Departamento de Engenharia Informática

Relatório Final

TRABALHO FINAL DE CURSO

do Curso de

LICENCIATURA EM ENGENHARIA INFORMÁTICA E DE COMPUTADORES (LEIC)

Ano Lectivo 2003 / 2004

N.º da Proposta: 4

Título: PUC- Sistema de Comunicações Pessoais e Unificada

Professor Orientador:

Alberto Silva _____

Co-Orientador:

Paulo Chainho (PT Inovação) _____

Alunos:

48244, David Martins _____

48248, Eurico Frade _____

Agradecimentos

Primeiro gostaríamos de dar uma palavra especial de agradecimento ao nosso orientador, Professor Alberto Silva, pelo apoio, disponibilidade e compreensão demonstrados ao longo do período de execução deste trabalho.

Gostaríamos ainda de agradecer a ajuda das pessoas já envolvidas no desenvolvimento deste projecto, sendo elas: Rui Maia, Marco Guimarães, Nuno César, Nuno Domingos e João Patriarca.

Gostaríamos também de agradecer a todos os nossos amigos que de certa forma nos deram o seu apoio e alguma sugestão para o desenvolvimento deste trabalho. De entre estes um agradecimento especial para António Silva e Tiago Venda, que sempre se disponibilizaram a emprestar-nos os seus telemóveis para podermos testar a nossa aplicação móvel.

A todos o nosso muito obrigado.

Índice

Índice de Figuras

1 Introdução

Hoje em dia, num mundo em que as pessoas vivem cada dia, sempre a correr de um lado para o outro, e que precisam estar sempre comunicáveis, não apenas pelo telefone, mas também por meio de email, ou de ter uma agenda à qual possam aceder estando em sua casa ou no escritório, estando na rua ou no automóvel no meio de um grande engarrafamento, a mobilidade e a portabilidade tem vindo a tornar-se cada vez mais num factor de elevada importância no sucesso do lançamento de produtos das tecnologias de informação.

Este trabalho de final de curso, enquadra-se no âmbito deste tipo de produtos, uma vez que tem como objectivo permitir o acesso, a partir de uma larga variedade de dispositivos móveis, como telemóveis ou PDAs, e também através de um PC ligado à Internet, a um leque de serviços pessoais.

Mais especificamente, este trabalho centrou-se na integração de alguns módulos já desenvolvidos de modo a tornar o sistema robusto, e para que os módulos até aqui desenvolvidos comunicassem entre si e trouxessem alguma funcionalidade extra à utilização independente de cada módulo. Esta aplicação pretende demonstrar as capacidades do conceito "PUC"(*Personal Unified Communications*) numa infra-estrutura de redes de 3G/NGN e enquadra-se num projecto em parceria com INESC-ID e PT-Inovação [25]. Genericamente esta aplicação divide-se em duas partes: "Assistente pessoal de comunicações" – myCom [28] e "Assistente pessoal de agenda" – myAgenda [26]. No "Assistente pessoal de comunicações" o utilizador do sistema acede e utiliza as suas diversas contas de comunicação pessoais como por exemplo *email* e *instant messaging*. Com o "Assistente pessoal de agenda" o cliente pode marcar/desmarcar compromissos ou reuniões, e também aceder a funcionalidades habituais de um gestor de contactos centralizado, entre outras funcionalidades. Outra parte deste sistema consiste na aplicação para dispositivos móveis [27], que possui algumas das funcionalidades disponibilizadas pelos módulos myAgenda e myCom.

Este trabalho começou com um estudo profundo da arquitectura do módulo myAgenda [26], seguido de um estudo da arquitectura do módulo myCom [28]. Estando este estudo feito passou-se então à escolha de qual a arquitectura do sistema final, seguidamente procedeu-se à integração dos módulos. O próximo passo foi estudar a aplicação para dispositivos móveis [27] e suas tecnologias subjacentes, por fim procedeu-se à reconstrução da mesma aplicação.

O relatório encontra-se estruturado da seguinte forma:

No capítulo 2 é feita uma breve descrição do "estado da arte". Descreve-se as várias tecnologias de suporte analisadas, e algumas das quais usadas, na implementação deste projecto. Foi dada especial importância às plataformas J2EE [1] e J2ME [2], uma vez que foram as plataformas de suporte deste trabalho. A primeira foi utilizada na aplicação cliente

acedida pela web e no desenvolvimento do servidor central da aplicação e a segunda utilizada no desenvolvimento da aplicação para os dispositivos móveis. Na segunda parte deste capítulo, é feito um resumo da pesquisa feita relativamente às aplicações existentes no mercado que de alguma maneira se podem comparar com a aplicação desenvolvida no âmbito deste TFC. Pesquisa esta que permitiu obter uma ideia de quais a funcionalidades habituais deste tipo de produtos.

No capítulo 3 dá-se uma visão do que é o sistema PUC. Apresenta-se a sua arquitectura geral com os componentes principais, assim como quais as funcionalidades desenvolvidas. Descreve-se também neste capítulo o modelo de casos de utilização correspondentes às funcionalidades presentes neste projecto.

O capítulo 4 detalha aspectos de análise e implementação do projecto. Este capítulo encontra-se dividido em duas partes principais, cada uma representando um aspecto diferente, como sejam: (1) o servidor central do sistema; e (2) a aplicação para dispositivos móveis. Em cada uma destas secções são descritas as principais decisões tomadas acerca das tecnologias, a arquitectura e outros aspectos importantes ocorridos ao longo do processo de desenvolvimento.

No capítulo 5 encontram-se descritos alguns tópicos descrevendo algumas alterações e novas funcionalidades que poderão aumentar a gama de funcionalidades oferecidas pelo sistema.

Por fim, no capítulo 6, apresentam-se as conclusões retiradas durante todo o processo de desenvolvimento deste projecto.

2 Estado da Arte

Neste capítulo são apresentadas algumas tecnologias de suporte utilizadas para o desenvolvimento deste projecto. São também descritas outras aplicações com características semelhantes, de forma a mostrar o segmento do mercado no qual este produto se insere e em que aspectos se diferencia e se equipara às alternativas existentes.

2.1 Tecnologias de Suporte

Relativamente às tecnologias de suporte, fala-se do J2EE [1] (*Java 2 Enterprise Edition*), que foi a plataforma utilizada no desenvolvimento do servidor PUC, do servidor aplicacional JBoss [3] e das ferramentas XDoclet [4] e Ant [5] que facilitaram bastante o desenvolvimento e implementação desta parte do projecto.

Em seguida é feita uma pequena descrição do J2ME [2] (*Java 2 Micro Edition*), que foi a plataforma utilizada para o desenvolvimento da aplicação cliente do PUC a ser instalada em telemóveis.

No final desta parte descreve-se ainda outras tecnologias, protocolos e ferramentas utilizadas, como por exemplo o Jabber [6], como protocolo de mensagens instantâneas e serviço de notificações, e o PostgreSQL [7], como suporte persistente dos dados.

2.1.1 J2EE

A plataforma J2EE [1] (*Java 2 Enterprise Edition*) define um padrão para o desenvolvimento de aplicações empresariais multicamada. O J2EE [1] aproveita a vantagem de muitas das características do J2SE [8] (*Java 2 Standard Edition*), como a portabilidade "*Write Once, Run Anywhere*", a API JDBC para acesso a base de dados, tecnologia CORBA para interacção com recursos empresariais existentes, e um modelo de segurança que protege os dados mesmo em aplicações web. Construído nesta base o J2EE [1] junta um suporte completo a *Enterprise JavaBeans*, *Java Servlets API*, *JavaServer Pages™* e tecnologia XML.

Em J2EE [1] o modelo aplicacional encapsula as camadas de funcionalidade em tipos específicos de componentes. A interacção de um cliente pode ser apresentada através de páginas HTML, através de *applets*, *java servlets*, *javaserver pages* ou aplicações Java. As componentes comunicam de forma transparente através de vários standard: HTML, XML, HTTP, SSL, RMI, IIOP e outros.

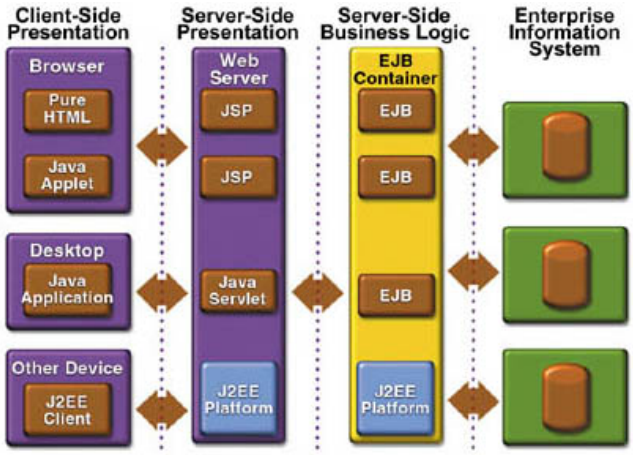

Figura 2.1 – Modelo aplicacional do J2EE

O J2EE [1] define uma plataforma standard em cima de uma série de sistemas empresariais, tais como sistemas de gestão de base de dados, monitores transaccionais, serviços de nomes e de directório, etc., unificando todos estes componentes num modelo aplicacional único.

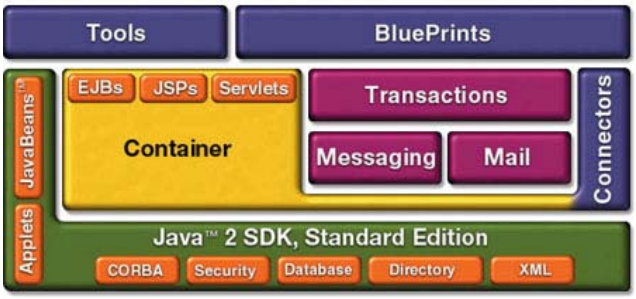

Figura 2.2 – Modelo aplicacional do J2EE

Deste modo, várias complexidades relacionadas com aplicações empresariais (gestão de transacções, gestão de ciclo de vida, gestão de recursos, etc.) estão embebidas na plataforma, disponíveis para todos os componentes que correm sobre a mesma.

Isto permite aos programadores libertarem-se destas preocupações, podendo-se concentrar em coisas mais específicas das suas aplicações tais como a lógica de negócio ou as interfaces.

A especificação J2EE [1] recomenda ainda as seguintes convenções em como agrupar as várias partes constituintes de uma aplicação empresarial, em colecções de arquivos:

- Ficheiros do tipo JAR contêm EJB [9] (*Enterprise JavaBeans*);
- Ficheiros do tipo WAR contêm *servlets*, páginas JSP, classes e bibliotecas utilitárias Java, documentos estáticos tais como páginas HTML, ficheiros de som e imagem, e *applets*, *beans* e classes do lado do cliente;
- Ficheiros do tipo EAR contêm ficheiros JAR, ficheiros WAR, e módulos aplicacionais do lado do cliente.

Uma aplicação empresarial conforme a especificação J2EE [1] vem então sobre a forma de um ficheiro do tipo EAR.

Concluindo, a plataforma J2EE [1] define a norma para o desenvolvimento de aplicações empresariais *multi-tier*. Simplifica as aplicações baseando-as em componentes modulares standard, fornecendo uma série completa de serviços a estes componentes, e fornecendo automaticamente alguns pormenores do comportamento da aplicação, sem requerer uma programação complexa. O standard J2EE [1] tem como filosofia, a divisão em camadas das aplicações, fornecendo mecanismos e padrões de desenvolvimento, tendo como objectivo a fácil adaptação e portabilidade de todas as aplicações.

2.1.1.1 JBoss

O JBoss [3] é um servidor aplicacional de EJB [9] (*Enterprise JavaBeans*) implementado em Java puro. Inclui o JBossServer, um contentor de EJBs (*EJB container*) e uma infra-estrutura JMX (*Java Management eXtensions*), o JBossMQ para mensagens JMS, JBossMail para mail, JBossTX para transacções JTA/JTS, JBossSX para JAAS baseado em segurança, JBossCX para conectividade JCA e JBossCMP para persistência CMP.

O JBoss [3] é integrável com o contentor de Servlet/JSP Tomcat ou com o servidor/contentor de servlet Jetty Web, e permite misturar e comparar estas componentes por JMX através da substituição de qualquer componente com uma implementação JMX-compliant para as mesmas APIs. O objectivo é oferecer uma implementação do modelo J2EE [1].

O JBoss [3] foi o servidor aplicacional escolhido para alojar o servidor PUC. A escolha deste servidor aplicacional deve-se ao facto de ser *open-source* e de ter vindo a ser utilizado no desenvolvimento do sistema PUC. De seguida é apresentada uma figura que ilustra, em alto nível, a integração de um sistema com Jboss [3] e vários componentes inerentes.

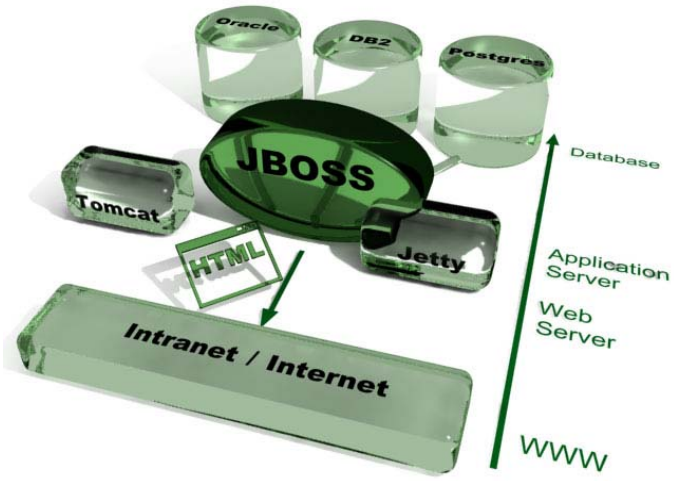

Figura 2.3 – Arquitectura de um sistema com JBoss

2.1.1.2 XDoclet

O XDoclet [4] é uma ferramenta de geração de código que permite programação orientada por atributos em Java. O XDoclet [4] faz um *parser* pelos ficheiros fonte e gera muitos dos artefactos como descritores XML, assim como o código fonte a partir deles. Estes ficheiros são gerados de *templates* que usam a informação providenciada no código e nas etiquetas do JavaDoc. Permite ainda ao programador fazer integração contínua no desenvolvimento orientado a componentes.

Embora o XDoclet [3] tenha origem como uma ferramenta para a criação de EJBs [9], evoluiu para num motor de geração de código. Esta ferramenta consiste num núcleo e num número de módulos que se encontram sempre em crescimento. É bastante fácil escrever novos módulos se houver uma necessidade de utilização de uma nova componente. O XDoclet [4] já vem com um conjunto de módulos para a geração de diferentes tipos de ficheiros, mas os utilizadores podem escrever os seus próprios módulos (ou modificar os já existentes) se quiserem estender as funcionalidades já disponibilizadas.

No âmbito do J2EE [1], o Xdoclet [4] é uma ferramenta poderosa já que as suas propriedades de geração de código avançadas facilitam (em aplicações que se pretendam escaláveis e modulares) a escrita de aplicações J2EE [1]. Para este efeito, é possível gerar várias classes, interfaces e ficheiros que fazem parte de uma aplicação conforme a especificação J2EE [1], apenas escrevendo a implementação de um *bean*. É devido a estas vantagens que se utilizou esta ferramenta no desenvolvimento deste projecto.

