da apresentação, a partilha de informação é realizada através de objectos de sessão *Web*. A esse nível a integração exige que os vários serviços apareçam de forma coerente e transparente aos utilizadores finais. A figura seguinte é um exemplo da integração conseguida dos vários serviços.

Figura 4.23: Interface Web do serviço myComs

A figura 4.23 ilustra a interface *Web* do *"myComs"*, podendo-se verificar que existem

cinco zonas perfeitamente distintas:

- topo e fundo, que são estáticas ao longo de toda a interacção do utilizador com o sistema;
- à esquerda, uma árvore de navegação no sistema, também ela estática;
- na zona central, a interface dedicada à apresentação do serviço de correio electrónico; e por último
- à direita a zona dedicada ao serviço de mensagens instantâneas e notificações de presença.

Nesta última zona (à direita) pode-se verificar ainda que no seu topo permite que o utilizador altere o seu estado de presença; imediatamente abaixo verifique a sua lista de amigos, com os seus estados de presença respectivos; e por último, uma janela com as mensagens instantâneas recebidas. (Para o utilizador iniciar comunicação com um dos seus amigos basta premir sobre o nome do amigo correspondente.)

Esta divisão de apresentações foi conseguida recorrendo a *framesets* do HTML. Pode ser consultado com maior detalhe na secção 4.6.2. Com esta técnica as actualizações, a acontecerem, serão parciais apenas ao nível do correio electrónico ou ao nível das mensagens instantâneas e de presença não envolvendo a actualização da página total.

4.6 Interfaces homem–máquina

Como referido na secção 3.1, é objectivo desta aplicação poder ser disponibilizada segundo várias interfaces homem–máquina. No entanto, tratando-se de uma versão protótipo foi seleccionada apenas uma das várias interfaces possíveis, a interface *Web*. Estão, no entanto, a decorrer outros trabalhos paralelos que têm como objectivo disponibilizar outras interfaces, nomeadamente interface voz (*voiceXML*) e interface móvel (J2ME).

Ainda antes de se entrar nas secções de interfaces homem–máquina convém apresentar as possibilidades de navegação da aplicação.

4.6.1 Possibilidades de navegação

Esta secção ilustra as possibilidades de navegação na aplicação *"myComs"*. Para isso a figura 4.24 ilustra um diagrama de actividades possíveis na aplicação.

Existem dois blocos distintos que correspondem a dois estados distintos do utilizador: anónimo e autenticado. Enquanto anónimo o utilizador, ao contrário do que a figura deixa parecer, está limitado nas opções de navegação que pode efectuar, nomeadamente está limitado à introdução, à página de ajuda, e à página de autenticação, onde poderá registar-se ou autenticar-se. Uma vez autenticado poderá efectuar todas as operações

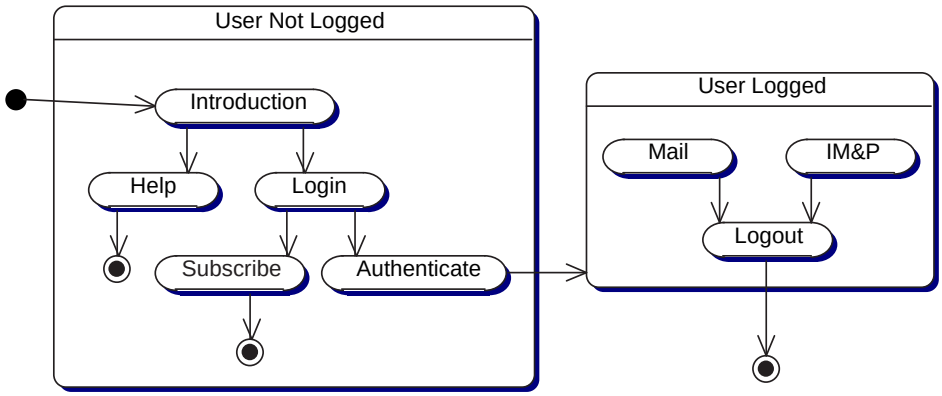

Figura 4.24: Diagrama de actividades possíveis na aplicação

relativas à sua caixa de correio, todas as operações relativas às mensagens instantâneas e notificações de presença, e eventualmente desligar-se do sistema. São inúmeras as acções que se podem realizar em cada um dos serviços e por conseguinte não vale a pena repetir essa enumeração até porque ao longo do documento as várias funcionalidades já foram descritas.

A próxima secção ilustra as várias interfaces possíveis no utilizador, seus significados, mediante interacção do utilizador com o sistema.

4.6.2 Página de entrada

Para iniciar uma sessão com o *"myComs"* (supondo que o dito sistema se encontra instalado e configurado adequadamente na máquina `server_name` e no porto `port_number`) o utilizador deverá aceder ao seguinte endereço:

```
http://server_name:port_number/myComs/
```

A partir deste endereço são geradas dinamicamente as páginas HTML que permitem a interacção homem–máquina correspondente.

Numa fase inicial, antes de qualquer autenticação o utilizador pode apenas consultar informação genérica relacionada com a aplicação.

A figura 4.25 que se segue representa uma imagem real da aplicação e uma imagem esquematizada da imagem real realçando as diferentes zonas na página.

O *layout* utilizado na apresentação da informação é comum em todas as interacções do utilizador com o sistema. As zonas topo, fundo, e explorador são zonas que se mantêm constantes ao longo da sessão. A zona principal é dinâmica e altera-se função das interacções do utilizador. Actualizando apenas a zona principal permite redução da largura de banda ocupada na rede. Para conseguir esta funcionalidade foram usadas *tags* HTML

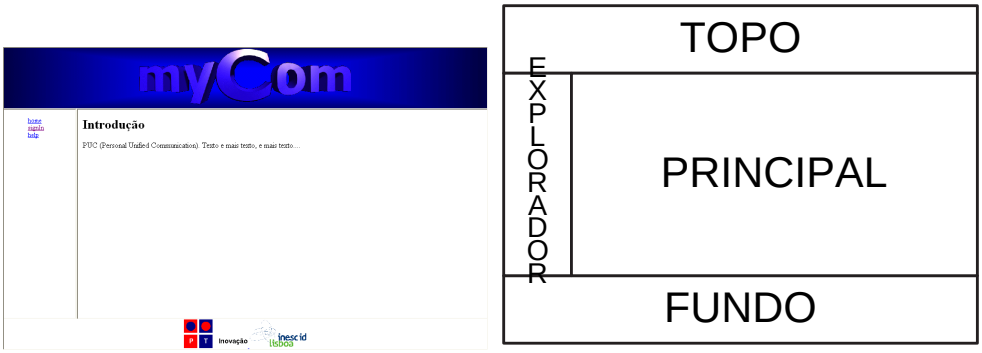

Figura 4.25: Página de entrada e respectiva disposição

framesets e *frames*, conforme se pode ver no excerto da página inicial, correspondente à listagem 4.16.

Listagem 4.16: Página inicial em HTML

```
root/view/index.jsp
<HTML>
  <HEAD>
    <TITLE>MyComs</TITLE>
    ...
  </HEAD>
  <FRAMESET name='frameset1' rows='20%, 70%, 10%' framespacing='1'
            frameborder='0' onunload='window.unload()'>
    <FRAME name='topFrame' src='./static/topBanner.html' class='top' scrolling='no'¿
    <FRAMESET name='frameset2' cols='15%, 85%' frameborder='1'>
      <FRAME name='explorerFrame' src='./explorer/explorer.jsp' class='explorer'¿
      <FRAME name='mainFrame' src='./static/intro.html' class='main'>
    </FRAMESET>
    <FRAME name='bottomFrame' src='./static/bottomBanner.html' class='bottom' scrolling='no'>
    ...
  </FRAMESET>
</HTML>
```

4.6.3 Página de autenticação

Esta página é retornada ao utilizador quando este, no quadro do *explorer*, carrega no *link* *"sign In"*. Nesta página o utilizador tem duas opções: (1) autentica-se no sistema; (2) carrega no *link* para se subscrever. Qualquer erro que exista no processo de autenticação é mostrado na própria página, no topo, com a descrição do erro. Quanto à subscrição é referida na secção imediatamente a seguir.

4.6.4 Página de subscrição

A página de subscrição não é mais do que um formulário que deve ser preenchido pelo utilizador. A figura 4.27 exemplifica a página correspondente ao formulário a preencher

Figura 4.26: Página de autenticação

pelo utilizador. Alguns itens são obrigatórios, outros são pura informação opcional. Existem dois tipos de validação: (1) como já referido, validação de campos obrigatórios; (2) validação do *login* e *password* do utilizador. A segunda parte da validação é realizada pelo componente empresarial, e aí não há dúvidas que deve ser ele o responsável por tal validação. É que para a primeira parte da validação, campos obrigatórios, pode-se admitir vários sítios onde o fazer: (1) imediatamente no utilizador, à custa de *script*; (2) no *JavaBean* na componente *Web*; (3) no componente empresarial. No primeiro, em caso de erro, evita-se tráfego desnecessário, mas requer a utilização de *script* que em alguns ambientes móveis poderá não existir; para além disso a própria página ficaria mais pesada e portanto se imaginarmos que na maior parte das situações não existe erro então há tráfego desnecessário. A terceira pareceu-me, embora provavelmente a conceptualmente mais correcta, descabida por estar a promover demasiado tráfego numa validação que pode ser feita logo no local. Portanto foi utilizada a segunda hipótese que fica no meio termo, ou seja, a validação é feita na componente *Web*. Pode ser encarado como trabalho futuro, e admitindo que não é factor preocupante o tráfego na rede, a implementação das duas primeiras soluções, ou seja, faz logo a validação no utilizador se tiver recursos para isso, aumentando a eficiência em caso de erro; se não tiver, essa validação é feita na componente *Web*.

Após submeter o formulário, em caso de erro é retornada a mesma página aparecendo a mensagem de erro sob o campo onde ocorreu. Aparece logo todos os erros cometidos e não apenas um de cada vez. Se o formulário não contiver erros é retornada uma outra página com a confirmação da informação submetida e aceite.

Figura 4.27: Página de subscrição

4.6.5 Página após autenticação

Após autenticado o utilizador é livre de realizar todas as operações possíveis a cada um dos serviços. Poderia aqui haver algum tipo de controlo de acesso a serviços função da subscrição dos mesmos. Com certeza a isso corresponderia diferentes custos ao utilizador. Mais uma vez seria assunto para trabalho futuro. Neste protótipo, o utilizador uma vez subscrito tem acesso a todos os serviços disponíveis na aplicação *"myComs"*. Todos eles aparecem integrados na mesma apresentação.

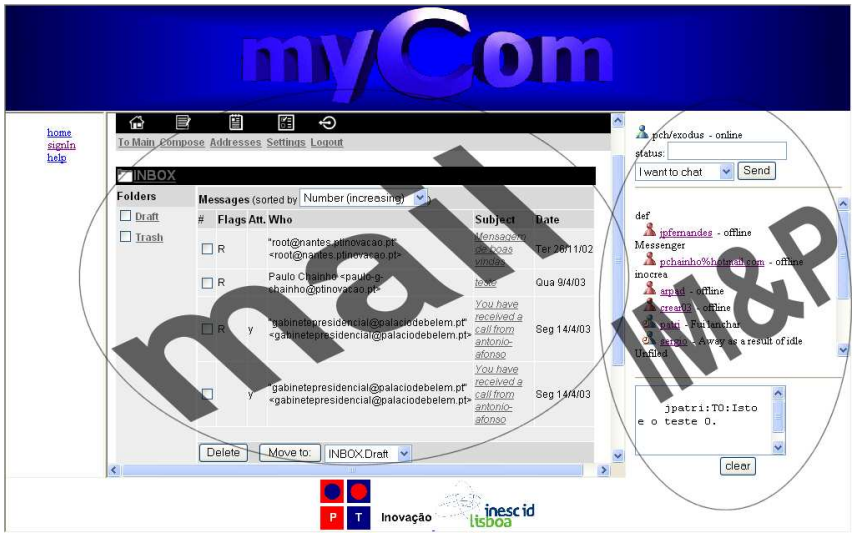

Figura 4.28: Página de serviços

Neste momento conforme se pode constatar na figura 4.28 apenas estão dois serviços disponíveis: (1) o serviço de correio electrónico; (2) o serviço de mensagens instantâneas e notificações de presença. Foram considerados igualmente como serviços o processo de

subscrição e o processo de autenticação, no entanto, para o utilizador é natural que não os considere como tal, uma vez que tem pouca interacção com eles, ou porque correspondem ambos a um processo natural na utilização de uma aplicação.

No serviço de correio electrónico pode-se realizar todas as operações associadas a uma caixa de correio electrónico. As operações podem-se dividir em quatro áreas de acção: pastas; mensagens; lista de contactos; e preferências da caixa de correio. A tabela 4.1 enumera todas as operações dentro de cada área.

Áreas de acção	**Operações possíveis**
Pastas	Criar; remover; seleccionar para ver resumo de mensagens
Mensagens	Ver detalhada; remover; mover entre pastas; ver ficheiros acoplados à mensagem; mudar os atributos de recepção de uma mensagem; compor uma mensagem nova e enviá-la.
Lista de contactos	Criar; remover; editar.
Preferências	Tudo o que corresponde à configuração da caixa de correio.

Tabela 4.1: Operações possíveis no serviço correio electrónico

Sem dúvida que os dois primeiros itens são os que representam na realidade o serviço de correio electrónico. Os outros dois são mais valias herdados da versão original JWMA. Inclusive, a lista de contactos, no contexto PUC, é considerada como outro serviço de mais alto nível, o *"myContacts"*, que está a ser realizado por um outro grupo de trabalho. Na versão herdada, tanto a lista de contactos como as preferências estão a ser guardadas no sistema de ficheiros local, não promovendo persistência ao longo de várias distribuições da aplicação. Na versão final, as soluções continuam as mesmas, podendo então passar como trabalho futuro a criação de componentes que tratassem e representassem esta informação. Ainda assim, não esquecer que a gestão destas componentes são da responsabilidade de outros dois serviços do PUC que estão a ser desenvolvidos em paralelo: *"myAgenda"* e *"myContacts"*.

No serviço de mensagens instantâneas a apresentação está dividida em três zonas. Na realidade está dividida em quatro zonas, mas visíveis são apenas as três primeiras; a última corresponde à *Applet* que não tem representação gráfica (ver secção 4.2.3).

Listagem 4.17: Página inicial em HTML do serviço de mensagens instantâneas

```
1 userAgent/view/index.jsp
```

```
<HTML>
  ...
  <FRAMESET rows='100, *, 220, 0'>
    <FRAME name='myPresenceFrame'
                        src='./userAgent/xmppMyPresence.jsp'
                        noresize scrolling=no frameborder=0>
    <FRAME name='rosterFrame' src='./userAgent/xmppRoster.jsp'
                        scrolling=yes frameborder=0>
    <FRAME name='messagesFrame' src='./userAgent/xmppMessages.jsp'
                scrolling=no frameborder=0>
    <FRAME name='notifierFrame' src='./userAgent/xmppNotifier.jsp'
                        noresize scrolling=no frameborder=0>
    ...
  </FRAMESET>
</HTML>
```

Pode-se verificar na listagem 4.17 que o espaço destinado à última *frame* tem dimensão 0. As três zonas visíveis correspondem: estado do utilizador; lista de amigos; zona de recepção de mensagens. As acções que se poderão realizar em cada uma das áreas estão mostradas na tabela que se segue.

Áreas de acção	**Operações possíveis**
Estado do utilizador	Envio do estado de presença do utilizador. Um de cinco; Envio de estado no formato texto livre.
Lista de amigos	Actualização automática do estado de presença da lista de amigos; Iniciação de comunicação.
Mensagens recebidas	Recepção automática de mensagens; Limpar a zona de mensagens.

Tabela 4.2: Operações possíveis no serviço mensagens instantâneas e notificações de presença

Na tabela 4.2 está dito que permite enviar um de cinco estados de presença. Na realidade os estados de presença são apenas dois: um de indisponibilidade; outro de disponibilidade. Dentro da disponibilidade é que há quatro modos de presença: *chat*; *away*; *extended away*; *dnd* (*do not disturb*). Algo inovador num serviço deste tipo é permitir juntamente com o estado de presença ainda mandar um texto livre descrevendo o estado. Por exemplo: juntamente com o estado *extended away* escrever "fui à pesca".

A lista de amigos pode estar organizada por grupos. As funcionalidades de criar, remover, configurar grupos não estão implementadas neste serviço, no entanto, sabe mostrar o grupo caso exista. Para além dos grupos, um amigo é mostrado sob a forma: boneco que revela o seu modo de presença; nome pelo qual quer ser conhecido (*nickname*); estado associado ao modo de presença.

As mensagens recebidas estão divididas em três campos separados pelo carácter ":": nome do remetente; assunto; mensagem.