2.1.1.3 Ant

O Ant [5] é uma ferramenta construída em Java, em teoria é uma aplicação tipo Make, mas diferente de todas as ferramentas deste tipo já existentes como são: *make*, *gnumake*, *nmake*, *jam*. Em vez de um modelo onde é estendido com comandos de *shell*, o Ant [5] é estendido usando classes Java. Em vez de escrever comandos de *shell*, os ficheiros de configuração são baseados em XML, onde são descritas as várias tarefas que serão executadas. Cada tarefa é corrida por um objecto que implementa uma interface de tarefa particular. Isto retira algum poder expressivo, que está inerente à possibilidade de construir um comando de *shell*, mas oferece a vantagem de ser independente da plataforma utilizada. No caso de ser necessário um comando de *shell* existe uma tarefa <exec> que permite que diferentes comandos sejam executados baseados no sistema operativo onde está a ser executado.

2.1.1.4 Struts

O Struts [10] implementa o modelo MVC (*Model-View-Controller*), numa tentativa de separar as várias partes envolvidas num projecto distribuído. Os pedidos são feitos no browser do cliente e tratados pela parte do *Controller* do Struts [10]. Um pedido normalmente está mapeado numa acção que irá efectuar o processamento que necessitar e interagir com a parte do *Model*, onde reside a funcionalidade dos serviços. O mapeamento entre os pedidos e as acções está descrito num ficheiro de configuração, que é obrigatoriamente denominado por *struts-config.xml*. Os mapeamentos são constituídos pelo tipo de pedido, qual a acção a executar, e respectivos redireccionamentos em casos de erro ou sucesso.

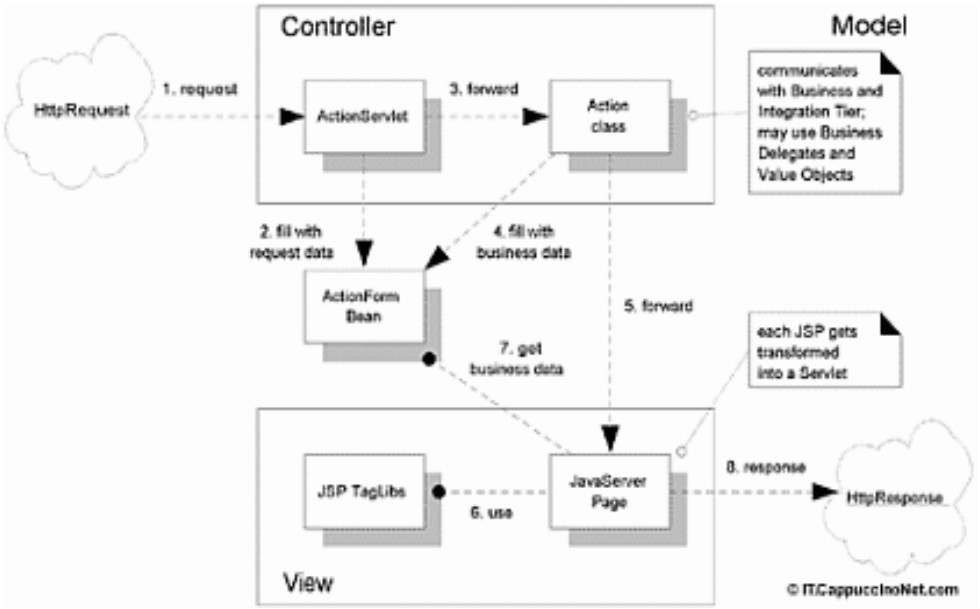

Figura 2.4 – Diagrama de Fluxos do Struts

Depois de acedidos os serviços do *Model*, são enviados os resultados pretendidos para a(s) *View*(s), neste caso para o browser, podendo no entanto existir várias vistas para uma mesma acção. Associadas à *View* estão algumas *Tag Libraries* do Struts [10] que visam a comunicação entre o *Controller* e a *View*, ou alguma funcionalidade adicional. Essas *Tag Libraries* necessitam de um ou vários ficheiros de propriedades que fazem a sua tradução em *Tags* HTML. Existe ainda um ficheiro adicional, o *web.xml*, que serve para fazer o mapeamento dos *servlets*.

2.1.1.5 Tiles

O Tiles [11] é uma *framework* que permite de forma fácil criar um conjunto de páginas (ou uma aplicação completa) com uma interface utilizador consistente, a mesma barra de navegação, o mesmo cabeçalho, o mesmo rodapé, etc. em cada página. Neste sentido esta *framework* permite a divisão de uma página de interface com o utilizador em várias partes,

sendo o conteúdo de cada parte manipulável de maneira transparente e singular. Através de um ficheiro XML que define os vários ecrãs e de um conjunto de *tags* embebidas nas páginas JSP, para inserir conteúdo estático e dinâmico, o Tiles [11] permite a criação de vistas a partir de componentes, possibilitando uma organização personalizada, adicionando flexibilidade, reusabilidade, consistência e facilidade de manutenção. A figura seguinte ilustra bem uma página interface utilizador neste conceito, integrando vários componentes reutilizáveis numa mesma página.

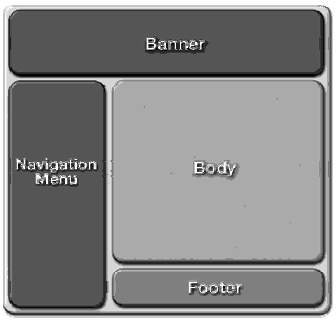

Figura 2.5 – Página exemplo do Tiles

Esta *framework* está particularmente bem integrada com o Struts [10] pois as equipas envolvidas nos dois projectos reconheceram a sua complementaridade e decidiram desenvolver esforços no sentido de cooperar. Desta forma, um programador pode especificar a definição de uma página do Tiles [11] como sendo a vista de *target* de uma acção do Struts [10]. Como o Struts [10] e o Tiles [11] estão em conformidade com a especificação da *tag library* das JSP, as *tags* de ambas as *frameworks* podem ser misturadas em páginas JSP.

O Struts [10] e o Tiles [11] providenciam uma base sólida para aplicações complexas e de dimensão considerável e aderem ao paradigma MVC e permitem aos programadores criar aplicações que podem crescer de uma forma sustentada e controlada de acordo com as necessidades de negócio e mudanças do mundo real.

2.1.1.6 EJB (*Enterprise JavaBeans*)

A tecnologia dos EJB [9] permite simplificar a aproximação ao desenvolvimento de aplicações com várias camadas, permitindo reduzir a complexidade nas aplicações, assim como possibilitando aos programadores focarem-se mais na lógica do negócio. O J2EE [1] é a evolução natural da tecnologia EJB [9] que oferece aos programadores a capacidade de modelar os objectos essenciais em dois tipos de componentes: *Session Beans* e *Entity Beans*. Para além destes dois tipos de componentes existe ainda um outro tipo que não é utilizado neste projecto os *Message Driven Beans*.

Os *Session Beans* representam o comportamento associado às sessões dos clientes. Estes são menos fiáveis do que os *Entity Beans*, pois se o servidor ou o cliente falham, o *Session Bean* e todos os dados a si inerentes são perdidos. Existem dois tipos de *Session Beans*: *Stateful* e *Stateless*. Um *Stateless Session Bean* não mantém o estado durante a sua interacção com o cliente. O termo *stateless* indica que a instância do *bean* não tem qualquer estado associado com um cliente específico. Dois pedidos diferentes do mesmo cliente podem ser tratados por diferentes instâncias de *Stateless Session Beans*. Por sua vez, um *Statefull Session Bean* mantém um estado com o respectivo cliente durante a ocorrência da sessão com o mesmo. Múltiplos pedidos do mesmo cliente numa sessão irão ser tratados pelo mesmo *bean*. O estado é mantido durante a sessão, sendo destruído quando o cliente remove o *bean* ou termina a interacção.

Os *Entity Beans* representam uma colecção de dados e encapsulam as operações nos dados que representam. Quando um *Entity Bean* é criado, os dados a si inerentes são escritos para a linha da tabela em base de dados apropriada, assim como se os dados existentes num *Entity Bean* forem alterados, os dados correspondentes na linha da tabela da base de dados são também alterados. Os dados inerentes a um *Entity Bean* são persistentes, pois ficam na base de dados mesmo se a aplicação ou o servidor terminarem ou se ocorrer uma falha da base de dados.

2.1.1.7 XML, SOAP e Web Services

O XML [12] (*Extensible Markup Language*) é uma linguagem para trocas de informação na web. O protocolo SOAP [13] (*Single Object Acess Protocol*), também conhecido como *Service Oriented Architecture Protocol*, é usado para trocar informação estruturada num ambiente descentralizado e distribuído e usa o XML [12] para formatar os dados transmitidos. Como o SOAP [13] invoca funções de baixo nível, a linguagem de descrição de serviços web, WSDL [14] (*Web Services Description Language*) foi definida de modo a facilitar a descrição dos serviços web. O WSDL [14] é um complemento ao SOAP [13] na medida em que facilita a interoperabilidade entre serviços na web. Os web services [15] permitem a comunicação de aplicações e sistemas heterogéneos remotos via objectos de comunicação. O XML [12], SOAP [13] e os web services [15] constituem as peças chave para o desenvolvimento de novas aplicações web e de novos serviços de comunicação multimédia.

2.1.2 J2ME

O J2ME [2] é a plataforma Java adaptada aos dispositivos embebidos tais como telemóveis, PDA's, computadores de bordo de automóveis, etc..

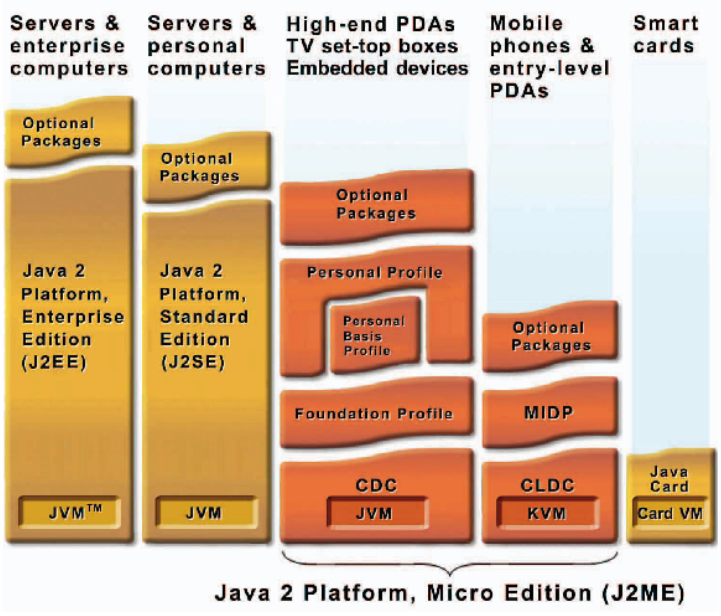

Figura 2.6 – Comparação entre as plataformas J2EE, J2SE e J2ME

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

2.1.2.1 Configurações

Actualmente o J2ME [2] possui duas configurações:

Connected Device Configuration **(CDC) –** Esta configuração foi desenvolvida para dispositivos com maior capacidade de memória, processadores mais rápidos e ligações com maior largura de banda, tais como computadores de bordo de automóveis ou PDA topo de gama.

Connected Limited Device Configuration **(CLDC) –** 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.

2.1.2.2 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 4 "perfis". Uma para a configuração CLDC e três para a CDC. São as seguintes:

***Mobile Information Device Profile* (MIDP) –** O MIDP foi desenvolvido para telemóveis e PDA 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 [2] deste projecto.

***Foundation Profile* (FP) –** 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 FP é o perfil de mais baixo nível para CDC fornecendo uma implementação do CDC com capacidade de ligação a rede que pode ser usado para implementações 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* (PP) –** O PP é o perfil CDC direccionado para dispositivos que necessitam de uma completa interface utilizador gráfica ou suporte para *applets* Internet, tais como PDA topo de gama e consolas de jogos. Inclui bibliotecas JAVA *Abstract Window Toolkit* (AWT) e proporciona *applets* de fácil utilização baseados na Web desenvolvidos para utilização num ambiente *desktop*. O PP substitui a tecnologia Personal Java (utilizada por exemplo no *Nokia Communicator*) e proporciona uma fácil migração para a plataforma J2ME [2].

***Personal Basis Profile* (PBP) –** O PBP, é um subconjunto do PP que proporciona um ambiente aplicacional para dispositivos com ligação a 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 computadores de bordo de automóveis e quiosques de informação.

2.1.3 PostgreSQL

O PostgreSQL [7] é um servidor de bases de dados relacionais *open-source*, tendo sido o servidor escolhido para alojar o modelo de dados do sistema PUC. Esta escolha deve-se ao facto de, numa primeira análise, ser *open-source*, e, numa análise mais detalhada, verificar-se que é bastante completo no que diz respeito a propriedades de um sistema de base de dados relacional (como por exemplo integridade referencial), sendo um sistema escalável, fiável, com uma boa performance, e que possui uma larga comunidade de utilizadores, em acentuado crescimento, o que nos permitiu uma aprendizagem e depuração mais rápidas das situações problemáticas com que nos íamos deparando.

Este sistema de gestão de base de dados suporta SQL92, SQL99 e disponibiliza algumas propriedades fundamentais:

- *Queries* complexas;
- Chaves estrangeiras;
- *Triggers*;
- *Views*;
- Integridade transaccional;
- Controlo de concorrência;

2.1.4 Jabber

O Jabber [6] é um protocolo de comunicação, definido em XML [12], que disponibiliza um conjunto de funcionalidades potentes e extensíveis. Recentemente, foi aprovada a criação de um grupo de trabalho do IETF (*Internet Engineering Task Force*) para a normalização deste protocolo definido, neste contexto, como XMPP [16] (*eXtensible Messaging and Presence Protocol*).

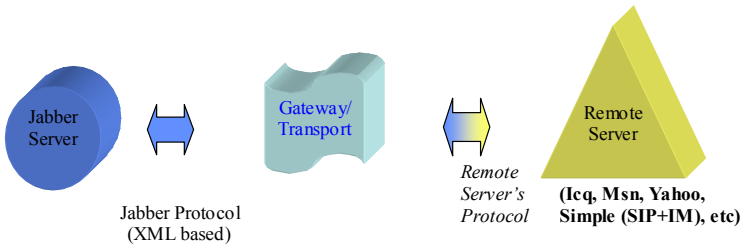

Figura 2.7 – Arquitectura Jabber

O Jabber [6] é um protocolo de "Presença e de Mensagens Instantâneas" definido e desenvolvido de acordo com o paradigma do o*pen source*. Esta tecnologia é suportada por uma grande comunidade de utilizadores e de programadores. Apesar de ser uma tecnologia o*pen source* o Jabber [6] tem um forte apoio da indústria, sendo "Jabber.com" o seu principal representante comercial.

O protocolo Jabber [6] permite efectuar operações de rede adicionais ao nível da aplicação, como sejam pesquisas de bases de dados, gestão de acesso, etc.

2.1.5 JWMA

O JWMA [17] (*Java Web Mail Architecture*) é um projecto Java *open-source* que oferece uma solução de correio electrónico com interface Web. Suporta-se nomeadamente nas

seguintes tecnologias/API: *Java Mail API,* para acesso ao servidor de *email*; e *Java Server Pages* e *Java Servlet API*, para gestão e produção de interfaces Web.

A figura seguinte ilustra a arquitectura do JWMA [17], a qual segue o padrão MVC (*Model-View-Control*), padrão arquitectural bastante conhecido que distingue claramente componentes de modelo, de componentes de controlo, e de componentes da apresentação.

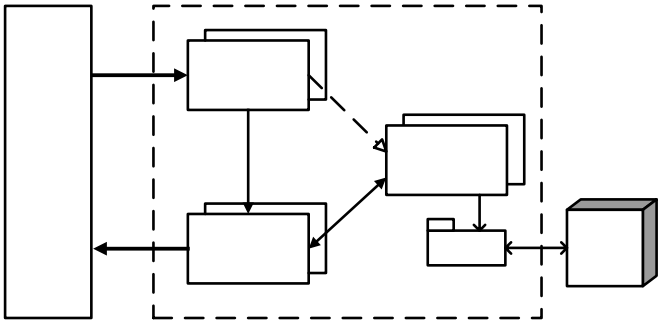

Figura 2.8 – Arquitectura de software do JWMA

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 e é através deles que é realizado o acesso ao servidor de *email*. As *JSPs* (*Java Server Pages*) desempenham o papel de componentes de apresentação e através delas são criadas as páginas HTML com que o utilizador vai interagir.

Request Servlet (Controller) Browser Redirect Response JSP (View)

2.2 Aplicações Semelhantes

Em seguida vamos falar de algumas aplicações com ideologia e funcionalidades semelhantes ao sistema PUC.

2.2.1 Alcatel e-Communication Center

Um dos produtos de referência na área das comunicações unificadas é o *Alcatel e-Communication Center* [18]. O *e-Communication Center* é um conjunto de aplicações, que usa as mais recentes tecnologias nesta área. Estas aplicações permitem aos utilizadores do sistema, o controlo e a gestão de chamadas, mensagens, directório, usando para tal qualquer interface móvel ou fixa. O *Alcatel e-Communication Center* [18] está dividido em quatro grupos de aplicações.

- *unified messaging*;
- *softphone* (uma aplicação de telefonia para PC);
- assistente pessoal;

- *Personal Information Manager* (PIM);

Figura 2.9 – Visão geral do e-Communication Center da Alcatel

2.2.1.1 Unified messaging

Esta é uma ferramenta VXML que permite interactuar com o Microsoft Exchange™ e com o Lotus Domino™. É uma aplicação que lida com todos os tipos de mensagens, apenas com uma caixa de correio multimédia.

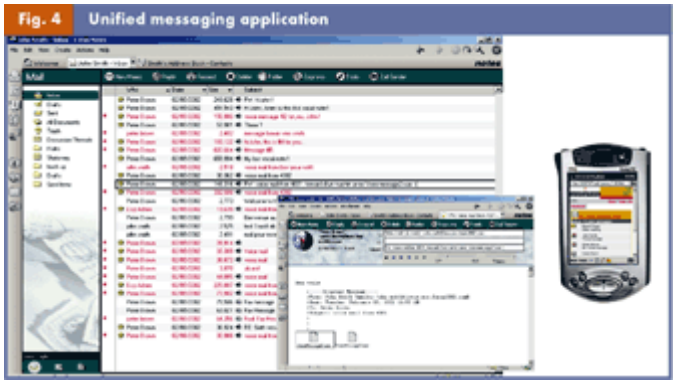

Figura 2.10 – Unified Messaging da Alcatel

2.2.1.2 Assistente Pessoal

Esta ferramenta Web e VXML interactua directamente com Microsoft Exchange/Outlook™ e Lotus Domino/Notes™. Inclui um gestor de chamadas, que gere que chamadas são prioritárias e para onde devem de ser dirigidas. Ao utilizador é dada a oportunidade de configurar e activar o reencaminhamento de chamadas, dependendo da hora do dia, da pessoa que está a ligar e de eventos associados a si.

Figura 2.11 – Assistente Pessoal da Alcatel

2.2.1.3 Personal information manager (PIM)

Esta aplicação serve de *frontend* ao Microsoft Exchange/Outlook™ e Lotus Domino/Notes™ sendo o organizador de contactos, agenda e calendário.

Figura 2.12 – PIM da Alcatel

2.2.2 Chandler

O Chandler [19] é um projecto da OSAF (*Open Source Applications Foundation*) e é um PIM (*Personal Information Manager*) criado com a intenção de ser usado no dia a dia nas funções de informação e comunicação, como compor e ler *emails*, gerir eventos de calendário e manter uma lista de contactos. Devido à facilidade com que os utilizadores do Chandler [19] podem partilhar informação com outros utilizadores, este sistema pode ser denominado como o primeiro Gestor de Informação Interpessoal.

O objectivo do Chandler [19] é ser um PIM *open source* para as funcionalidades de *email*, calendário, contactos, notas, e gestão de informação geral, mas também uma plataforma para o desenvolvimento de aplicações de gestão de informação. Presentemente este projecto está ainda em fase de desenvolvimento.

Podemos então dizer que os atributos chave deste projecto são a facilidade de partilha de informação, a possibilidade de os utilizadores trabalharem independentemente de administração e servidores externos e um compromisso da qualidade geral do produto.

De seguida apresentam-se algumas das características do Chandler [19].

2.2.2.1 Gestão de Informação Geral

Com o Chandler [19], os utilizadores serão capazes de organizar diversos tipos de informação de sua conveniência. O sistema terá não só uma grande habilidade de associar e interligar itens, mas também de juntar e recolher itens relacionados num só lugar criando "vistas" de muitos tipos de dados sensíveis ao contexto, misturando e combinando *emails*, *mailing lists*, mensagens instantâneas, notas, contactos, tarefas, *blogs*, páginas web, documentos, apresentações, favoritos, fotos, mp3, etc. A informação é guardada em repositórios na máquina local do utilizador, noutras máquinas e em recursos partilhados como servidores.

Esta é uma aproximação bastante diferente comparativamente com a dos PIM actuais. Por exemplo, os utilizadores normalmente podem apenas ver um dado *email* numa certa pasta, agrupado apenas com outros *emails*. No mundo centrado no utilizador do Chandler [19], a base da ligação dos itens está nas mãos do utilizador e é pura e simplesmente facilitado em detrimento de ser imposto por software.

O Chandler [19] permite ao utilizador estar de olho em muitas actividades que estejam em andamento. A gestão da extensão de actividades ao longo do tempo requere a habilidade de recolher apenas os conjuntos certos de itens relacionados. Demasiada informação e o utilizador não consegue encontrar o que é relevante, demasiado pouca informação e o item necessário não está presente. Esta habilidade de recolher a informação relevante a partir de diferentes fontes está no centro do desenho deste sistema. Um utilizador pode, por exemplo, relacionar uma variedade de *emails*, contactos, documentos, eventos de calendário numa colecção *ad-hoc* relacionada com um projecto específico.

O Chandler [19] permite um mecanismo de busca unificada sobre toda a informação do utilizador, tanto no seu computador pessoal como em dados guardados noutros repositórios pela rede, permitindo ainda salvar as buscas para poderem ser reutilizadas.

Os utilizadores que não sabem programar têm a possibilidade de modificar e estender o programa de várias formas, por exemplo, providenciando maneiras de gerir automaticamente tarefas complexas e assíncronas como organizar uma reunião e responder automaticamente a eventos à medida que são marcados.

2.2.2.2 Power Email

Com o Chandler [19] os utilizadores têm a capacidade de gerir grandes volumes de *emails* mais eficientemente e efectivamente do que com os clientes de correio actuais. Além disso para reduzir o número de teclas pressionadas e acções do rato necessárias para processar mensagens, o sistema tem ferramentas fáceis de usar para assistir os seus utilizadores na organização das suas mensagens.

Em particular, é reconhecido que várias pessoas usam o *email* como uma forma muito reduzida de lista de afazeres, deixando mensagens num local bastante visível até terem sido lidas, realizadas e/ou respondidas. Este sistema tem características explícitas para facilitar os aspectos de "gestão de tarefas" do *email*.

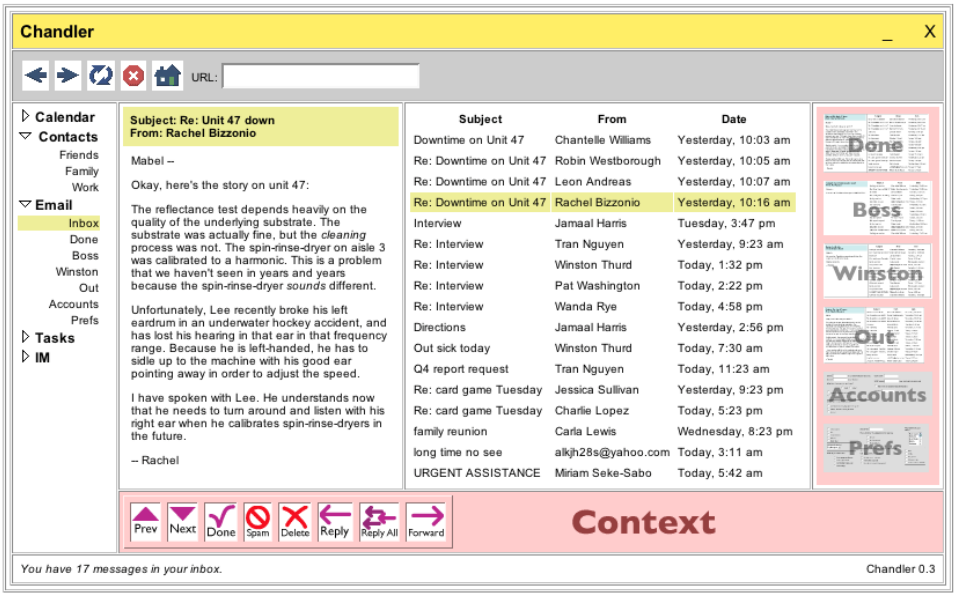

Figura 2.13 – Módulo de email do Chandler

2.2.2.3 Calendar

O Chandler [19] permite aos utilizadores partilhar uma parte da informação do seu calendário (incluindo tempo ocupado/livre) numa base *ad-hoc*. Em particular, hoje em dia os utilizadores podem apenas partilhar e usar tempo livre/ocupado com outros utilizadores, recorrendo a sistemas caros e centralmente geridos. O sistema de calendário *peer-to-peer* deste sistema permite que qualquer subgrupo de utilizadores (por exemplo uma pequena empresa, um grupo de investigação, etc.) marque eficientemente reuniões, consulte os calendários dos outros e veja as sobreposições dos múltiplos calendários simultaneamente.

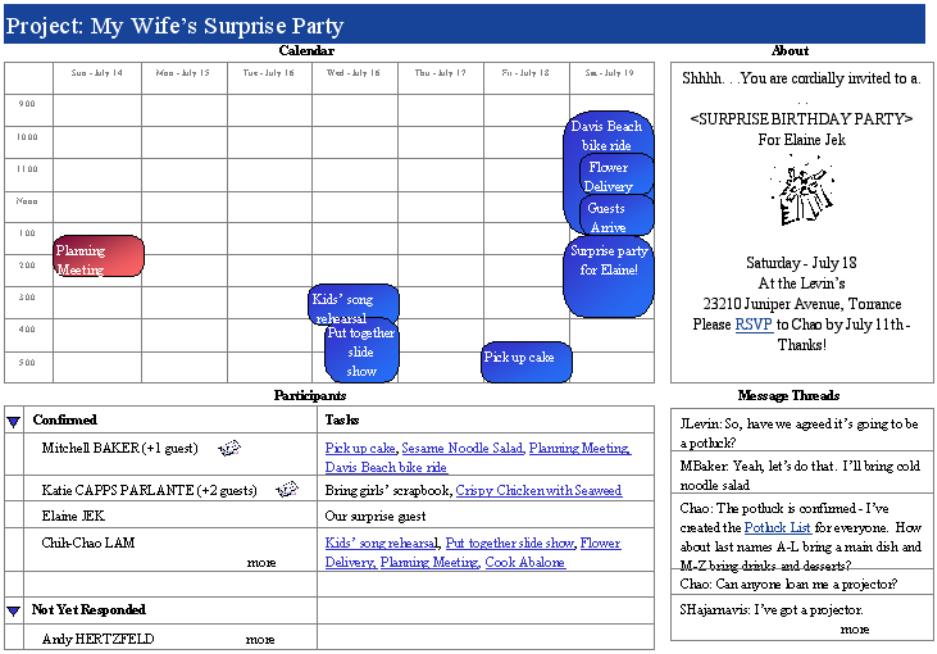

Figura 2.14 – Módulo de calendário do Chandler

2.2.2.4 Partilha e Colaboração

O Chandler [19] faz com que seja extremamente fácil partilhar informação, não apenas calendários, permitindo uma aproximação mais próxima, redução de esforço e coordenação de actividades.

O sistema facilita a sincronização de informação e actualização entre repositórios. Se um utilizador subscreveu o contacto de outro utilizador, e essa informação muda, então essas alterações serão automaticamente propagadas para o primeiro utilizador actualizando o seu repositório.

O ambiente de colaboração do sistema pode ser usado para facilitar a discussão, organização e coordenação de projectos, criar e rever documentos, e gerir a corrente de informação e tarefas. Este ambiente de construído tendo por base duas capacidades coordenadas: enviar e receber mensagens que formem um fluxo (*email*, *blogs*), criando, revendo e publicando documentos.

2.2.2.5 Chandler como uma plataforma

O Chandler [19] não está limitado às funcionalidades que a OSAF oferece. Para além de ser uma poderosa aplicação, é também uma plataforma extensível que pode ser instalada e depois estendida por programadores e utilizadores finais de várias maneiras.

Os utilizadores finais podem criar vistas customizadas do universo de dados aos quais têm acesso, misturando dados locais com remotos com políticas de partilha sofisticadas para

permitir o controlo de acesso. Pode-se ainda alterar o aspecto da aplicação, mudar as cores, imagens, adicionar novos botões, itens do menu e funcionalidades personalizadas. Finalmente os utilizadores podem ensinar agentes a realizar acções complexas que têm lugar num certo espaço de tempo.

Os programadores podem criar novos pacotes para gerir qualquer tipo de informação. Os próprios pacotes são escritos em *wxPython* (*Python* + *wxWindows*) e podem fazer praticamente tudo e têm acesso a todos os subsistemas. Pacotes podem conter agentes que podem automaticamente responder a eventos para benefício do utilizador.

Os blocos que constituem o sistema podem ser usados independentemente da aplicação. Os programadores podem assim utilizar as *frameworks* de rede, partilha e segurança e a *framework* de interfaces com o utilizador para criar uma nova geração de aplicações Internet centradas na informação.

2.2.3 Elefante

O Elefante [20] trata-se de uma aplicação que funciona como um serviço de agenda virtual e que oferece nos seus serviços um calendário, disponível na vista diária, semanal e mensal, no qual o utilizador pode obter informação sobre tarefas agendadas para os vários dias. O utilizador tem ainda a possibilidade de fazer a marcação de tarefas e notas de diferentes tipos como a marcação de aniversários, compromissos, no qual existe um largo leque de tipos de compromissos, de pagamentos como o pagamento da conta de telemóvel ou ainda da marcação de datas especiais. Estão também disponíveis aos utilizadores variados serviços de informação como serviços de noticias, horóscopo, entretenimento, finanças e previsões do tempo. Para além destas funcionalidades disponibilizadas, existe ainda a possibilidade de guardar uma lista de contactos.

Relativamente à acessibilidade a este serviço, esta é feita principalmente através de uma interface Web. No entanto existe ainda a possibilidade de aceder através de WAP no qual o Elefante [20] é pioneiro no desenvolvimento deste tipo de tecnologia para Internet sem fios. O sistema permite ainda fazer sincronização de dados com um PDA. Para tal é usado um módulo *Hands*, que permite, via *HotSync*, o fluxo de informações sobre compromissos, tarefas, aniversários, datas comemorativas e horóscopo da agenda do sistema para o Palm. Uma das vantagens é que no caso de já existir uma agenda no PDA, os dados são adicionados à agenda sem apagar os anteriores. O sistema permite ainda o envio de mensagens de sms para aviso sobre tarefas, ou notícias de diferentes tipos.