Através do nome do amigo, na lista de amigos, é permitido ao utilizador iniciar comunicação com o amigo. Aí abrirá uma nova janela de *browser* onde terá oportunidade de optar por um meio de comunicação. Neste momento estão previstos três: enviar mensagem por correio electrónico; enviar mensagem instantânea; clicar para iniciar ligação telefónica. Na realidade só um está disponível neste protótipo: o envio de mensagens instantâneas. O envio de correio electrónico não foi feito uma vez que já estava disponível na zona de correio electrónico; o clicar para iniciar ligação telefónica foi desenvolvido por um elemento da PTInovação. Novamente, esta integração poderia ser inserida como trabalho futuro. Uma vez escolhida a opção de envio de mensagem instantânea aparece nova janela de *browser*, fechando a anterior, onde o utilizador poderá acrescentar ao nome do amigo o assunto e a mensagem a enviar.

Capítulo 5

Conclusões

Este projecto teve início no ano 2002 com uma perspectiva de duração de um ano. Portanto deveria estar concluído no final do mesmo ano. Na realidade essa data não foi cumprida e o projecto teve apenas conclusão já em meados do ano 2003. Assume-se que o projecto concluiu com a entrega do relatório final que aconteceu em Novembro de 2003; no entanto a concepção e implementação do sistema acabou uns meses antes (Agosto). Pelo meio, em Maio de 2003, houve a participação, a convite da *Sun Microsystems*, na conferência *Java On Demand 2003*, organizada pela mesma empresa.

Entre diferentes motivos, uma das razões de tal atraso deveu-se ao facto da tese se enquadrar no contexto de um projecto concreto, em que as tecnologias que deram suporte à realização deste sistema foram novidade, e como tal existiu uma curva de aprendizagem acentuada, e que no final não deixou de influenciar o tempo total da realização do projecto. Para além das tecnologias houve também um esforço adicional em termos de prospecção, instalação e utilização de diversas ferramentas de suporte e de desenvolvimento, nomeadamente:

- Tecnologias e especificações: J2EE, em particular Java Servlets, JSP, EJB, JavaMail, JNDI, JDBC, LDAP e protocolo Jabber.
- Ferramentas de suporte: Servidor de Servlets (Tomcat/Catalina) ou Jetty; Servidor de Correio Electrónico (SMTP – qmail; IMAP – courier); Servidor de IM (jabberd); Servidor de EJB (JBoss); Servidor LDAP (openLdap); pacote JWMA; e pacote JabberBeans.
- Ferramentas de desenvolvimento: IDE (netbeans); ant.

Ainda em relação às tecnologias e ferramentas deve-se referir que o sistema foi realizado sobre tecnologias que se baseiam no paradigma código livre. Esta decisão inicial do projecto, apesar de apresentar as vantagens reconhecidas no movimento código livre, apresentou entre outros os seguintes problemas: documentação fraca ou praticamente

inexistente (e.g., no caso do JBoss); e ferramentas de desenvolvimento de software e de administração dos servidores fracamente integradas e pouco produtivas.

Para além das considerações que confirmam a acentuada curva de aprendizagem tecnológica exigida - e que justificam o atraso e as dificuldades no desenvolvimento do protótipo apresentado - , outros aspectos importam referir, designadamente:

- A especificação J2EE (versão 1.3) e a plataforma JBoss utilizada (em particular a versão 3.0) encontram-se já num bom nível de arquitectura, robustez, desempenho e flexibilidade.
- Ultrapassada a referida curva de aprendizagem e a definição de um ambiente de desenvolvimento integrado (e.g., com as ferramentas acima referidas), o desenvolvimento de software torna-se razoavelmente produtivo.
- Em particular na área do negócio das telecomunicações, a escolha de referência Java e J2EE é adequada tendo em conta as múltiplas iniciativas emergentes, tais como JAIN[1] [51] ou SIP-Servlet[2] [50].

No capítulo 1 foi referido que para que uma aplicação com interface com o utilizador tivesse sucesso, e no âmbito das operadoras de telecomunicações, era importante que conseguisse introduzir contexto nas suas operações. Os próximos parágrafos avaliam se o sistema desenvolvido é baseado em contexto ou não, nomeadamente, se consegue dar resposta aos cinco requisitos que são considerados como características de um sistema baseado em contexto[3]: "localização", "quem", "quando", "quê", "como".

Localização: Pensando em serviços baseados em localização tradicionais, tais como o sistema reagir à localização física do utilizador notificando-o, por exemplo, com uma farmácia próxima do local onde se encontra ou um restaurante com determinada especialidade gastronómica, o sistema não apresenta esta capacidade. Não apresenta porque não tem recursos para saber a localização física do utilizador. Para poder ter serviços com estas características era necessário dar ao sistema a indicação do local físico onde o utilizador se encontra. O sistema pode adquirir essa informação por diferentes meios: ser o próprio utilizador a fornecer essa informação, por um GPS[4], ou por informação da própria rede móvel. Para qualquer uma das duas últimas soluções o utilizador precisaria de equipamento adicional, nomeadamente ou um receptor GPS ou um telemóvel.

[1] *JAIN Service Logic Execution Environment.* Conjunto de APIs que permite portabilidade de serviços, independência da rede, e desenvolvimento aberto para redes de comunicações sem fios, dados e telefonia.

[2] Conjunto de APIs que permitem que aplicações SIP (*Session Initiation Protocol*) sejam distribuídas e geridas segundo o modelo utilizado nas *Servlets*.

[3] Conhecido também pelos 5 W's: *Where*, *Who*, *When*, *What*, *hoW*.

[4] *Global Positioning System*

Quem: O sistema tem características que permitem responder perfeitamente ao tópico "quem". Por exemplo associando o *login* de um utilizador a uma conta, o utilizador apenas tem acesso à sua conta e não de outro. Por exemplo um utilizador consegue apenas ver as mensagens recebidas por correio electrónico da sua conta; nunca conseguirá ver mensagens pertencentes a outras contas (i.e. contas pertencentes a outros utilizadores).

Quando: Quanto ao "quando", o sistema também possui exemplos que mostram saber reagir a esta característica. Por exemplo, a notificação de uma marcação na agenda. Esta notificação é uma consequência temporal; acontece quando for atingido determinado dia, determinada hora, revelando noção temporal.

Quê: Em relação ao "quê" o sistema não apresenta características que se possa dizer que saiba responder a este tópico. Na realidade, também existe alguma dificuldade em caracterizar o que é que é saber responder a "quê".

Como: Por último, o "como" pode-se confundir com o "onde" na medida que poder-se-ía imaginar por exemplo o dispositivo de "onde" se acede ao sistema. Na realidade, o tipo de dispositivo com que se acede ao sistema enquadra-se em "como" e neste sentido o sistema suporta diferentes medias e como tal consegue responder a esta característica.

Destes parágrafos podemos concluir que o sistema é baseado em contexto.

Uma outra particularidade que deve ser também analisada no nosso sistema é a aquisição de contexto. A aquisição de contexto pode ser realizada de duas formas possíveis: manualmente através da inserção pelo próprio utilizador; automaticamente através de mecanismos que permitam fazer essa aquisição automática. Dentro das duas possibilidades, a automática torna o sistema mais rico e completo e mais inteligente do que se a aquisição tiver que ser feita manualmente. No nosso caso concreto, ou o sistema já tem pelo próprio tipo de serviço, o contexto integrado, e portanto sabe reagir a esse contexto, ou então terá que ser dado manualmente. Neste momento, o sistema não tem mecanismos que permitam filtrar automaticamente contexto das acções do utilizador. É sem dúvida um ponto de expansão do nosso sistema.

Outras considerações que devem ser igualmente tomadas, uma vez que serviram de orientação à realização deste sistema, referem-se aos princípios enumerados na secção 3.2:

- Abstracção das tecnologias de rede;
- Ubiquidade de tipo de acesso e de tipo de media;
- Elevado número de utilizadores e de transacções;

- Gestão e subscrição flexível e dinâmica de serviços;
- Tecnologia Java e Open-Source.

Abstracção das tecnologias de rede. Este princípio foi conseguido. Tendo sido utilizadas bibliotecas que abstraem o meio de comunicação libertou o programador do conhecimento da tecnologia de rede utilizada. A maior parte das bibliotecas utilizadas são standards definidos por entidades e organizações competentes na área da Internet. À partida alterando alguma tecnologia de rede será suficiente utilizar a biblioteca adequada não sendo por isso necessário alterar qualquer coisa no sistema.

Ubiquidade de tipo de acesso e de tipo de media. Este princípio foi também conseguido pois foram desenvolvidas uma série de interfaces (HTML, j2me, voiceXML) que permitem ao utilizador aceder aos serviços através de diferentes equipamentos terminais (PC, telemóvel, PDA, telefone).

Elevado número de utilizadores e de transacções. O estudo e análise deste tópico foi objecto de um trabalho final de curso [8] no Instituto Superior Técnico. À custa de um nó com várias máquinas que mantém réplicas do estado de sessão do sistema foi possível responder a centenas de pedidos simultâneos efectuados pelos clientes. Para além de melhorar a capacidade de resposta, a criação de um nó com várias máquinas, veio permitir igualmente que o sistema se torne tolerante a falhas, ou seja, sempre que uma máquina falhe, outra encarrega-se de responder ao pedido realizado pelo cliente uma vez que mantém o mesmo estado de sessão da máquina que falhou.

Gestão e subscrição flexível e dinâmica de serviços. Princípio parcialmente conseguido. Um utilizador consegue-se registar dinamicamente no sistema, no entanto não foi desenvolvida a possibilidade de configuração e gestão dos serviços subscritos. Também não houve essa necessidade uma vez que não existem assim tantos serviços. No fundo a componente de gestão não foi implementada. Com a evolução do sistema, e portanto, com a criação de novos serviços passa já a ser aconselhável existir alguma preocupação nesta funcionalidade. No fundo, para este sistema ser comercial é importante que o utilizador possa ter a opção de decidir quais os serviços que pretende, até porque isso terá consequência directa no valor que a empresa fornecedora dos serviços irá cobrar.

Tecnologia Java e Open-Source. Por último, este princípio também foi seguido. Todo o sistema foi desenvolvido utilizando a linguagem Java, o que torna o sistema independente da plataforma computacional. Por sua vez para implementação dos serviços recorreu-se única e exclusivamente a bibliotecas código livre. Aliás, já foi referido que

as únicas ferramentas proprietárias utilizadas foram para a realização deste documento, nomeadamente ferramentas de edição de imagens e de tabelas.

No capítulo 2, secção 2.2, foram analisados uma série de sistemas similares e identificadas várias características nesses mesmos sistemas. Inclusive, no final da secção foi apresentada a tabela 2.3 onde se resumiu e onde se analisou essas características.

Enquanto perspectiva global do sistema, e uma vez que se tratam de aplicações actuais, e que disponibilizam serviços actuais, é relevante comparar as características e especificações encontradas nessas aplicações com as encontradas no sistema criado no contexto deste projecto. A tabela 5.1 é uma replicação da tabela 2.3 com mais uma coluna correspondente às características encontradas no nosso sistema.

A tabela 5.1 está organizada da seguinte forma: no eixo das abcissas estão enumeradas as aplicações analisadas; no eixo das ordenadas estão identificados tópicos para cada uma das características principais PUC e PIM; ainda estão identificados alguns tópicos, que por não se enquadrarem no conceito PUC ou PIM, foram reunidos em tópicos genéricos; na última coluna estão todas as características correspondentes ao sistema criado no contexto deste projecto.

			Outlook	Elefante	OutSystems	Weblicon	Hambo	Chandler	PUC
Funcionalidades	**PIM**	Agendas	x	x		x	x	x	x
		Contactos	x			x	x	x	x
		Tarefas	x	x		x	x	x	x
		Notas	x						
	PUC	Email	x		x	x	x	x	**X**
		IM					x		**X**
		SMS			x	x	x		x
		MMS			x	x			
		VoIP				x			x
Outros Aspectos		Interfaces	PC	browser, WAP	browser, WAP	browser, WAP, j2me, i-mode, voiceXML	browser, WAP, i-mode, voiceXML	Não está claro	browser, WAP, j2me, voiceXML
		Web services			x	x			
		Colaboração	x					x	
		Notificação	local	email, SMS			email, SMS		
		Plataforma	Windows		.Net	j2ee	Java	Python, wxWindows	j2ee
		Código livre					x	x	x
		Sincronização	x	HotSync		SyncML	x	x	x

Tabela 5.1: Características das aplicações

Olhando de uma forma geral verificamos que o nosso sistema está bem completo comparado com os outros sistemas. Convém referir que as características marcadas no nosso sistema correspondem ao sistema global, portanto, correspondem ao conjunto dos quatro trabalhos realizados neste projecto, não apenas ao que foi desenvolvido no contexto desta

tese. Os serviços que foram realizados no contexto desta tese encontram-se demarcados dos restantes apresentando um foco mais forte.

Onde não existiu grande investimento foi na parte de disponibilizar os serviços via *Web services*, na colaboração e na notificação. Uma interface por *Web services* é importante quando pretendemos recorrer aos serviços para integração noutras plataformas. Não foi o caso, uma vez que tudo foi desenvolvido baseado na linguagem Java e foi desenvolvida uma solução completa, desde a concepção do próprio serviço até à sua apresentação ao utilizador final. De qualquer forma seria sempre uma mais valia disponibilizar mais esta interface que, com a nova especificação J2EE 1.4, fica inclusive facilitado o seu desenvolvimento uma vez que passa a fazer parte da especificação dos EJBs a componente dos *Web services*. A parte da colaboração e notificação era uma questão de criar serviços onde constassem estas características. Para a notificação, por exemplo, o suporte está criado com a disponibilização dos serviços de correio electrónico, de mensagens instantâneas e com a noção de presença do utilizador. Para a colaboração poder-se-á dizer que também existe algum suporte com a noção de grupos onde diferentes pessoas podem por exemplo colaborar na marcação de uma reunião.

Na componente SMS apenas é possível enviar mensagens e apenas se o utilizador estiver ligado ao sistema através do telemóvel. Através da interface *browser* essa funcionalidade não está disponível.

Nas interfaces está presente a interface WAP, ou seja, poder aceder aos serviços através de um *browser* instalado num dispositivo móvel. Na realidade, o sistema não teve alterações de fundo para suportar esta interface uma vez que tanto um *browser* num PC como um *browser* num dispositivo móvel interpretam o mesmo HTML. Acontece é que para minimizar o peso num dispositivo móvel nem todas as especificações para interpretação de HTML estão implementadas (nomeadamente aquelas que não são standards, por exemplo a *tag frameset*), ou por exemplo, um *browser* num dispositivo móvel pode não conseguir executar código Java. Estas situações foram previstas com soluções alternativas. Por exemplo no caso do código Java que corre do lado do cliente para o serviço das mensagens instantâneas. Se o *browser* não suportar a execução de uma máquina virtual entra automaticamente em funcionamento a primeira solução estudada, a solução por *polling* que não necessita de código Java a correr do lado do cliente. Estas soluções estão detalhadas na secção 4.4.3. Quanto à apresentação não houve preocupação, ou seja, é natural que num *browser* de um dispositivo móvel as páginas pareçam menos elegantes que num *browser* de um PC.

Na parte da sincronização foi criado um protocolo específico para o efeito [15]. Esse protocolo permite a sincronização em ambos os sentidos, ou seja, a actualização de informação tanto do dispositivo móvel para a base de dados central como também da base de dados central para o dispositivo móvel. Esta funcionalidade ainda está incompleta

uma vez que não resolve conflitos de sobreposição de eventos. Consideram ainda como trabalho futuro a utilização de um standard, por exemplo *SyncML*, para sincronização. A vantagem é óbvia porque passa a conseguir sincronizar não apenas com o nosso sistema mas também com todos os outros sistemas que suportem este standard de sincronização.

Devido à imaturidade na tecnologia é natural que algumas das opções tomadas na altura fossem diferentes das que seriam tomadas hoje se o mesmo trabalho estivesse a ser realizado actualmente. Tal facto deve-se à distância temporal que existiu e por conseguinte o afastamento do projecto, e portanto o ganho de uma visão mais global do próprio sistema. Algumas decisões foram também condicionadas à imposição de prazos que existiam, ou à utilização de determinadas tecnologias (e.g., a utilização do JWMA para suporte do cliente de correio electrónico). Por estas e por outras razões é razoável que exista um tópico onde se identifique o que é que poderia ser feito como trabalho futuro.

5.1 Trabalho Futuro

Pode-se prever o trabalho futuro segundo duas perspectivas: (1) trabalho futuro focado nas alterações, modificações ou evoluções concretas sobre o que foi feito; tem carácter mais tecnológico, mais incisivo; ou (2) trabalho futuro numa perspectiva global de aplicação, e por conseguinte o que é que este sistema poderá vir a oferecer, adicionalmente, no futuro; por exemplo a identificação de outros serviços.

Segundo a primeira perspectiva, no decorrer do capítulo 4 foram, sempre que justificável, identificadas possíveis alterações às opções tomadas. Nesse sentido a tabela 5.2 reúne a generalidade dos pontos onde existem esses comentários. Na primeira coluna está identificada a secção onde aparece a referência ao trabalho futuro; a segunda coluna corresponde ao resumo do texto que aparece no documento. Pelo facto de ser um resumo, esta tabela não dispensa a consulta do referido texto.