2.2.4 Weblicon

O PIM Weblicon [21] é um *organizer online*, que possui as funcionalidades de conversação por mensagem e gestão de informação pessoal. Este sistema inclui Email, SMS, GroupSMS e MMS bem como lista de contactos integrada, calendário, lista de tarefas e sincronização utilizando SyncML. A arquitectura de multi-acesso suporta o acesso Web via HTML, ou por um *applet* Java, acesso móvel através de WAP ou um cliente J2ME [2] para telemóveis com suporte de Java e ainda acesso através uma interface VoiceXML para acesso por voz ao *organizer*. A figura seguinte mostra então esta arquitectura do sistema.

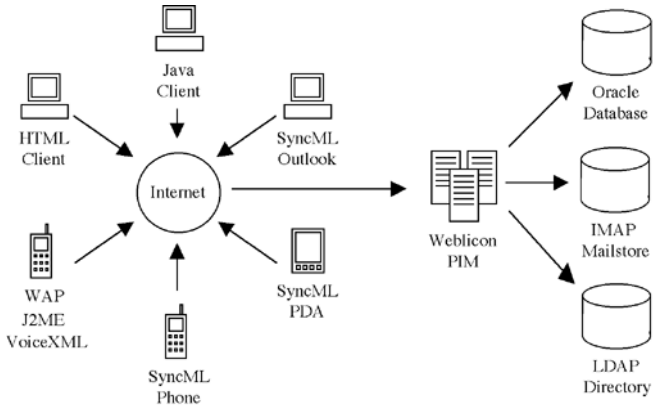

Figura 2.15 – Arquitectura multi-acesso do Weblicon

2.2.4.1 HTML e WAP

As interfaces HMTL e WAP do Weblicon [21] são extremamente fáceis de utilizar uma vez que se assemelham bastante aos PIMs que o utilizador está acostumado a utilizar. A este produto foi recentemente atribuído o prémio "*Usability* 2010" que certifica a simplicidade da interface utilizador proporcionando uma interface bastante intuitiva.

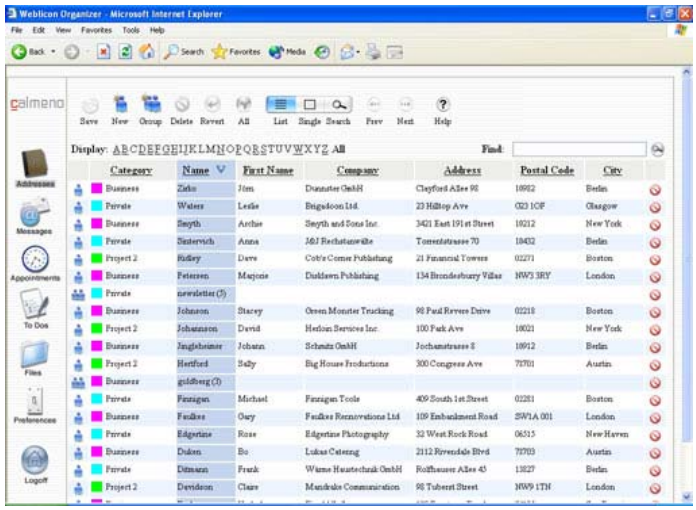

Figura 2.16 – Aspecto da interface Web do Weblicon

2.2.4.2 Applet Java

O cliente Java é único no mundo dos PIM, e permite *drag&drop*, manipulação directa e uma interface utilizador bastante atractiva que é capaz de competir com os principais produtos PIM de desktop.

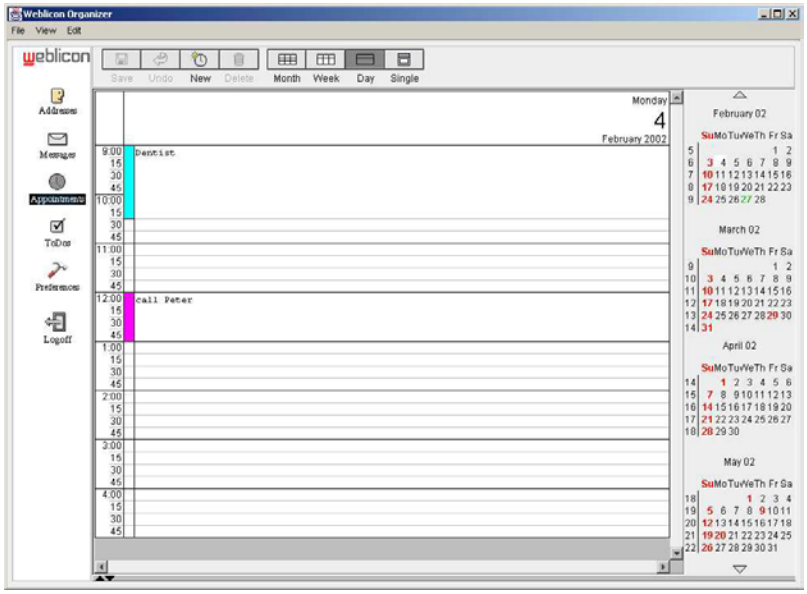

Figura 2.17 – Interface Java do Weblicon

O cliente Java proporciona o melhor das aplicações desktop e Web uma vez que permite uma utilização similar a uma aplicação desktop e ao mesmo tempo armazena os dados no servidor central disponibilizando o acesso a outros clientes.

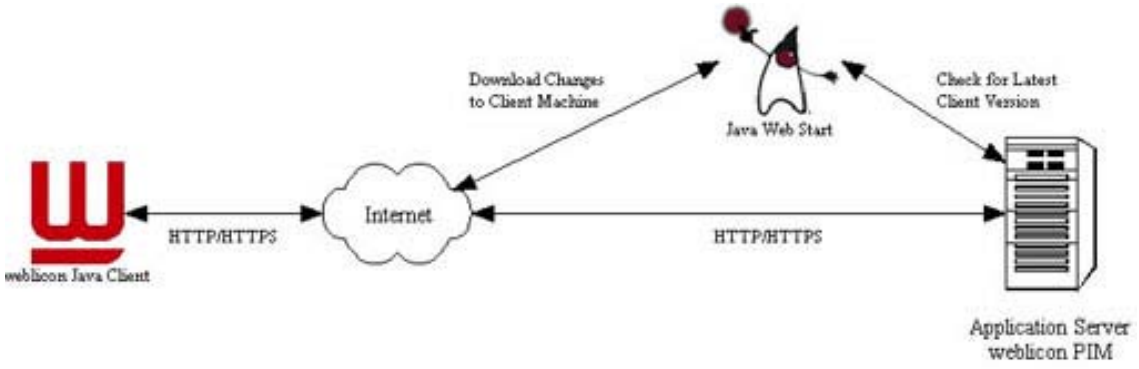

Figura 2.18 – Arquitectura geral do cliente Java do Weblicon

2.2.4.3 SyncML

A tecnologia certificada SyncML usada pelo Weblicon [21] permite a integração do PIM alojado centralmente com as aplicações mais populares como o Microsoft Outlook, Palm PDAs, PocketPCs. Graças ao *standard* SyncML, o Weblicon [21] é o primeiro fornecedor de

tecnologia na Europa a disponibilizar sincronização *Over-The-Air* (OTA) a telemóveis com suporte a SyncML.

2.2.4.4 Cliente VoiceXML

Na versão actual da interface voz, o Weblicon [21] permite ao utilizador aceder ao seu *email* e proximamente irá permitir acesso a todas as funcionalidades PIM. Os utilizadores ligam para um número de telefone central e falam com uma "secretária virtual", que é capaz de entender comandos de voz simples e lê ainda os *emails* do utilizador.

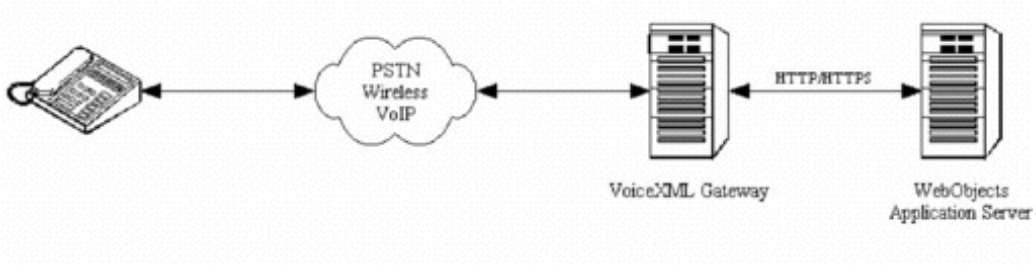

Figura 2.19 – Visão geral do funcionamento do portal de voz do Weblicon

2.2.4.5 Cliente J2ME

Para completar a sua linha de produtos, o Weblicon [21] possui um cliente J2ME [2] o qual proporciona uma melhor interacção com o utilizador para os telemóveis Java da geração 2.5. Os utilizadores podem fazer o *download* do cliente J2ME [2] para o seu telemóvel e usar uma interface gráfica de utilizador apelativa para aceder aos dados do sistema. Este cliente é o acesso móvel da nova geração superando por muito o WAP em termos de usabilidade.

Figura 2.20 – Aspecto do cliente J2ME do Weblicon

2.3 Conclusão acerca do estado da arte

Hoje em dia, o negócio de qualquer grande empresa envolve a interacção entre clientes e empregados. Para haver competição no mercado, para encarar as necessidades dos clientes e para melhorar o fluxo de comunicação, bem como enfrentar os novos desafios (mobilidade, tempo de resposta, etc), as empresas necessitam de mudar os seus sistemas de comunicações. Para fazer face a esta realidade os sistemas de comunicação unificada, terão com certeza um papel fundamental no sucesso de qualquer grande empresa.

Após uma análise detalhada de várias aplicações de funções semelhantes ao PUC, concluímos que o desenvolvimento de software com este tipo de funcionalidades está a receber uma atenção bastante grande por parte das grandes empresas de software. Estas empresas aperceberam-se da existência de um grupo cada vez maior de possíveis clientes com a necessidade de acesso a este tipo de aplicações em qualquer lugar e a partir de diferentes dispositivos.

De notar especialmente os sistema Weblicon [21] e Chandler [19], que se assemelham bastante ao objectivo proposto para o PUC uma vez que, no caso do Weblicon [21] possui para além da interface Web a possibilidade de utilização utilizando apenas a voz ou também a partir de dispositivos com suporte a J2ME [2] e no caso do Chandler [19] para além de funcionalidades semelhantes ser também um projecto com uma ideologia *open-source*.

3 Visão Geral do Projecto PUC

O PUC ("*Personal Unified Communications*") é um conceito que faz cada vez mais sentido nos nossos dias em que os utilizadores possuem várias contas em diferentes tipos de serviços de comunicação pessoal como sendo, o *email*, os serviços de *chat*, os livros de endereços e agenda, etc. em que por exemplo não é possível utilizar apenas uma lista de contactos em todas as contas com as implicações que tal tem ao nível da duplicação de informação e na actualização desta.

Este trabalho teve como objectivo a integração de vários módulos já desenvolvidos no âmbito do projecto PUC e ainda o desenvolvimento de novas funcionalidades do sistema. Dos módulos já desenvolvidos encontravam-se os módulos myAgenda, myCom e as interfaces para dispositivos móveis.

O sistema resultante pode-se dividir em duas aplicações distintas mas que partilham entre si os dados que descrevem as diversas entidades presentes no conceito PUC. As aplicações designam-se por myAgenda (assistente pessoal de agenda) e myCom (assistente pessoal de comunicações).

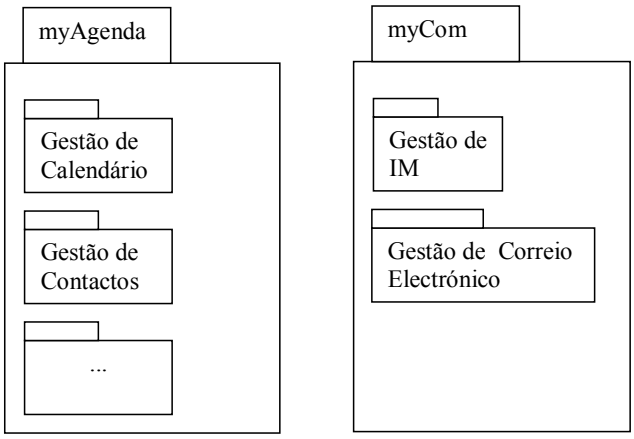

Figura 3.1 – Pacotes de funcionalidade de cada uma das aplicações

Com a aplicação myAgenda pretende-se disponibilizar as funcionalidades normais de um PIM ("*Personal Information Manager*") tais como gestão de contactos, calendário, notas e favoritos. Relativamente à aplicação myCom esta disponibiliza a gestão da conta de *email* do utilizador bem como da conta de *instant messaging* (*chat*).

O sistema permite o acesso ao PUC não só a partir de um PC utilizando um browser, mas também a uma gama de dispositivos móveis com suporte a J2ME [2], capacidade que pode ser considerada como praticamente um *standard* nos telemóveis mais recentes. Para além do acesso através destes dispositivos, o sistema pretende ainda que seja possível a interacção do

utilizador com a aplicação através da voz utilizando o *standard* VoiceXML, acesso este que não se encontra no âmbito deste trabalho.

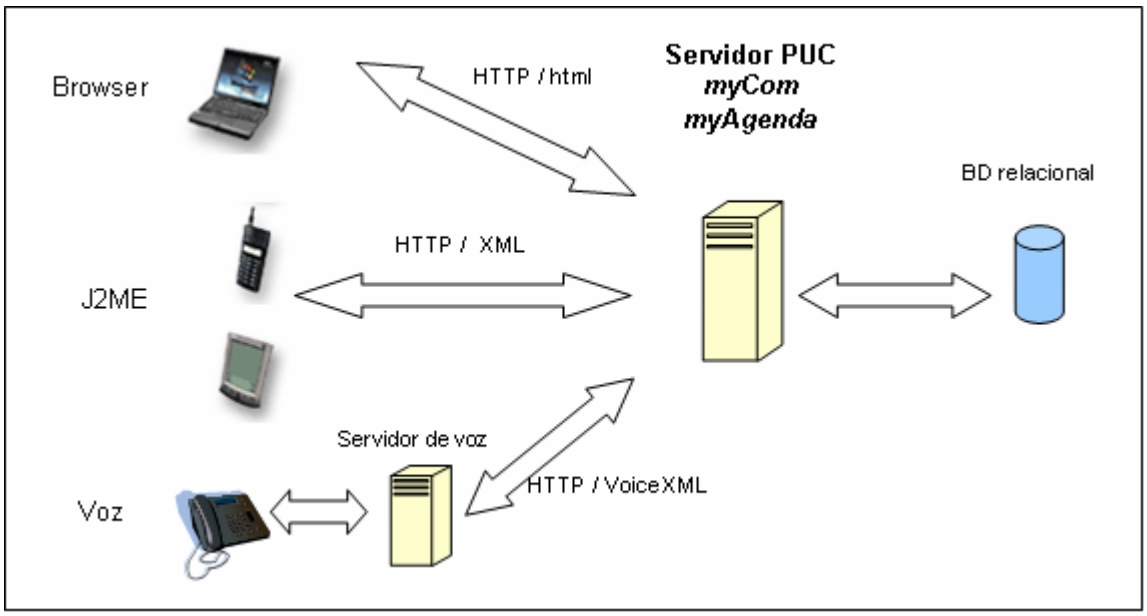

Figura 3.2 – Arquitectura tecnológica do projecto PUC

Como a figura anterior mostra, toda a informação contida no PUC, é mantida num servidor central. Tal permite a utilização em diferentes locais sem necessidade de o utilizador transportar essa informação com ele. Existe apenas um caso que nem toda a informação do servidor está permanentemente actualizada, tal acontece na utilização através do telemóvel com o cliente J2ME [2], em que só quando o utilizador o indica explicitamente é que a informação contida no servidor é actualizada com a do telemóvel e vice versa.