Segundo a segunda perspectiva, como trabalho futuro e enquanto serviços a disponibilizar, a secção 3.5 descreve alguns conceitos importantes a ter em conta para a criação de serviços inovadores. De seguida estão enumerados os conceitos identificados nessa mesma secção:

- Integração de sensores no sistema para fornecer informação na forma de contexto;
- Identificadores únicos para que tudo possa ser conhecido e possa existir interacção;
- Gestão de informação. Dada a diversidade de informação são exigidas regras coerentes na gestão da mesma (ver o sistema *"Chandler"* - secção 2.2.4);

Secção	Trabalho Futuro
Secção 3.8.3	observar as extensões ao protocolo Jabber e implementar aquelas que pareçam razoáveis
Secção 4.2.1	utilização de outros tipos de autenticação, nomeadamente a API JAAS, a utilização de um canal seguro HTTP/SSL, ou recorrer a processos de autenticação fornecidos pela plataforma
Secção 4.2.1	utilização em todos os módulos da biblioteca serviceLocator
Secção 4.3.1	utilização de um componente Entity para representar uma entrada no servidor LDAP
Secção 4.3.2	existir apenas uma única interface remota que representasse todo o serviço de mail em vez de cada componente ter uma interface remota (implementação do padrão session facade)
Secção 4.3.3	utilização de um componente empresarial do tipo MDB (Message Driven Bean) ou mesmo utilização do JCA em vez da biblioteca XMPP
Secção 4.6.4	implementação da validação directamente no utilizador se tiver recursos para isso; se não tiver, essa validação continua a ser feita na componente Web
Secção 4.6.5	controlo de acesso a serviços função da subscrição dos mesmos
Secção 4.6.5	criação de componentes que dessem suporte à persistência da lista de contactos e preferências do utilizador
Secção 4.6.5	integração dos três serviços de iniciar comunicação: enviar mail; enviar uma mensagem instantânea; iniciar uma ligação telefónica

Tabela 5.2: Trabalho futuro no contexto da implementação

- Gestão de grupos para facilitar a gestão do acesso a informação partilhada;
- Grupos de colaboração igualmente para facilitar a gestão do acesso a informação partilhada;
- Sistema activo, ou seja um sistema que não se limite apenas a reagir às acções do utilizador.

Para além dos conceitos identificados e que estão contextualizados na secção 3.5, foi apresentado nessa mesma secção cenários possíveis para a descrição de alguns novos serviços. Os exemplos referidos mostram a potencialidade dos serviços que podem ser criados tendo por base os serviços de comunicação e de gestão de informação pessoal.

Conforme se pode verificar, e dado os conceitos e exemplos, reforço a ideia que, para a concepção de novos serviços, depende apenas de **imaginação**.

Apêndice A

Instalação e Configuração

Esta secção descreve toda a parte de distribuição da aplicação, enumera os recursos utilizados, e descreve alguns aspectos específicos de configuração.

A.1 Recursos e ferramentas

Todas as ferramentas de desenvolvimento e de suporte à aplicação utilizadas têm uma característica comum. Todas são código aberto. É de realçar esta característica uma vez que permite realizar todo o desenvolvimento para gerar uma aplicação final, para além de suportar a sua execução. Digamos que em todo o desenvolvimento as únicas ferramentas que não são código aberto são as que estão a ser utilizadas para realizar este documento; estamos a falar do editor de texto e editores de imagem.

Enumeração dos recursos e tecnologia:

- JBOSS 3.x.x - motor J2EE [49]
- Jetty - servidor *Web*, servidor de *Servlets* e *Java Server Pages* [66]
- NetBeans - ambiente integrado e edição [68]
- Ant - ferramenta integradora de desenvolvimento; em teoria é a geração actual da conhecida ferramenta *make* [82]
- Servidores/ protocolos
    - openLDAP - servidor LDAP (*Lightweight Directory Access Protocol*) [71]
    - qmail - SMTP (*Simple Mail Transport Protocol*) [4]
    - courier - IMAP (*Internet Message Access Protocol*) [84]
    - jabberd - *Instant Message and Presence server* [47]
- Bibliotecas
    - as bibliotecas relativas à especificação J2EE

- no contexto do serviço de correio electrónico: `javamail.jar`; `activation.jar` (*Activation Framework*); `castorxml.jar` (*Castor XML*); `xerces.jar` (*Xerces-J*); e `jakarta-oro.jar` (*Jakarta-ORO*)
- no contexto do serviço IM: `jabberbeans.jar` (*JabberBeans*)

A.2 Organização da aplicação de distribuição

A figura A.1 mostra que a aplicação a distribuir é representada por um único ficheiro com extensão `.ear` (*enterprise archive*). Este ficheiro contém todos os ficheiros EJB JAR e todos os ficheiros JAR componentes *Web*, que por norma apresentam a extensão `.war`. Todos juntos formam o que é denominado uma aplicação J2EE. A aplicação contém ainda o seu próprio descritor XML onde descreve os módulos envolvidos (EJB e componentes *Web*) para além de outros elementos como por exemplo ícones, descrições, entre outros. Nesta aplicação ainda existe uma directoria `library` com os *jars* auxiliares dos EJBs e o ficheiro `jndi.properties` onde estão definidas variáveis de ambiente para os componentes empresariais. Foi uma das formas encontradas para permitir que um EJB consiga aceder a um ficheiro com propriedades; foi utilizada outro tipo de solução no componente de correio electrónico, onde não é obrigatório que seja um ficheiro com o nome `jndi.properties`, nem tão pouco que seja de propriedades e onde pode ter um caminho específico que não a raíz da aplicação. Estes problemas e soluções adoptadas estão descritos nas secções 4.4.1 e 4.4.2.

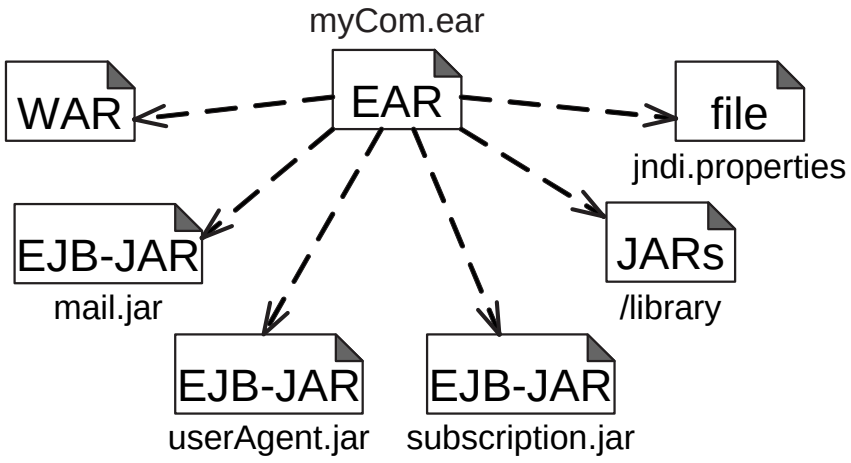

Figura A.1: Componentes da aplicação a distribuir

A.3 Ferramenta Ant

Como já foi referido na secção A.1 a ferramenta *Ant* é uma ferramenta integradora de desenvolvimento, digamos a nova geração da conhecida ferramenta *make*. É uma ferramenta utilizada em todo o processo desde o código fonte até à geração da aplicação de distribuição. Todo o processo corresponde a compilar o código fonte, corresponde a empacotar em *jars* os diferentes componentes, a criar a estrutura da aplicação de distribuição, e por último a empacotar a aplicação empresarial pronta para fazer a distribuição. Inclusive, permite fazer a própria distribuição. O *Ant* para além de automatizar todo este processo tem a filosofia do *make*: só faz se precisar.

O *Ant* tem o conceito de *target*. Ou seja, permite na sua evocação especificar o objectivo pretendido, que nem sempre poderá ser realizar a distribuição. É neste sentido que aparece esta secção: enumerar os vários *targets* e explicar as suas acções.

O *Ant* tem como *input* um ficheiro em XML onde estão descritos os vários *targets*. Por norma esse ficheiro tem o nome `build.xml`; poderá ter outro nome e nesse caso deverá ser especificado na linha de comando na chamada ao *Ant*; caso contrário, por omissão, o *Ant* procura um ficheiro `build.xml`.

O ficheiro em XML não é mais do que um conjunto de *targets*. Em cada *target* aparece o seu nome, os nomes dos *targets* de que depende e no seu interior está descrito o que deve realizar (exemplo na listagem A.1).