Este trabalho foca-se na integração das várias componentes desenvolvidas até ao momento, excepto a componente de acesso por voz. Para além desta integração, é da nossa responsabilidade o processo de sincronização entre os dispositivos móveis e o servidor e ainda a implementação de novas funcionalidades como sejam a gestão de notas e favoritos, a possibilidade de os eventos do calendário possuírem vários participantes e se poder programar repetições desses mesmos eventos, a modificação da representação dos contactos, etc.

3.1 Análise de Requisitos

Nesta secção apresenta-se, sobre a forma de casos de utilização, uma análise detalhada das funcionalidades do projecto PUC.

3.1.1 Actores

A aplicação PUC será utilizada por três tipos de utilizadores, cada um associado a um determinado perfil de utilização. Na figura seguinte estão ilustrados os três tipos de actores previstos assim como as relações entre si.

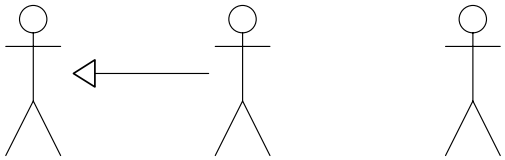

Figura 3.3 – Actores

O UAnónimo é um actor com funcionalidades limitadas, podendo apenas obter informação sobre os serviços oferecidos e de consequente subscrição desses serviços.

O actor UMembro é uma particularização do actor UAnónimo, uma vez que tem acesso a todas as operações disponíveis para o utilizador UAnónimo, mais as funcionalidades dos serviços subscritos por si.

O actor UGestor tem acesso completo relativamente à gestão de contas dos membros do sistema, tendo só acesso à informação de registo de todos os utilizadores.

3.1.2 Casos de Utilização do Actor UAnónimo

O actor UAnónimo só pode aceder às funcionalidades de subscrição de serviços e de consulta de informação do sistema. Estas acções são realizadas única e exclusivamente através da interface Web.

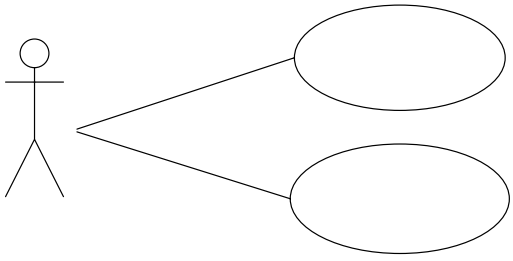

Figura 3.4 – Casos de utilização do actor UAnonimo

3.1.3 Casos de Utilização do Actor UMembro

O actor UMembro acede ao sistema com o objectivo de realizar um conjunto de operações, que se podem dividir nos seguintes pacotes complementares de casos de utilização: (1) gestão de notas; (2) gestão de favoritos; (3) gestão de calendário; (4) gestão de contactos; (5) gestão

de Caixa de Correio; e (6) gestão do serviço de *instant messaging*. Os quatro primeiros encontram-se agregados na aplicação myAgenda e os dois últimos na aplicação myCom. Temos ainda os casos de utilização para a (7) gestão de preferências.

3.1.3.1 Casos de Utilização da Gestão de Notas

Relativamente às funcionalidades relacionadas com a gestão de notas e para o caso da interface web, cada membro pode realizar os seguintes casos: (1) adicionar nota (cria uma nova nota dentro da pasta actual); (2) listar notas; (3) ver detalhes de uma nota; (4) alterar nota; (5) apagar nota; (6) mover nota (permite mover uma ou mais notas para uma pasta diferente da actual); (7) criar pasta (cria uma nova pasta, dentro da pasta actual); (8) alterar pasta; e (9) apagar pasta (apaga a pasta e todas as subpastas e/ou notas que a pasta contenha).

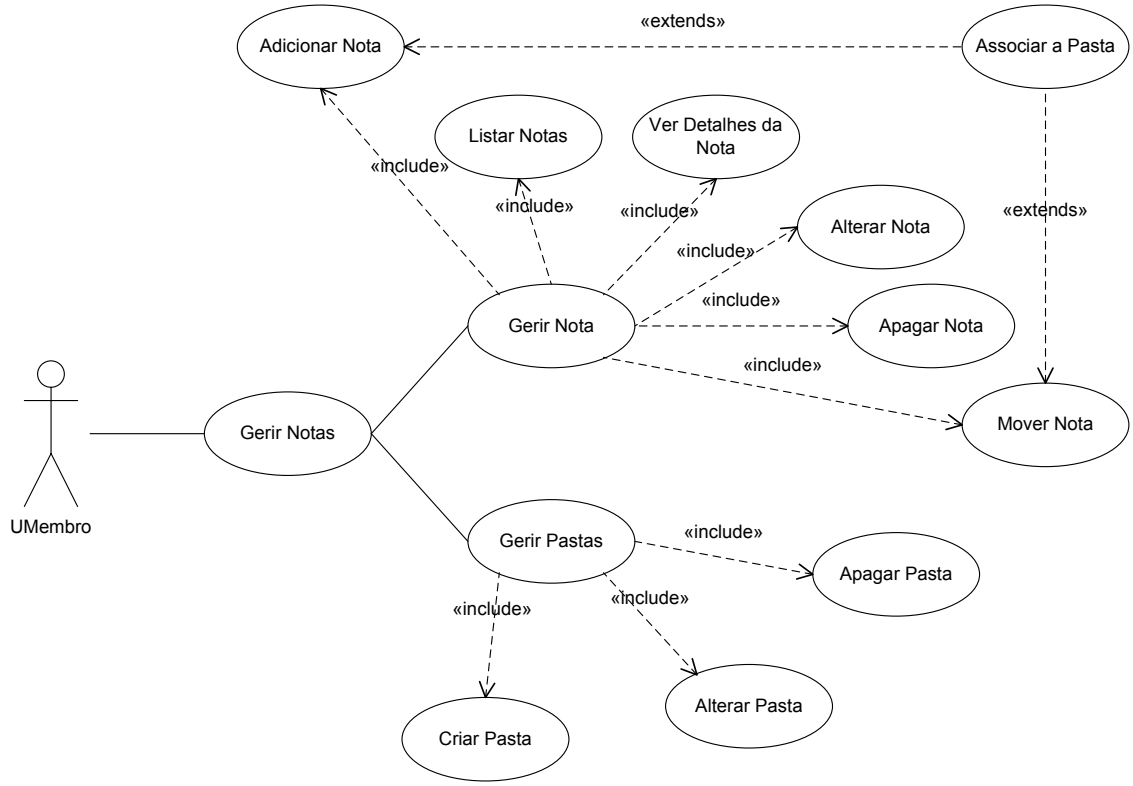

Figura 3.5 – Casos de Utilização da Gestão de Notas para a interface web

No caso das interfaces móveis o utilizador não possui as funcionalidades relativas à gestão de pastas, mas possui a funcionalidade de sincronização das notas.

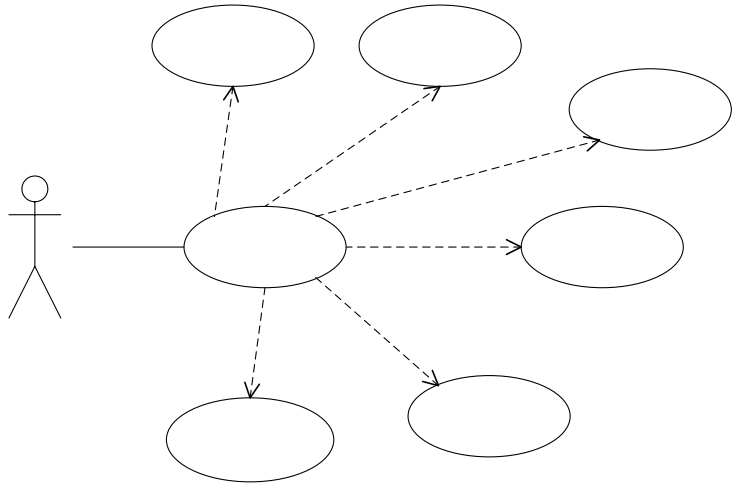

Figura 3.6 – Casos de Utilização da Gestão de Notas para a interface móvel

3.1.3.2 Casos de Utilização da Gestão de Favoritos

Relativamente às funcionalidades relacionadas com a gestão de favoritos e para o caso da interface web, cada membro pode realizar os seguintes casos: (1) adicionar favorito (cria um novo favorito dentro da pasta actual); (2) listar favoritos; (3) ver detalhes de um favorito; (4) alterar favorito; (5) apagar favorito; (6) mover favorito (permite mover um ou mais favoritos para uma pasta diferente da actual); (7) criar pasta (cria uma nova pasta, dentro da pasta actual); (8) alterar pasta; e (9) apagar pasta (apaga a pasta e todas as subpastas e/ou favoritos que a pasta contenha).

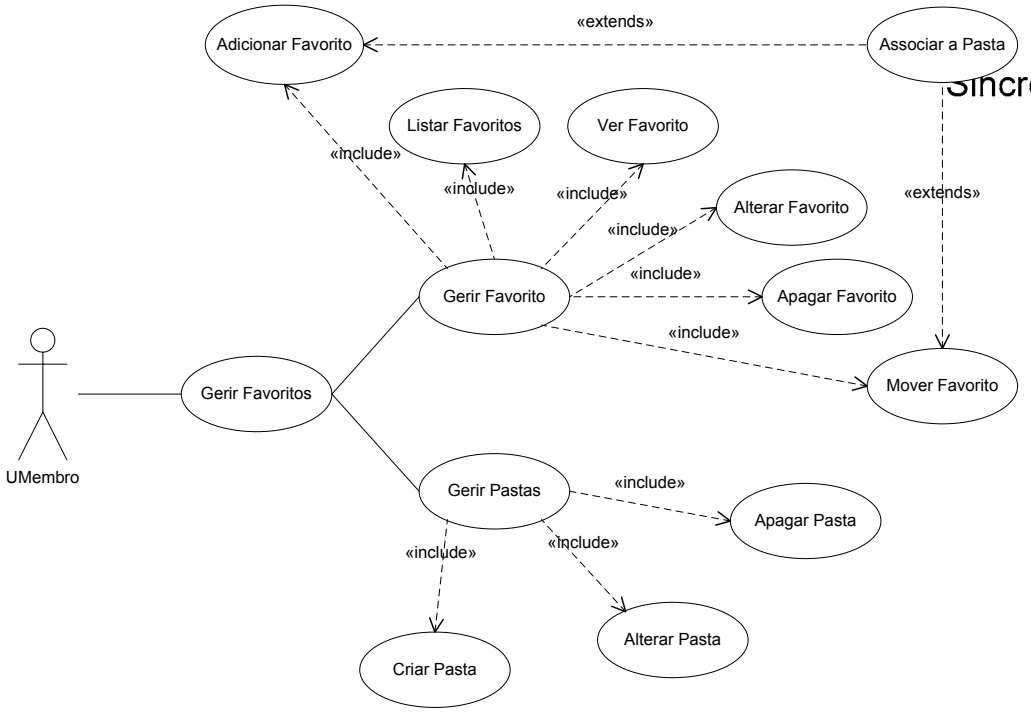

Figura 3.7 – Casos de Utilização da Gestão de Favoritos para a interface web

No caso das interfaces móveis o utilizador não possui qualquer funcionalidade relativa aos favoritos.

3.1.3.3 Casos de Utilização da Gestão do Calendário

Relativamente às funcionalidades relacionadas com a gestão de calendário para a interface web, cada membro pode realizar os seguintes casos: (1) definir modo de visualização do calendário (diário, semanal ou mensal); (2) adicionar tarefa; (3) listar tarefas; (4) ver detalhes da tarefa; (5) alterar tarefa; (6) apagar tarefa.

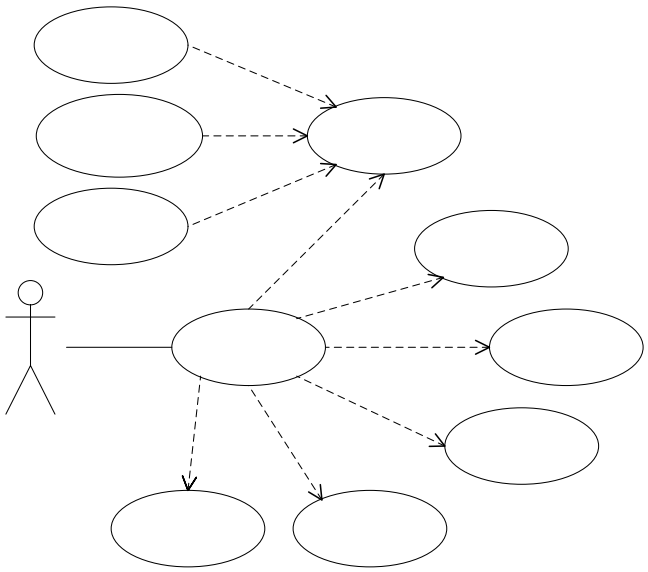

Figura 3.8 – Casos de Utilização da Gestão de Calendário para a interface web

Para as interfaces móveis estão disponíveis os mesmos casos descritos em cima e ainda o caso de sincronizar tarefas.

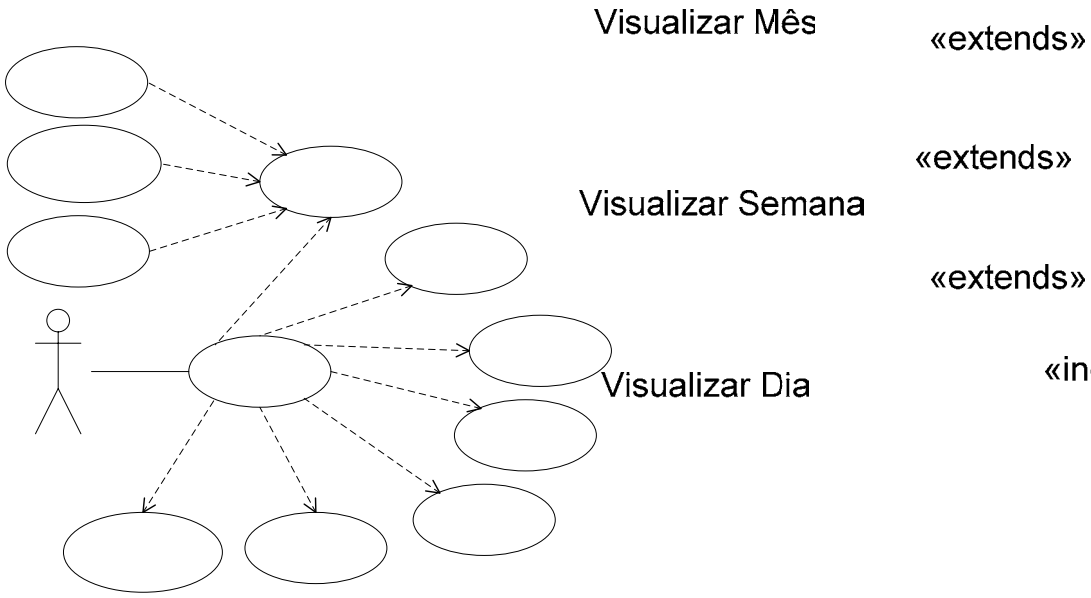

Figura 3.9 – Casos de Utilização da Gestão de Calendário para a interface móvel

3.1.3.4 Casos de Utilização da Gestão de Contactos

Na gestão de contactos os utilizadores podem realizar os seguintes casos: (1) adicionar contacto; (2) listar contactos; (3) ver detalhes de um contacto; (4) alterar contacto; (5) apagar contacto; (6) mover contacto de grupo; (7) criar grupo; (8) listar grupos; (9) alterar grupo; e (10) apagar grupo.