Listagem A.1: Exemplo da definição de um target no ficheiro build.xml

```
/PUCProject/myComs/build.xml
...
  <!-- ================================================================== -->
  <!-- Create ear application                                             -->
  <!-- ================================================================== -->
  <target name="ear" depends="init,war,subscription,userAgent,mail">
    <mkdir dir="${dist}${/}library" />

    <copy todir="${dist}${/}library">
      <fileset dir="${lib}" />
    </copy>

    <jar jarfile="${deploy}${/}myComs.ear">
      <fileset dir="${dist}"
        includes="*.jar,*.war,library${/}**,jndi.properties" />
      <metainf dir="${descriptors}${/}application"  />
    </jar>
  </target>
...
```

Na directoria do projecto existe diferentes directorias para o serviço de correio electrónico (`/webmailSOURCES`), para a biblioteca XMPP (`/xmpp`), para a aplicação *"myComs"* (`/myComs`) e para a biblioteca de localização de serviços (`/resources`). Em cada uma

destas directorias existe um ficheiro `build.xml`, todos eles com os mesmos objectivos mas respeitantes a cada um dos módulos. No `/myComs` há uma preocupação de integração do próprio `/myComs` com o do `/webmailSOURCES`; os outros, `/resources` e `/xmpp`, sendo bibliotecas, quando gerados copiam-se automaticamente para a directoria `/myComs/lib`. Embora não seja padrão seguido em todos os módulos, no `/myComs` o `build.xml` é uma integração de outros ficheiros XML onde cada um tem objectivos específicos. A tabela A.1 mostra essa agregação e os objectivos de cada um.

`build.xml`	`compile.xml`	Compila todos os ficheiros com código fonte
	`subscription.xml`	Gera o componente empresarial associado à subscrição
	`userAgent.xml`	Gera o componente empresarial associado ao serviço de mensagens
	`war.xml`	Gera a componente Web da aplicação myComs

Tabela A.1: Estrutura do ficheiro build.xml da directoria myComs

Claro que tudo o que está associado ao serviço de correio electrónico, não estando na mesma árvore de directorias do `/myComs`, aparece directamente no ficheiro `build.xml`. Com o exemplo que se segue na listagem A.2 verifica-se que para compilar, para além de recorrer ao *target compile* do ficheiro `compile.xml` recorre igualmente ao *compile* do ficheiro `build.xml` do *webmail*.

Listagem A.2: Definição do target de compilação do módulo myComs

```
/PUCProject/myComs/build.xml
...
  <!-- ================================================================== -->
  <!-- Compiles the source code                                           -->
  <!-- ================================================================== -->
  <target name="compile" depends="init">
        <ant antfile="compile.xml" target="compile" />
        <ant antfile="${webmail}${/}build.xml" target="compile" />
  </target>
...
```

Para terminar, e para justificar a inserção desta secção, como já foi referido, enumera-se os diferentes *targets* e os seus objectivos. A mesma tabela aparece sempre que seja executado o *Ant* sem especificar o *target* pretendido (há um parâmetro na definição do campo *project* do ficheiro `build.xml` que permite indicar qual o *target* a executar por omissão).

Target	Acção
init	inicia nomes e classpath
compile	compila sources
war	cria ficheiro war (componente web)
subscription	cria ficheiro subscription jar (componente ejb)
userAgent	cria ficheiro userAgent jar (componente ejb)
mail	cria ficheiro mail jar (componente ejb)
ear	cria ficheiro ear (aplicacao empresarial)
deploy	copia ear para a directoria de deploy do JBoss
farm	copia ear para a directoria Jboss\server\all\farm
all	copia ear para a directoria Jboss\server\all\deploy
clean-classes	apaga classes
clean-war	apaga war e directoria (componente web)
clean-subscription	apaga subscription jar e directoria (componente ejb)
clean-userAgent	apaga userAgent jar e directoria (componente ejb)
clean-mail	apaga mail jar e directoria (componente ejb
clean-ear	apaga ear (aplicacao empresarial)
clean-deploy	undeploy de default
clean-farm	undeploy de farm
clean-all-deploy	undeploy de all deploy
clean-all	apaga tudo

Tabela A.2: Acções possíveis do build.xml da directoria myComs

A.4 Aspectos específicos de configuração

A.4.1 No contexto Web

Não existe qualquer aspecto específico de configuração a salientar.

A.4.2 No contexto do servidor JBoss

Não existe qualquer aspecto específico de configuração a salientar.

A.4.3 Componente de subscrição

Existe um ficheiro de propriedades relativas essencialmente ao servidor de LDAP. Encontra-se no pacote `eis.util` e o seu nome é `jndi.properties`. Esse ficheiro deve estar coerente com o esquema de directorias utilizado no servidor. Numa fase inicial chamava-se `directory.properties` e a razão da alteração está descrito na secção 4.4.1. Pode-se verificar que na directoria onde este ficheiro existe encontra-se outro que poderá ter o nome `jndi.properties PTi` ou `jndi.properties Inesc` função do que está a ser utilizado como `jndi.properties`. É evidente que cada um reflecte características específicas inerentes a cada um dos locais de distribuição.

A.4.4 Componente de correio electrónico

Existem dois ficheiros que requerem especial atenção: `jwma.properties` e `site_template.xml` (ambos se encontram na directoria `etc`).

O ficheiro `site_template.xml` representa o padrão de informação das preferências para um novo utilizador. No seu conteúdo existem campos que definem a árvore de directorias do servidor de correio electrónico a utilizar.

Listagem A.3: Definição do esquema de directorias do servidor de correio electrónico

```
1 etc/site_template.xml
2 <?xml version="1.0"?>
3 <preferences>
4   ...
5   <rootfolder>INBOX</rootfolder>
6   <draftfolder>INBOX.Draft</draftfolder>
7   <trashfolder>INBOX.Trash</trashfolder>
8   ...
9 </preferences>
```

São esses campos que precisam ficar coerentes com o esquema de directorias do servidor de correio electrónico. O exemplo referido na listagem A.3 é para o servidor de correio electrónico *qmail*.

O ficheiro `jwma.properties` é o principal ficheiro de configuração do JWMA. Nele pode-se encontrar informação de propriedades de administração, propriedades de novo utilizador, propriedades do servidor de correio electrónico, configuração de *logs*, configuração do processamento, aliases dos controladores e aliases das apresentações. Regra geral, as configurações necessárias centram-se nas propriedades do servidor de correio electrónico, nomeadamente no endereço IP do servidor de correio electrónico.

Listagem A.4: Excerto do ficheiro de configuração do JWMA

```
1 etc/jwma.properties
2 ...
3 \\Postoffice settings
```

```
4  jwma.postoffice.protocol=imap
5  jwma.postoffice.host=172.29.250.35
6  ...
7  \\Mail transport settings
8  jwma.mailtransport.protocol=smtp
9  jwma.mailtransport.host=172.29.250.35
10 ...
```

Para mais informações sobre este ficheiro (excerto na listagem A.4), sobre outras propriedades ou o que elas significam, consultar o endereço:

`http://jwma.sourceforge.net/configuration/jwmaproperties_doc.html`

A.4.5 Descritores das aplicações empresariais

Uma outra chamada de atenção tem a ver com os descritores das aplicações empresariais. Por questões de validação sintáctica, todos eles fazem referências a DTD (*Document Type Definition*). Acontece que essas referências são referências para DTD que se encontram no disco, na directoria `file:///c:/JPatri/PUCProject/dtd/`. Foi uma forma de se poder trabalhar sem ter ligação à rede já que na versão original as referências são para o servidor da Sun (`http://java.sun.com/dtd/`). No projecto existem as directorias `descriptors` que é a que a aplicação usa, `descriptors_remotos`, `descriptors_locais`, `descriptorsNoCluster`, ou `descriptorsCluster`. Quando se pretender mudar as referências dos DTD para o servidor da Sun (fica independente da localização dos DTD), ou o contrário, é suficiente copiar os descritores adequados para a directoria `descriptors`.

Apêndice B

Glossário

Anónimo Nome genérico do utilizador que não está registado no sistema PUC.

Contexto Contexto é tudo o que possa influenciar o fluxo normal de um diálogo, de um pensamento, ... Neste trabalho o diálogo verifica-se entre o utilizador e o sistema. Sistematizando, contexto ou uma aplicação ser baseada em contexto corresponde a ter a capacidade de responder aos cinco Ws: *Where*, *When*, *Who*, *What*, *hoW*.

Correio electrónico Designação para o vulgarmente chamado mail. Corresponde a um cliente que gere e visualiza as mensagens residentes no servidor de correio electrónico.

DTMF *Dual Tone Multi Frequency*. É usado por telefones de teclado; atribui um par de frequências específicas, ou tom, para cada tecla.

EIS *Enterprise Information System*. Corresponde à infraestrutura de informação crítica para o processo de negócio de uma empresa. Exemplos de EISs incluem bases de dados relacionais, sistemas de planeamentos de recursos (ERP - *Enterprise Resource Planning*), sistemas de processamento de transacções.

EJB *Enterprise Java Bean*. Especificação da Sun para componentes empresariais.