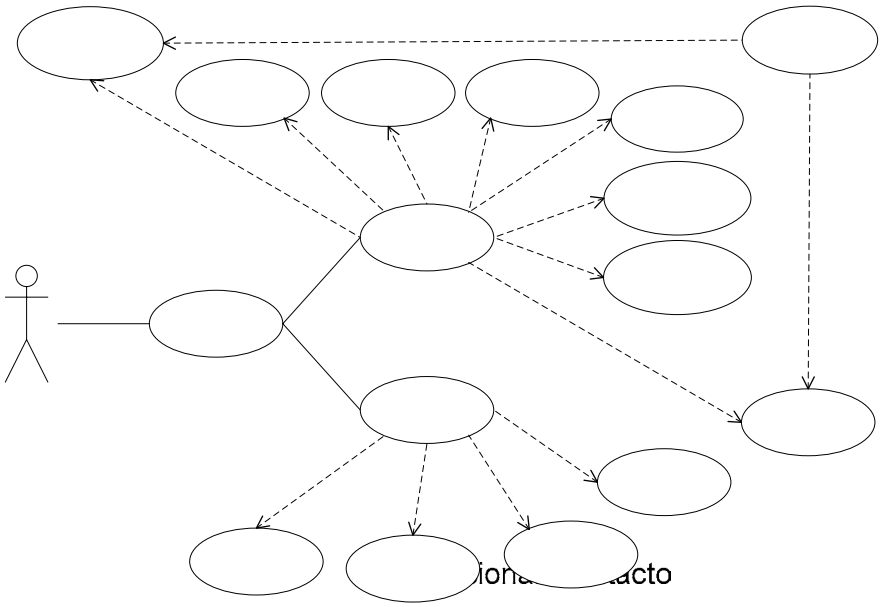

Figura 3.10 – Casos de Utilização da Gestão de Contactos para a interface web

No caso das interfaces móveis estão disponíveis os mesmos casos descritos em cima e ainda o caso de sincronizar contactos e grupos.

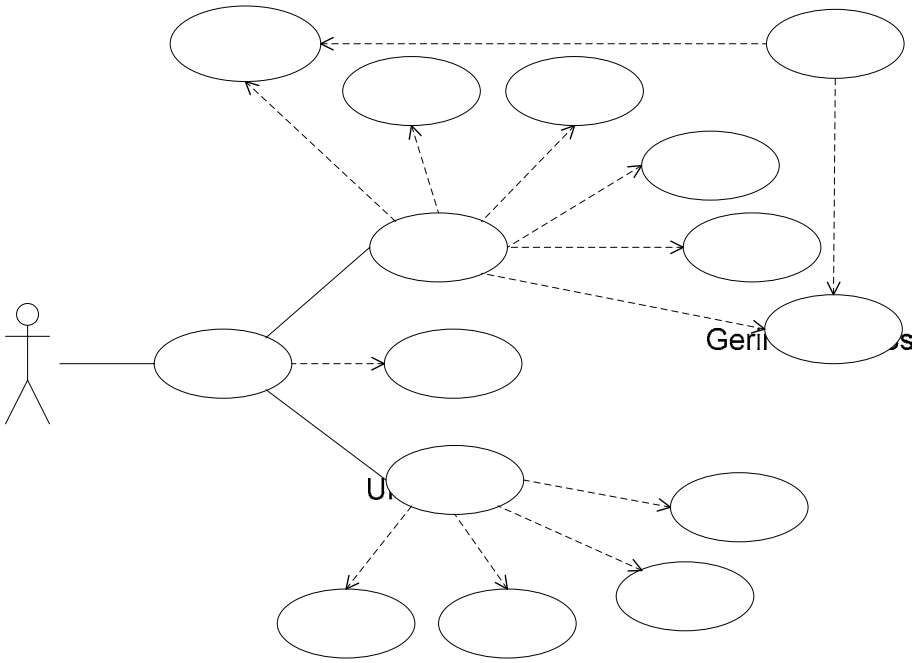

Figura 3.11 – Casos de Utilização da Gestão de Contactos para a interface móvel

3.1.3.5 Casos de Utilização da Gestão de Instant Messaging

Na gestão do serviço de *instant messaging*, e no caso de estar a aceder ao sistema através da interface web, os membros podem realizar os seguintes casos: (1) enviar mensagem; (2) receber mensagem; (3) visualizar lista de *buddies*; (4) adicionar *buddy*; (5) alterar *buddy*; (6) apagar *buddy*; (7) alterar estado de presença.

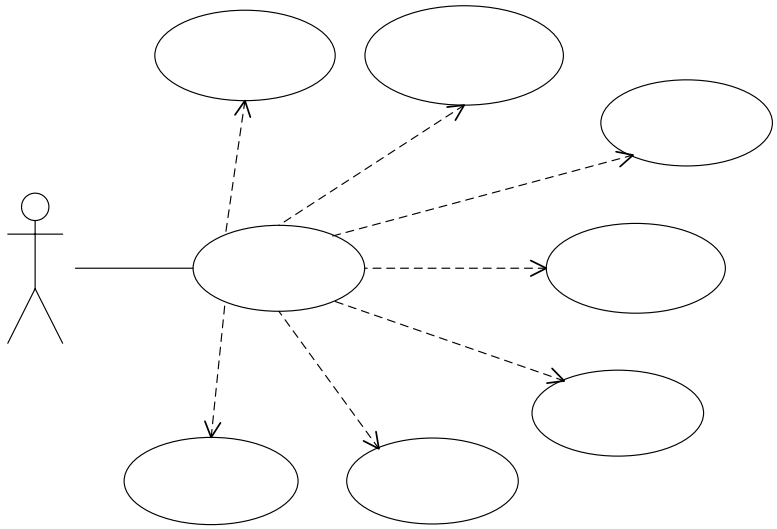

Figura 3.12 – Casos de Utilização da Gestão de Instant Messaging para a interface web

Esta opção não está disponível para as interfaces móveis.

3.1.3.6 Casos de Utilização da Gestão da Caixa de Correio

Na gestão da caixa de correio, no caso em que o utilizador acede ao sistema através da interface web, os membros podem realizar os casos: (1) enviar mensagem; (2) listar mensagens recebidas; (3) ver detalhes da mensagem; (4) mover mensagem para outra pasta; (5) apagar mensagem; (6) criar pasta (cria uma nova pasta, dentro da pasta actual); (7) apagar pasta (apaga a pasta e todas as subpastas e/ou mensagens que a pasta contenha); e (8) listar pastas.

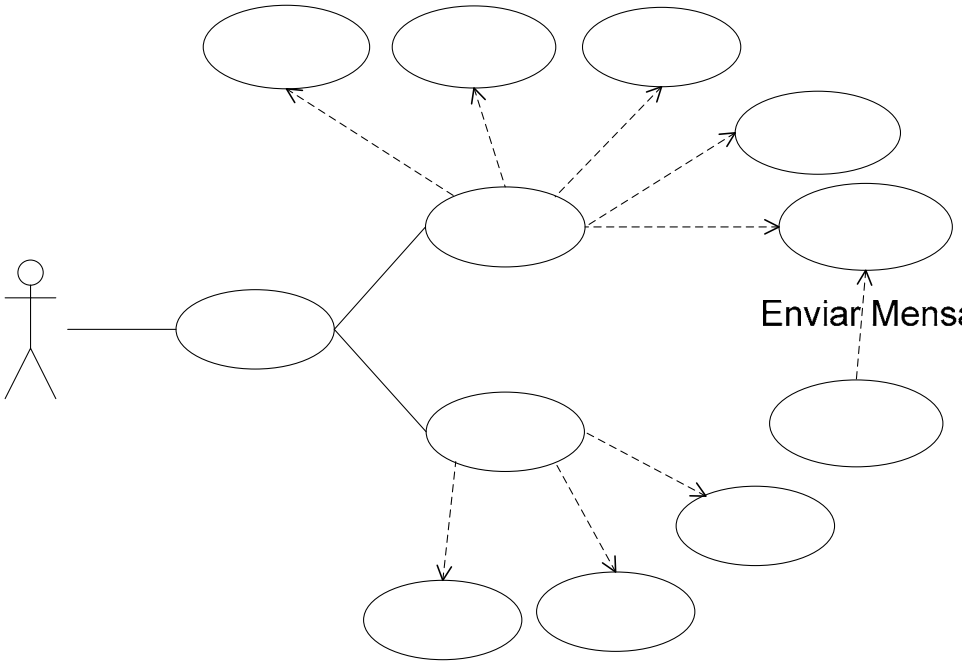

Figura 3.13 – Casos de Utilização da Gestão da Caixa de Correio para a interface web

No caso das interfaces móveis estão disponíveis os casos: (1) enviar mensagem; (2) listar mensagens; (3) ver detalhes da mensagem; (4) apagar mensagem; (5) Enviar/Receber Mensagens (Sincronizar).

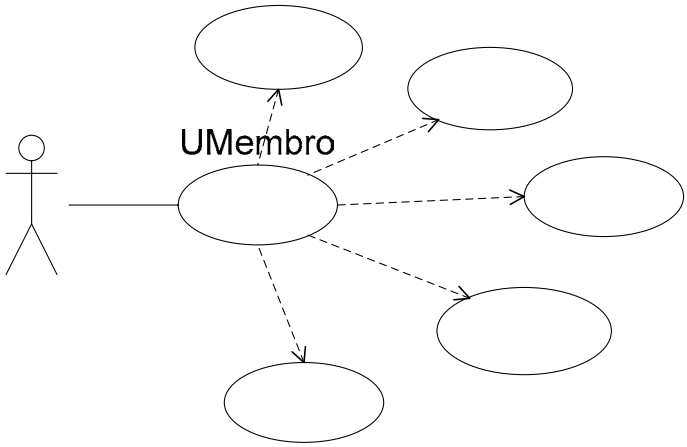

Figura 3.14 – Casos de Utilização da Gestão da Caixa de Correio para a interface móvel

3.1.3.7 Casos de Utilização da Gestão de Preferências

Na gestão de preferências, no caso em que o utilizador acede ao sistema através da interface web, os membros podem realizar os casos: (1) Subscrever um Novo Serviço; (2) Criar Categoria de Calendário; (3) Editar Categoria de Calendário; (4) Visualizar Categoria de Calendário; (5) Remover Categoria de Calendário; (6) Listar Categorias de Calendário; (7) Editar Contacto Próprio; e (8) Visualizar Contacto Próprio.

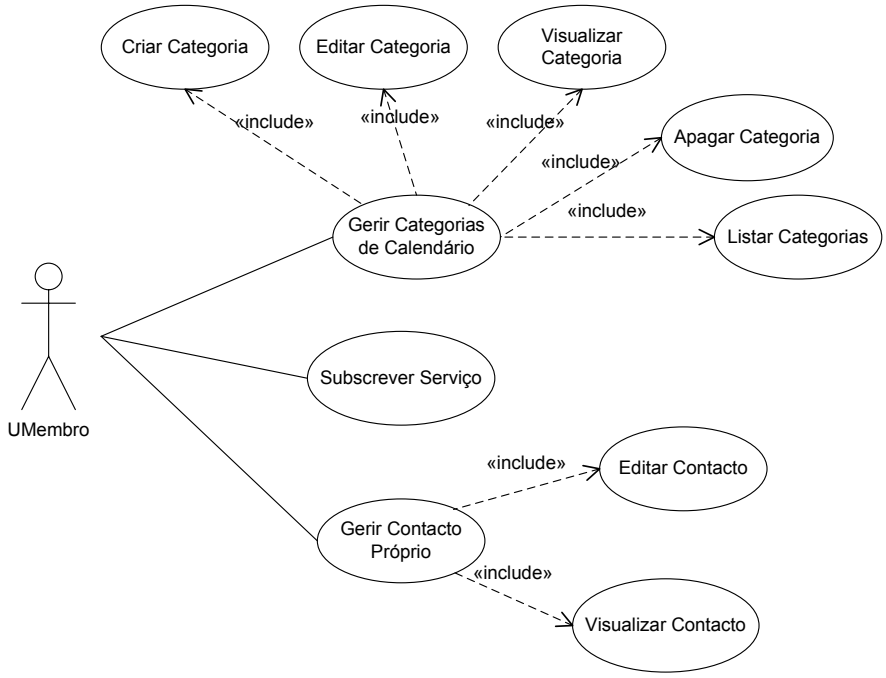

Figura 3.15 – Casos de Utilização da Gestão de Preferências para a interface web

No caso das interfaces móveis estão disponíveis os casos: (1) alterar cores da interface; (2) alterar perfil do utilizador; e (3) alterar detalhes da ligação ao sistema.

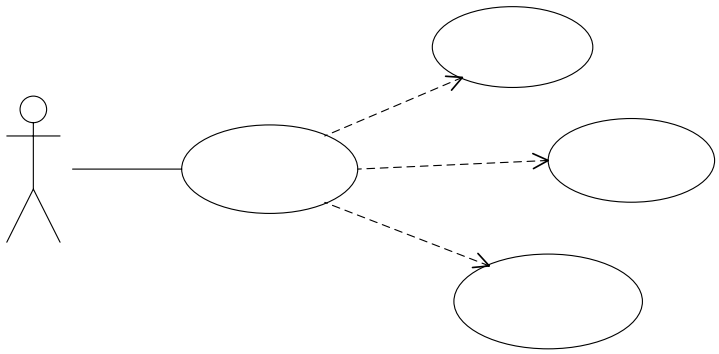

Figura 3.16 – Casos de Utilização da Gestão de Preferências para a interface móvel

3.1.4 Casos de Utilização do Actor UGestor

O actor UGestor só pode aceder às funcionalidades de administração do sistema. Estas acções são realizadas única e exclusivamente através da interface Web.

3.1.4.1 Casos de Utilização da Gestão de Subscrições

Na gestão de subscrições, no caso em que o utilizador acede ao sistema através da interface web, o gestor pode realizar os casos: (1) Validar Subscrição; (2) Suspender Subscrição; (3) Invalidar Subscrição; (4) Visualizar Subscrição; e (5) Listar as Subscrições Existentes.