Endereço electrónico Designação genérica da identificação recursos fundamentais deste projecto, nomeadamente: servidor *Web*/WAP; serviço de correio electrónico (*email*); serviço de correio de voz (*voiceMail*); e serviço de mensagens instantâneas (IM).

Gestor Utilizador do sistema, com privilégios máximos, responsável pela administração e configuração do sistema.

GPS *Global Positioning System*. Sistema de navegação baseado em satélites compostos por uma rede de 24 satélites colocados em órbita terrestre pelo Departamento de Defesa dos Estados Unidos.

IETF *Internet Engineering Task Force*. Comunidade internacional aberta para definição de standards para a Internet.

IM *Instant Messaging*. Serviço de troca de mensagens instantâneas e notificações de presença.

IMAP *Internet Message Access Protocol*. Protocolo standard para acesso a mensagens. Protocolo usualmente utilizado no acesso a um servidor de correio electrónico. Difere do protocolo POP por manter todas as mensagens no servidor. Descarrega apenas os *headers* das mensagens para visualização rápida e descarrega uma cópia da mensagem para visualização da mesma.

J2EE *Java To Enterprise Edition*. Plataforma standard especificada pela Sun para aplicações do lado do servidor.

J2ME *Java To Micro Edition*. Plataforma standard especificada pela Sun para sistemas embebidos.

JCA *Java Connector Architecture*. Especificação da Sun para ligação de sistemas heterogéneos à plataforma J2EE.

JMX *Java Management eXtension*. Especificação da Sun para gestão standard de componentes.

LDAP *Lightweight Directory Access Protocol*. Protocolo standard para esquema de directorias.

Membro Nome genérico do utilizador registado no sistema PUC.

MMS *Multimedia Message Service*. Serviços disponibilizado pelas operadoras de telecomunicações para troca de mensagens multimédia (texto, imagens e vídeo).

OTA *Over The Air*. É a capacidade das transportadoras adicionarem novos tipos de serviços aos dispositivos móveis usando a rede sem fios em vez de obrigarem os clientes a deslocarem-se a algum sítio específico para reprogramarem os dispositivos.

PDA *Personal Digital Assistent*. Tipo de dispositivo móvel, normalmente muito associado a uma agenda electrónica, mas também com funcionalidades de comunicação (com acesso à Internet). Actualmente existem já dispositivos com capacidade de estabelecimento de chamadas telefónicas.

Pattern Identificação de um padrão de programação com consecutiva catalogação.

Peer-To-Peer Ligação ponto a ponto. Difere do conceito cliente servidor onde o servidor mantém várias ligações com os clientes. Exemplo de uma aplicação típica é o *KaZaA*.

PIM *Personal Information Management*. Sigla para representar serviços cujas funcionalidades estejam relacionados com a gestão de informação pessoal.

POP *Post Office Protocol*. Protocolo standard para acesso a mensagens. Protocolo usualmente utilizado no acesso a um servidor de correio electrónico. Difere do protocolo IMAP por descarregar as mensagens lidas do servidor de correio electrónico para o disco local. As mensagens descarregadas são apagadas.

PUC *Personal Unified Communication*. Nome de sigla do projecto com a designação de Comunicações Unificadas Pessoais para as redes de 3G/NGN).

Serviço Um serviço disponibiliza um conjunto de funcionalidades para um utilizador através de uma interface gráfica. São exemplos de serviços: serviço de apresentação de conteúdos HTML, serviço de apresentação de notícias, serviço de mensagens instantâneas.

SGML *Standard Generalized Markup Language*. É um standard internacional (ISO 8879:1985) para definir descrições de estruturas de diferentes tipos de documentos electrónicos.

SMS *Short Message Service*. Serviço disponibilizado pelas operadoras de telecomunicações para troca de mensagens curtas (no máximo 160 caracteres).

SMTP *Simple Message Transport Protocol*. Protocolo standard para envio de mensagens do cliente para o servidor, ou entre servidores, de correio electrónico.

SOAP *Simple Object Access Protocol*. É um protocolo simples baseado em XML para permitir a troca de informação entre aplicações sobre HTTP.

Terminal Designação genérica dada ao aparelho físico de acesso ao sistema PUC. Exemplos: telefone fixo; telefone móvel; PC; PDA.

Ubiquidade Dom de estar ao mesmo tempo em vários lugares. No contexto desta tese corresponde à capacidade de aceder aos serviços e à informação de qualquer sítio através de qualquer terminal.

XML *eXtensible Markup Language*. XML tem o objectivo de tornar fácil e directo o uso de SGML na *Web*: fácil de definir tipos de documentos, fácil de gerir documentos definidos em SGML, e fácil de transmiti-los e partilhá-los na *Web*.

XMPP *eXtensible Messaging and Presence Protocol*. Protocolo standard em XML para troca de mensagens e notificações de presença na Internet.

WAP *Wireless Access Protocol*. O protocolo WAR é o standard de eleição para serviços de informação em terminais sem fios como por exemplo telefones móveis digitais.

Referências

[1] http://ieeexplore.ieee.org/Xplore/DynWel.jsp.

[2] 3rd Generation Partnership Project (3GPP), *Mobile station application execution environment (MExE)*, Tech. report, 3GPP, December 2000.

[3] http://www.aim.com/.

[4] Dan Bernstein, *Qmail - smtp server*, http://www.qmail.org/.

[5] Bill Burke and Adrian Brock, *Aspect-oriented programming and jboss*, Tech. report, JBoss, 2003, http://www.oreilly.com/pub/a/onjava/2003/05/28/aop_jboss.html.

[6] http://www.osafoundation.org/Chandler_Compelling_Vision.htm.

[7] http://www.osafoundation.org/Chandler-Product_Roadmap.htm#theoryofadoption.

[8] João Clemente, *Personal unified communication - análise e avaliação de requisitos não funcionais*, Tech. report, Instituto Superior Técnico, Inesc-ID, 2003.

[9] The Middleware Company, *J2ee application server survey 2004*, http://www.middlewareresearch.com//endeavors/042708J2EESURVEY/endeavor.jsp.

[10] *Portugal's largest retailer, modelo continente, selects seebeyond to streamline supply chain and improve customer satisfaction*, http://ir.seebeyond.com/phoenix.zhtml?c=63418&p=irol-newsArticle&t=Regular&id=468128&.

[11] Alberto Rodrigues da Silva, João Patriarca, João Clemente, Paulo Chainho, and Paulo Ferreira, *Puc - sistema de comunicações pessoais para as redes de próxima geração*, III Congresso Iberoamericano de Telemática (Torre de las Telecomunicaciones de ANTEL, Montevideo, Uruguay), Outubro 2003, http://cita2003.fing.edu.uy/.

[12] Sun MicroSystems Developers, *Java authentication and authorization service*, http://java.sun.com/products/jaas/.

[13] Anind K. Dey and Gregory D. Abowd, *Towards a better understanding of context and context-awareness*, Tech. report, Graphics, Visualization and Usability Center and College of Computing, Georgia Institute of Technology, Atlanta, GA, USA, 1999, ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf.

[14] Programa do XV Governo Constitucional, *Biblioteca do conhecimento online*, http://www.b-on.pt/.

[15] Nuno Domingos and Nuno César, *Projecto puc - "personal unified communication" - sistemas embebidos para o contexto puc*, Tech. Report 149, Instituto Superior Técnico, Inesc-ID, 2003.

[16] Porto Editora, *Dicionário online de língua portuguesa, inglês/português/inglês e francês/português/francês*, http://www.portoeditora.pt/dol/.

[17] http://www.elefante.com.br/.

[18] Matjaz Juric et al, *J2ee design patterns applied*, 1 edition ed., Wrox Press Inc, June 2002, ASIN: 1861005288.

[19] Java Community Process (JCP) expert group (JSR-118), *Mobile information device profile 2.0*, Tech. report, Sun Microsystems, Novembro 2002.

[20] Java Community Process (JCP) expert group (JSR-129), *Personal basis profile specification 1.0*, Tech. report, Sun Microsystems, Junho 2002.

[21] Java Community Process (JCP) expert group (JSR-139), *Connected limited device configuration 1.1*, Tech. report, Sun Microsystems, Março 2003.

[22] Java Community Process (JCP) expert group (JSR-19), *Enterprise JavaBeans^{TM} 2.0*, Tech. report, Sun Microsystems, Março 2002, http://www.jcp.org/en/jsr/detail?id=19.

[23] Java Community Process (JCP) expert group (JSR-216), *Personal profile version 1.1 for the J2ME^{TM} platform*, http://www.jcp.org/en/jsr/detail?id=216.

[24] Java Community Process (JCP) expert group (JSR-217), *Personal basis profile version 1.1 for the J2ME^{TM} platform*, http://www.jcp.org/en/jsr/detail?id=217.