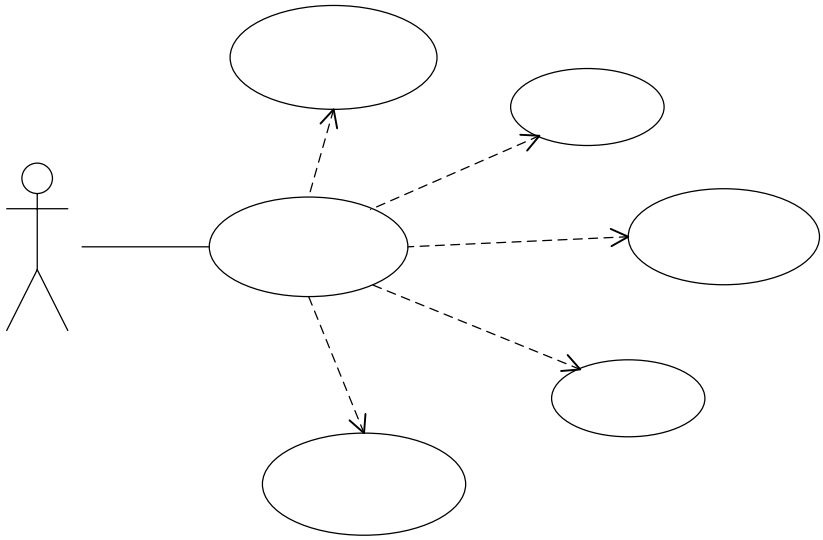

Figura 3.17 – Casos de Utilização da Gestão de Subscrições para a intercace web

Em seguida mostra-se por que estados pode passar a subscrição de um serviço no sistema.

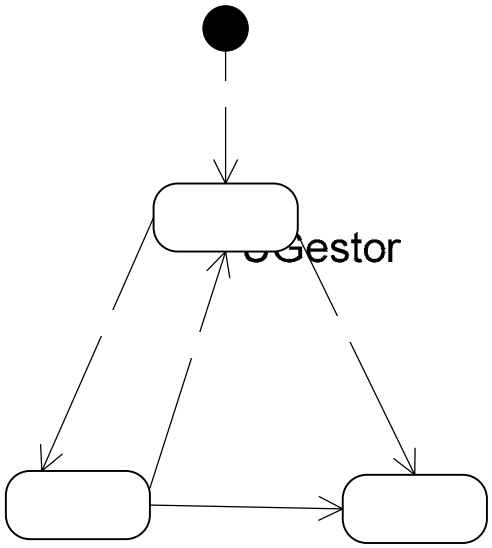

Figura 3.18 – Diagrama de estados de uma subscrição

4 Desenho e Arquitectura

Neste capítulo são descritas algumas decisões tomadas acerca das tecnologias, a arquitectura, aspectos importantes ocorridos no processo de desenvolvimento. Este capítulo encontra-se dividido em duas secções sendo elas o Servidor PUC e o Cliente PUC móvel. Esta divisão deve-se ao facto de o processo de desenvolvimento ter sido realizado ao nível do servidor e ao nível dos dispositivos móveis separadamente.

4.1 Servidor PUC

O servidor do sistema encontra-se desenvolvido na plataforma J2EE [1]. Encontra-se implementado seguindo o padrão de desenho MVC ("*Model View Controller*" = Modelo-Vista-Controlador). Para tal, utiliza-se, em geral, a *framework* aplicacional Struts [10] que facilita a utilização deste padrão uma vez que disponibiliza o seu próprio Controlador e é facilmente integrado com outras tecnologias para as componentes do Modelo e Vista. Para o Modelo, o Struts [10] pode interagir com as tecnologias *standard* de acesso a dados, como JDBC e EJB [9]. Relativamente à Vista, o Struts [10] funciona muito bem com *Java Server Pages* (JSPs).

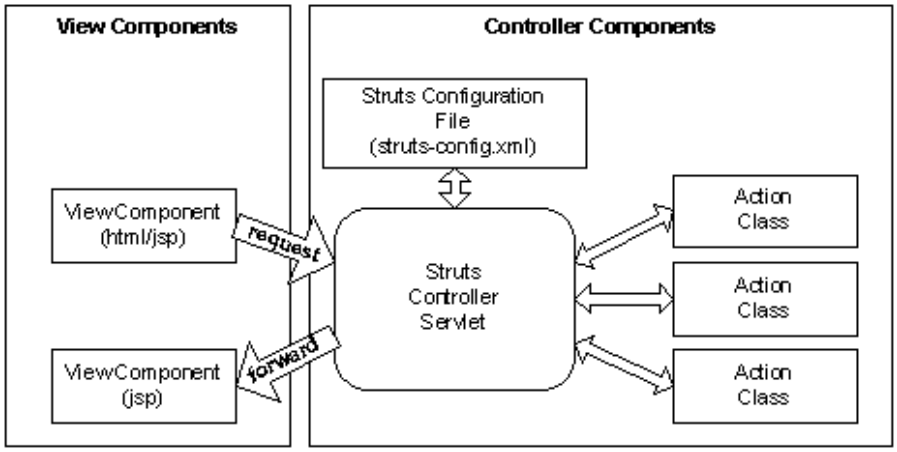

Figura 4.1 – Arquitectura da framework Struts

Este trabalho, nesta parte do sistema, centrou-se sobretudo na integração dos serviços myCom e myAgenda, adaptando o sistema myCom à arquitectura descrita nesta secção, no desenvolvimento de novas funcionalidades do sistema e ainda na adaptação deste módulo para a funcionalidade de sincronização do módulo das interfaces móveis.

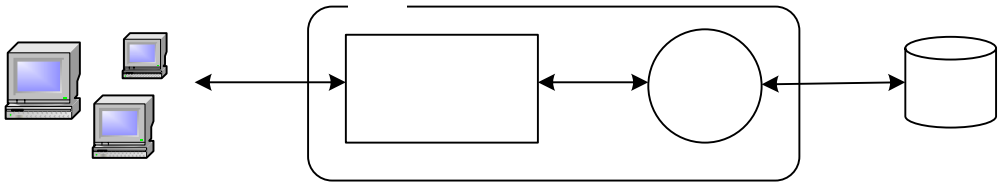

Figura 4.2 – Arquitectura simplificada do sistema

JBoss

Redesenhou-se a arquitectura do módulo myCom para a usada pelo módulo myAgenda, uma vez que esta é de bem mais fácil compreensão, actualização e suporte que a utilizada até aí. As alterações basearam-se principalmente na alteração do Controlador, que até aí era disponibilizada por *servlets* e passou-se a usar a *framework* Struts [10]. Quanto ao Modelo, continuou-se a utilizar EJB [9], mas passou-se a usar a ferramenta de geração automática de código XDoclet [4], usada já no módulo myAgenda, uma vez que facilita grandemente o desenvolvimento de todos os *beans* utilizados.

Struts

Utilizador

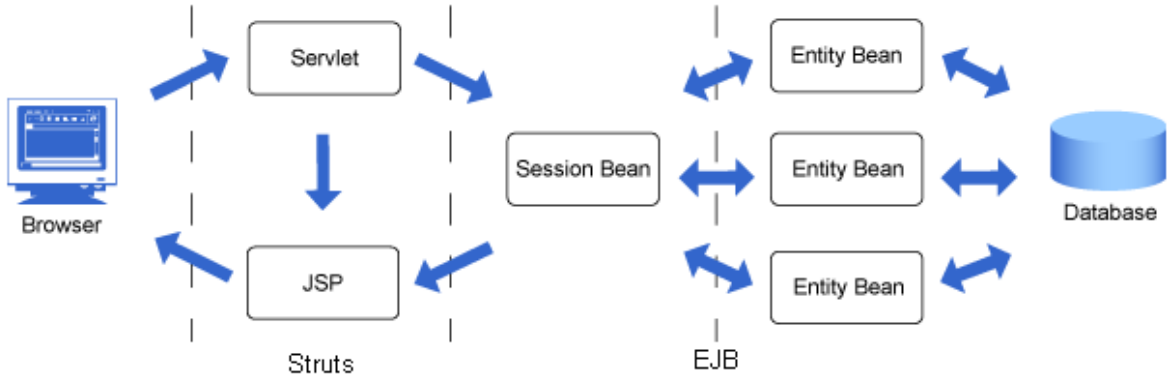

Figura 4.3 – Arquitectura do sistema

Um dos primeiros problemas aquando da junção dos dois módulos (myAgenda e myCom) foi o facto de os dois módulos usarem diferentes suportes para guardar a informação persistente. Como o módulo myCom só guardava a informação respeitante aos utilizadores utilizando um servidor LDAP, optou-se então por usar o modo de armazenamento utilizado no módulo myAgenda, isto é usando uma base de dados PostgreSQL [7]. Assim evitou-se que a informação se encontrasse duplicada em dois sítios diferentes. Esta opção levou então a outro problema, designadamente como partilhar a informação do utilizador entre os dois módulos. De modo a manter a aplicação o mais escalável possível mantiveram-se estes dois módulos como aplicações distintas, mas este facto levantou o problema de como partilhar os dados do utilizador, visto que duas aplicações distintas não podem partilhar a mesma sessão web por motivos de segurança. Assim a solução adoptada passou por colocar a informação necessária do utilizador no contexto de execução de um dos módulos visto que assim já se podia partilhar a informação aí presente entre as aplicações.

Depois de resolvido este pequeno problema seguiu-se então para o redesenho da arquitectura do módulo myCom e para a implementação de novas funcionalidades do sistema. Estas acções são então descritas nos pontos seguintes.

De realçar ainda que se fizeram algumas alterações na interface de modo a tornar esta mais perceptível e uniforme para o utilizador.

4.1.1 Módulo myCom

4.1.1.1 Email

Nesta parte do módulo myCom não foram realizadas muitas alterações. Depois de analisar o código de toda esta parte, chegou-se à conclusão que o melhor a fazer era actualizar o módulo JWMA utilizado para uma versão mais recente e então aí sim adaptar a arquitecta para a descrita anteriormente. A arquitectura actual desta parte da aplicação é a que se encontra na figura seguinte.

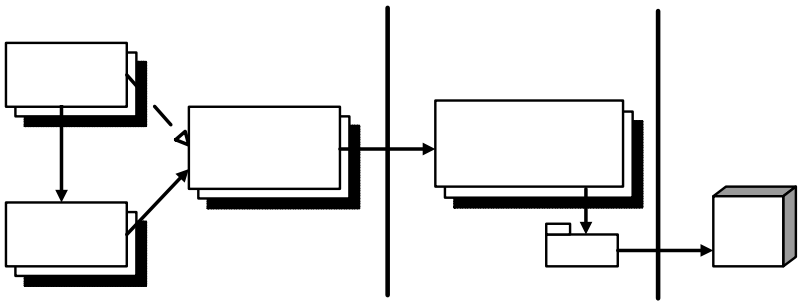

Figura 4.4 – Arquitectura do módulo de email

Devido ao facto relatado o trabalho consistiu na integração desta parte da aplicação com a lista de contactos do serviço myAgenda. Nomeadamente, neste módulo, a integração é observável quando o utilizador compõe uma mensagem, onde pode seleccionar para os campos To, Cc e Bcc, um dos seus contactos ou grupo de contactos existentes no seu livro de contactos; o mesmo acontece quando se está a visualizar uma mensagem e se pretende reencaminhar essa mensagem para alguém. Isto foi conseguido chamando funções pertencentes a alguns *beans* do módulo myAgenda que devolvem os contactos e os grupos de contactos do utilizador actual e fazendo o tratamento necessário aos dados (contactos) para que estes ficassem mais intuitivos para o utilizador.

4.1.1.2 Instant Messaging

Esta parte do módulo myCom utilizava a biblioteca Jabberbeans [22] para comunicação com o servidor Jabberd [23], biblioteca essa que entretanto foi descontinuada, devido a esse facto

e ao facto de que essa biblioteca era de difícil compreensão e utilização, dado que o programador não se pode abstrair do protocolo de comunicação entre a aplicação e o servidor, optou-se pela substituição desta biblioteca por uma mais recente. A escolha recaiu então na biblioteca Smack [24] que permite abstrair então do protocolo de comunicação com o servidor Jabberd [23]. Devido ao facto de se ter que actualizar a biblioteca de comunicação utilizada, esta parte da aplicação foi então desenvolvida de raiz, aproveitando então para o desenvolvimento ainda de novas funcionalidades que não se encontravam presentes na versão anterior, como sejam a adição, remoção ou alteração dos contactos na lista de amigos.

A arquitectura desta parte da aplicação é a que se mostra na figura seguinte.

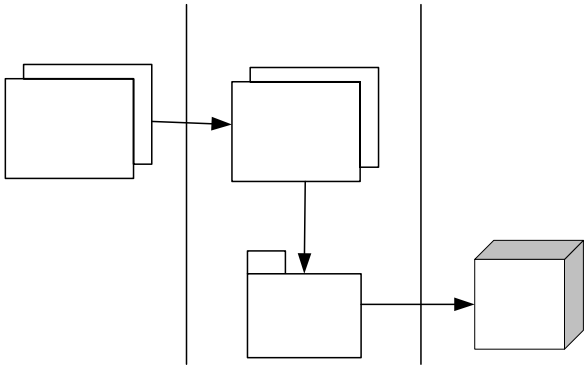

Figura 4.5 – Arquitectura do modulo de instant messaging

Este serviço é um serviço com características assíncronas tendo por base notificações assíncronas, notificações essas que têm de ser mostradas ao utilizador. 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 servidor 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. Daqui advinha o maior problema no desenvolvimento deste serviço. Como manter a informação actualizada para o cliente sem ser necessária qualquer acção da parte deste. Tomou-se então a opção da utilização de um suporte de comunicações ponto a ponto (*peer-to-peer*). 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.

Web Container EJBs C

Struts Actions Sessi

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 um *applet*. Foi ainda necessário do lado do servidor algum mecanismo que notificasse então o *applet* aquando da recepção de alguma notificação. Isto foi conseguido recorrendo à possibilidade de utilização de *listeners* oferecida pela biblioteca Smack [24]. *Listener* esse que é executado sempre que o

S

servidor recebe uma notificação de uma nova mensagem ou da mudança de estado de presença de algum contacto da lista de amigos do utilizador, e ao ser executado comunica com o *applet* no cliente para que o próprio *applet* possa fazer um pedido de actualização da página.

Nesta parte da aplicação realizou-se ainda uma integração mais profunda com o resto do sistema possibilitando que se possa adicionar um contacto à lista de *buddies* que já se encontre na lista de contactos do módulo myAgenda e ainda a possibilidade de adicionar um contacto à lista de *buddies* e ao mesmo tempo adicioná-lo também à lista de contactos do módulo myAgenda.

4.1.2 Módulo myAgenda

Na página inicial deste módulo efectuaram-se algumas alterações, entre elas a possibilidade do utilizador visualizar os eventos que tem agendados para o dia actual, não sendo necessário assim aceder à página do calendário. Nesta página ainda são disponibilizadas as funcionalidades provavelmente mais utilizadas, como seja criar contactos, listar contactos, etc.

De seguida é descrito o trabalho realizado para cada um dos componentes deste módulo.

4.1.2.1 Calendário

Nesta parte da aplicação apenas foram desenvolvidas novas funcionalidades que se descrevem de seguida.

Ao nível da criação de eventos implementou-se a funcionalidade de adicionar participantes a um evento, participantes esses que podem ser contactos ou grupos de contactos existentes no livro de contactos. Sendo estes participantes notificados por *email* (caso o possuam) da criação/alteração/cancelamento de um qualquer evento do qual são participantes. Caso um participante adicionado aquando da criação de um evento também seja assinante do serviço myAgenda, o evento criado também aparecerá no seu calendário podendo este visualizá-lo. Ainda na criação de eventos foi possibilitada a funcionalidade de definir repetições dos eventos, repetições essas que podem ser de 4 tipos:

- Diária – repete-se todos os dias, à mesma hora;
- Semanal – repete-se todas as semanas, no mesmo dia da semana;
- Mensal – repete-se todos os meses, no mesmo dia do mês;
- Anual – repete-se todos os anos, no mesmo mês e no mesmo dia.

Existe ainda a possibilidade de definir a data de término para a repetição.