[25] Java Community Process (JCP) expert group (JSR-218), *J2ME^{TM} connected device configuration (cdc) 1.1*, http://www.jcp.org/en/jsr/detail?id=218.

[26] Java Community Process (JCP) expert group (JSR-219), *J2ME^{TM} foundation profile 1.1*, http://www.jcp.org/en/jsr/detail?id=219.

[27] Java Community Process (JCP) expert group (JSR-30), *J2ME^{TM} connected, limited device configuration 1.0*, Tech. report, Sun Microsystems, Maio 2000.

[28] Java Community Process (JCP) expert group (JSR-36), *J2ME^{TM} connected device configuration 1.0*, Tech. report, Sun Microsystems, Março 2001.

[29] Java Community Process (JCP) expert group (JSR-37), *Mobile information device profile for the J2ME^{TM} platform*, Tech. report, Sun Microsystems, Setembro 2000.

[30] Java Community Process (JCP) expert group (JSR-46), *J2ME^{TM} foundation profile*, Tech. report, Sun Microsystems, Março 2001.

[31] Java Community Process (JCP) expert group (JSR-58), *JAVA^{TM} 2 platform, enterprise edition 1.3 specification*, Tech. report, Sun Microsystems, Setembro 2001, http://www.jcp.org/en/jsr/detail?id=58.

[32] Java Community Process (JCP) expert group (JSR-62), *Personal profile specification*, Tech. report, Sun Microsystems, Setembro 2002.

[33] Java Community Process (JCP) expert group (JSR-68), *J2ME^{TM} platform specification*, Tech. report, Sun Microsystems, Julho 2002, http://www.jcp.org/en/jsr/detail?id=68.

[34] Alan O. Freier, Philip Karlton, and Paul C. Kocher, *The ssl protocol version 3.0*, Tech. report, Netscape Communications Corporation, Março 1996, http://wp.netscape.com/eng/ssl3/.

[35] Marco Guimarães and Rui Maia, *Personal unified communications*, Tech. report, Instituto Superior Técnico, Inesc-ID, 2003.

[36] http://hambo.sourceforge.net/.

[37] http://www.icq.com/.

[38] IEEE (ed.), *Context-sensitive middleware for real-time software in ubiquitous computing environments*, Computer Science and Engineering Department, Arizona State University, Tempe, AZ 85287, USA, Object-Oriented Real-Time Distributed Computing, May 2001, Fourth IEEE International Symposium.

[39] IEEE (ed.), *Context-aware service protocol. an extensible and configurable framework for user context awareness services in pervasive computing systems*, vol. 3, Wireless Communications and Networking, March 2003, WCNC 2003.

[40] IEEE (ed.), *How to program pervasive systems*, Centre for Accident Research and Road Safety - Queensland (CARRS-Q), Queensland University of Technology, Australia, Database and Expert Systems Applications, September 2003, 14th International Workshop.

[41] IEEE (ed.), *Middleware design and human factor*, Department of Computer Science, Waseda University, Tokyo, Japan, Object-Oriented Real-Time Dependable Systems, October 2003, Ninth IEEE International Workshop.

[42] IEEE (ed.), *A network service framework for mobile pervasive computing*, vol. 2, Communication Technology Proceedings, April 2003.

[43] IEEE (ed.), *ubidss: a framework of multi-agent based proactive decision support system with context-awareness*, Software Technologies for Future Embedded and Ubiquitous Systems, May 2004, Second IEEE Workshop.

[44] IEEE/IFIP (ed.), *Pervasive computing and management*, vol. 1, Network Operations and Management Symposium, April 2004.

[45] *Most used operating systems*, http://www.findarticles.com/p/articles/mi_m0BNG/is_2003_Sept_25/ai_108127532.

[46] http://www.jabber.org/ietf/.

[47] *Jabberd - open source server implementation of the jabber protocols*, http://jabberd.jabberstudio.org/.

[48] *Protocolo xml standard para troca de mensagens na internet*, http://www.jabber.org/jeps/.

[49] JBoss Group, *Jboss documentation*, http://www.jboss.org.

[50] Sun MicroSystems (JSR-116), *Conjunto de apis enquadradas na iniciativa jain*, http://www.jcp.org/aboutJava/communityprocess/final/jsr116/.

[51] Sun MicroSystems (JSR-240), *Conjunto de apis para permitir o rápido desenvolvimento de serviços e produtos java relacionados com comunicações para a próxima geração*, http://java.sun.com/products/jain/.

[52] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin, *Aspect-oriented programming*, Tech. report, Xerox Palo Alto Research Center, Junho 1997.

[53] Glenn E. Krasner and Stephen T. Pope, *A cookbook for using the model-view-controller user interface paradigm in smalltalk-80*, Object-Oriented Programming **1** (1988), no. 3, 27–49, ISSN:0896-8438.

[54] Rafal Lukawiecki, *A subjective view of the next 10 years in it: Emergence of knowledge workers*, http://research.microsoft.com/collaboration/university/europe/Events/AcademicDays/Dubai/0404/379,11,Implications of Information as a Service.

[55] David Martins and Eurico Frade, *Personal unified communication - integração*, http://berlin.inesc.pt/alb/uploads/1/189/PUC3-Relatorio-Final.pdf, 2004.

[56] Mari Matsunaga, Enoki, and Takeshi Natsuno, *Plataforma para comunicações de telefones móveis*, http://www.nttdocomo.com/.

[57] http://www.microsoft.com/outlook/.

[58] http://messenger.msn.com/.

[59] Sun MicroSystems, *Api para desenvolvimento de aplicações de mail e de mensagens*, http://java.sun.com/products/javamail/index.jsp.

[60] ———, *Arquitectura de componentes para a plataforma java standard edition*, http://java.sun.com/products/javabeans/index.jsp.

[61] ———, *Extensão da funcionalidade de um servidor web*, http://java.sun.com/products/servlet/index.jsp.

[62] ———, *Java connector architecture – solução para o problema de conectividade entre muitos servidores de aplicações e sistemas de informação empresarial actuais*, http://java.sun.com/j2ee/connector/index.jsp.

[63] ———, *Java management extensions*, http://java.sun.com/products/JavaManagement/.

[64] ———, *Java server pages – criação dinâmica de conteúdo web*, http://java.sun.com/products/jsp/index.jsp.

[65] ———, *Over the air*, http://java.sun.com/products/midp/OTAProvisioning-1.0.pdf.

[66] *Jetty - java http&servlet server*, http://jetty.mortbay.com/.

[67] Netscape Navigator, *An exploration of dynamic documents*, http://wp.netscape.com/assist/net_sites/pushpull.html/.

[68] *Netbeans ide for java*, http://www.netbeans.org/.

[69] *Sincronização de dados para dispositivos em rede*, 2001, http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html#V101.

[70] *Open source initiative*, http://www.opensource.org/.

[71] *Implementação open source do lightweight directory access protocol*, http://www.openldap.org/.

[72] *Java jdbc ldap bridge driver*, http://www.openldap.org/jdbcldap/.

[73] OSAF, *Fusão da biblioteca de classes c++ wxwidgets com a linguagem de programação python*, http://www.wxpython.org/.

[74] http://www.outsystems.com/.

[75] http://www.outsystems.com/Site/customers.aspx.

[76] Jason Pascoe, *Adding generic contextual capabilities to wearable computers*, Tech. report, Computer Laboratory, University of Kent at Canterbury, Canterbury, Kent CT2 7NF, United Kingdom, 1998.

[77] *Portal de software open source*, http://sourceforge.net/.

[78] Apache Jakarta Project, *Tomcat*, http://jakarta.apache.org/tomcat/index.html.

[79] Bill N. Schilit, Norman Adams, and Roy Want, *Context-aware computing applications*, Tech. report, Dezembro 1994.

[80] *Most used operating systems*, http://www.specialtyauto.com/stats/0404/Browsers.htm.

[81] Python Team, *Linguagem de programação interpretada, interactiva, e object-oriented*, http://www.python.org/.

[82] *Apache ant - java-based build tool*, http://ant.apache.org/.

[83] *Tmn*, http://www.tmn.pt/.

[84] Sam Varshavchik, *Courier imap - imap server*, http://www.courier-mta.org/.

[85] *Voice extensible markup language*, Março 2000, http://www.voicexml.org/specs/VoiceXML-100.pdf.

[86] http://www.weblicon.net/.

[87] http://www.weblicon.net/html/customers.html.

[88] Shawn Wilton, *Biblioteca que encapsula a criação de tramas xml do protocolo jabber*, http://www.jabberstudio.org/projects/jabberbeans/project/view.php.

[89] Dieter Wimberger and Leonard Sitongia, *Implementação de um cliente web para acesso a uma conta de mail*, http://jwma.sourceforge.net/.