Ao nível das vistas, na vista mensal foi optimizado o algoritmo que verificava se um dia possuía eventos, dado que o algoritmo anterior acedia à lista de todos os eventos desse dia, e se houvesse eventos então marcava a célula de maneira diferente; assim optou-se por implementar uma nova função que procurava se o dia possuía algum evitando assim a leitura de todas as entradas na base de dados para esse dia. A vista semanal era estática não apresentando nenhuma informação contida na base de dados para essa semana, teve-se então de proceder à implementação de uma funcionalidade que permitisse a visualização dos vários eventos decorrentes ao longo dos dias da semana.

O modelo de domínio do calendário já existente possibilitava todas estas novas funcionalidades, daí que este se manteve inalterado, modelo esse que é apresentado na figura seguinte:

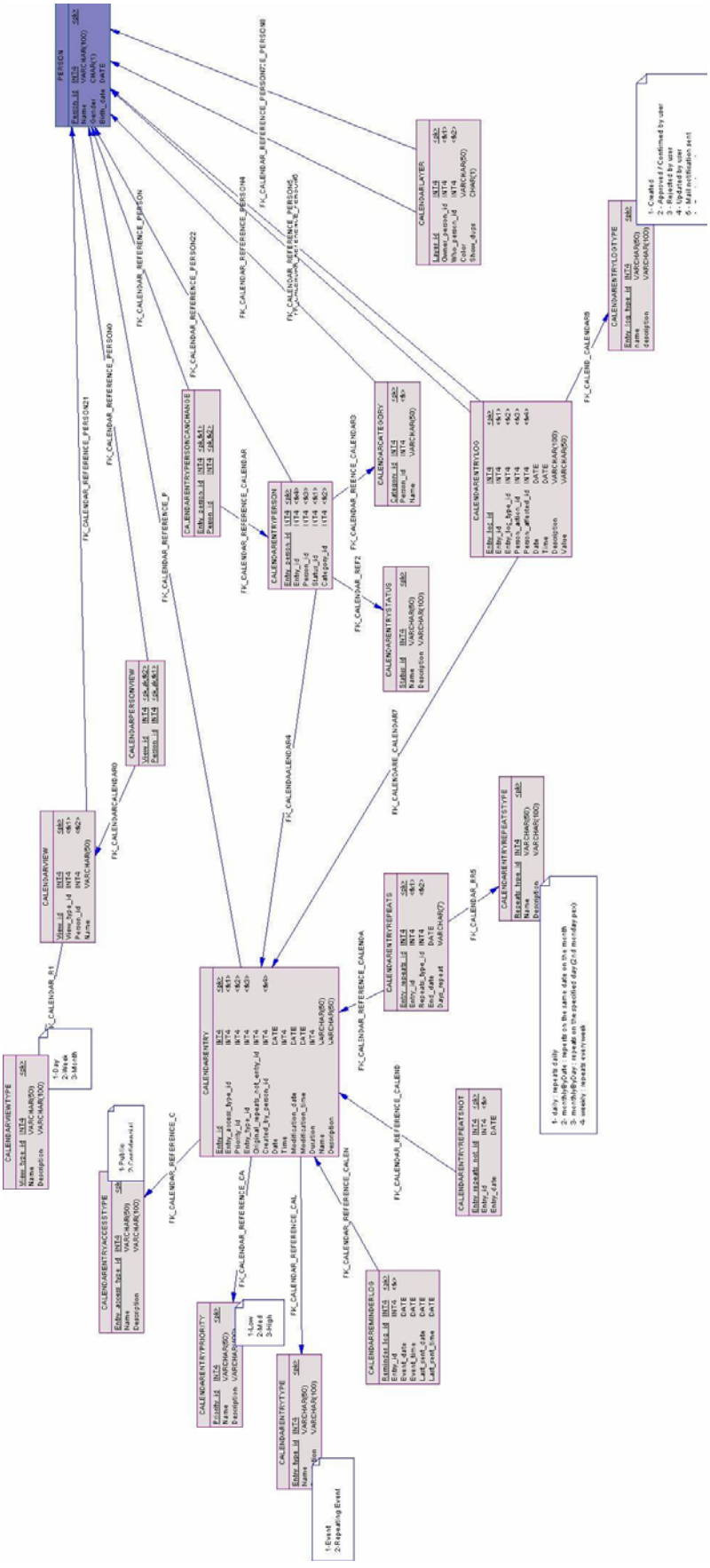

Figura 4.6 – Modelo de domínio do módulo de Calendário

4.1.2.2 Contactos

Nesta parte da aplicação começou-se por alterar o modo de representação dos contactos. Inicialmente um contacto só podia ter um tipo de telefone, um tipo de endereço electrónico e um tipo de morada. Se se quisesse ter um número de telefone e um número de telemóvel para uma mesma pessoa isso implicava ter dois contactos diferentes referentes à mesma pessoa, o mesmo acontecendo para diferentes tipos de endereços electrónicos e diferentes tipos de moradas, sendo que cada contacto individual possuía sempre um endereço electrónico, uma morada e um numero de telefone. Assim a alteração por nós efectuada consistiu em possibilitar que o mesmo contacto tivesse informação acerca de dois tipos de endereço electrónico (*email* e IM), três tipos de telefone (móvel, de casa e do trabalho) e dois tipos de morada (de casa e do trabalho), como acontece na maioria das aplicações que disponibilizam serviços deste género.

Implementaram-se ainda duas novas funcionalidades, casos de importar e exportar contactos em formato CSV (*Comma Separated Values*). Estas funcionalidades foram implementadas visto que a existência da funcionalidade de exportar para VCard só possibilitar a exportação de um contacto de cada vez. Estas novas funcionalidades facilitam assim a manutenção da lista de contactos entre o sistema e outras aplicações, como seja o Microsoft Outlook.

O modelo de domínio inicial dos contactos também possibilitava todas estas alterações, pelo que se manteve inalterado, modelo apresentado na figura seguinte:

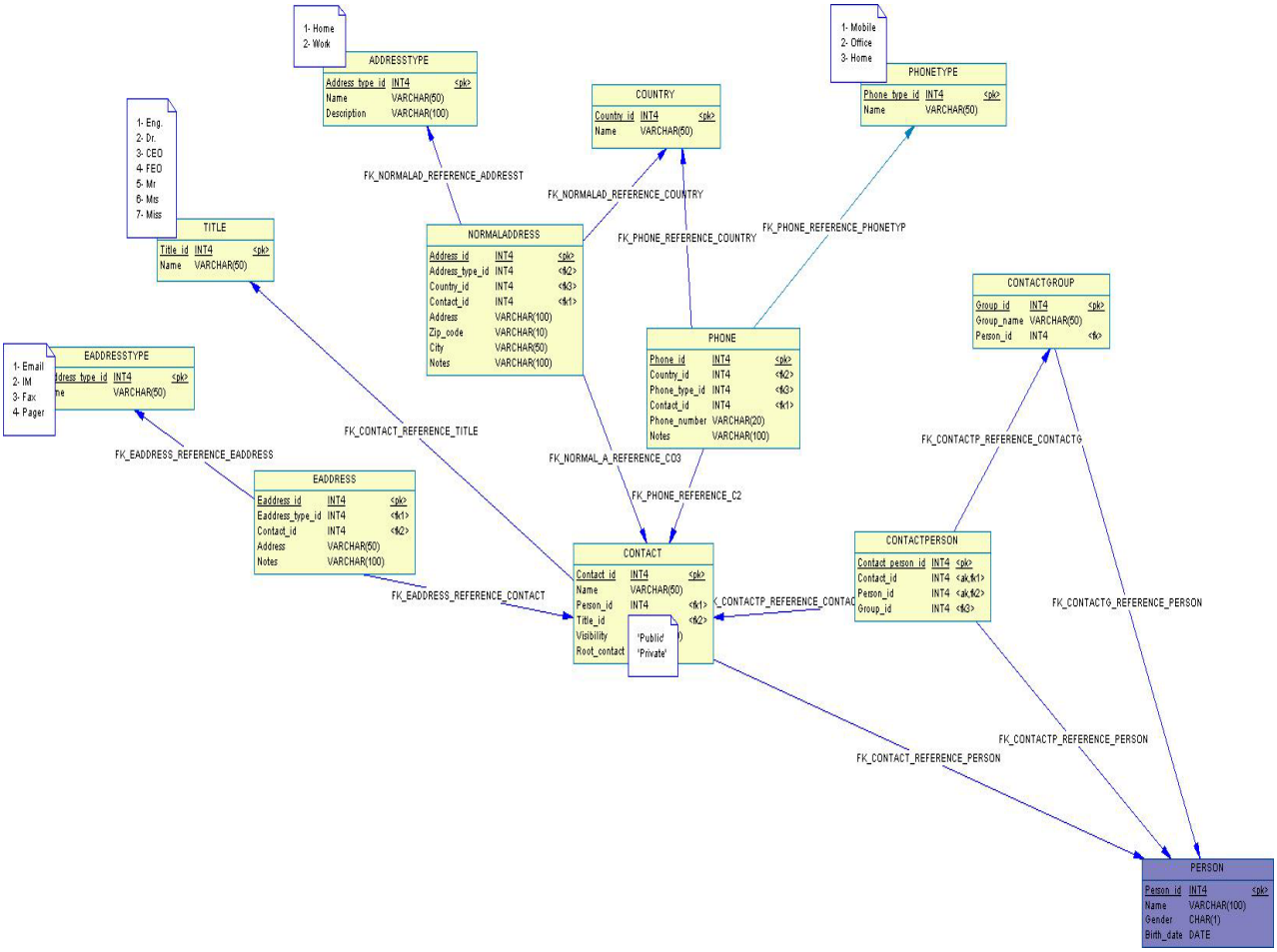

Figura 4.7 – Modelo de domínio do módulo de Contactos

4.1.2.3 Notas

Todos os casos de utilização desta parte da aplicação foram implementados no âmbito deste trabalho visto ainda não se encontram presentes no sistema, casos já descritos em cima. Para implementar estas funcionalidades foi desenvolvido tudo desde a camada de negócio à camada de apresentação.

Quanto ao modelo de domínio inicial da parte das notas, foi também por nós alterado para permitir a organização das notas por pastas, pastas essas que podem ainda conter outras pastas, tal como num sistema de ficheiros. Sendo assim o modelo de domínio final da parte das notas é o que mostra na figura seguinte.

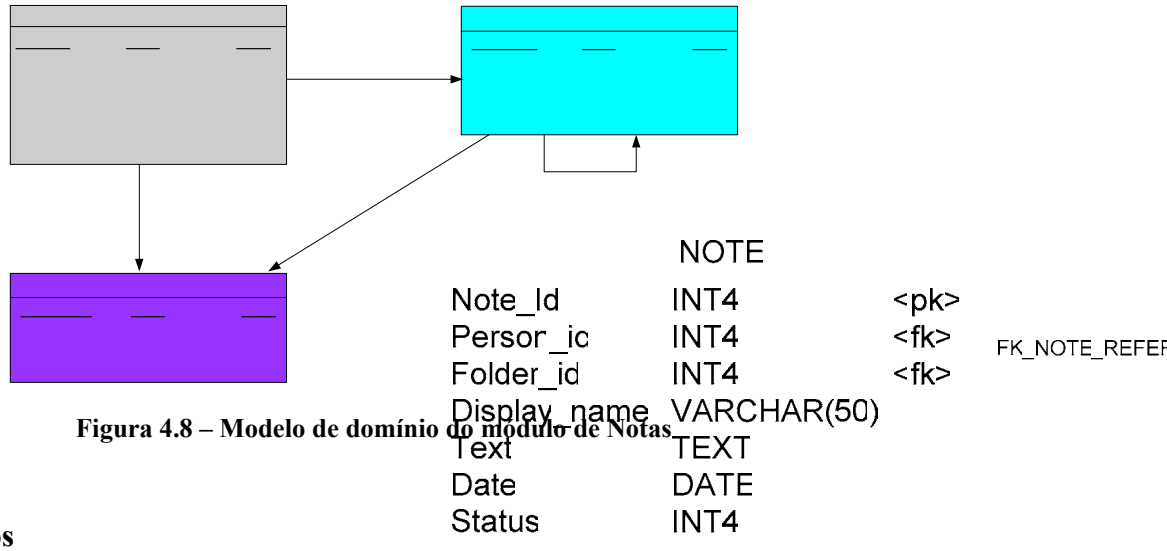

Figura 4.8 – Modelo de domínio do módulo de Notas

4.1.2.4 Favoritos

Todos os casos de utilização desta parte da aplicação foram também implementados no âmbito deste trabalho visto ainda não se encontram presentes no sistema, casos já descritos em cima. Para implementar estas funcionalidades foi desenvolvido tudo desde a camada de negócio à camada de apresentação.

Quanto ao modelo de domínio inicial da parte dos favoritos, foi também por nós alterado para permitir a organização dos favoritos por pastas, pastas essas que podem ainda conter outras pastas, tal como num sistema de ficheiros. Sendo assim, o modelo de domínio final da parte dos favoritos é o que mostra na figura seguinte.

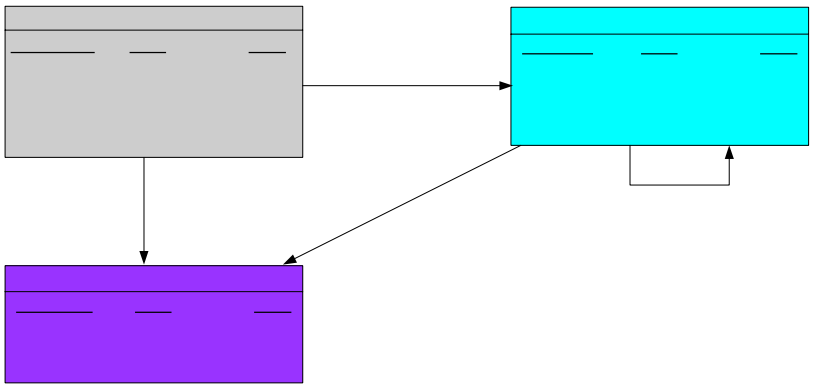

Figura 4.9 – Modelo de domínio do módulo de Favoritos

4.1.3 Módulo de administração

Neste módulo a única funcionalidade desenvolvida foi permitir ao administrador consultar os dados de uma subscrição, de modo a conhecer os dados do utilizador que pretende efectuar essa mesma subscrição.

Foram ainda alteradas as vistas retornadas aquando de uma acção de validação ou invalidação ou suspensão, para que se voltasse à vista anterior à acção, já que inicialmente voltava sempre para a vista das subscrições do dia actual.

4.1.4 Módulo das interfaces móveis

4.1.4.1 Sincronização

Neste módulo foram reutilizadas as acções já existentes, sendo apenas necessárias ligeiras alterações pontuais, sendo retornadas vistas diferentes das retornadas para a interface web. Estas vistas consistem em documentos XML [12], que transportam informação para a aplicação móvel. Todas as opções tomadas e o protocolo de sincronização encontra-se descrito mais abaixo no ponto 4.2.3..

Para esta opção de sincronização foi ainda necessário alterar o modelo de domínio, uma vez que passou a ser necessário saber qual o estado das diferentes entidades presentes na base de dados. Estados esses que passam por: novo, actualizado, apagado ou modificado. Esta alteração consistiu em acrescentar uma coluna que representasse o estado, às tabelas onde essa informação era necessária, sendo elas as tabelas de notas, de relação entre contacto e pessoa e de relação entre tarefas e pessoa. De modo a manter o valor do estado coerente com as acções realizadas pelo utilizador foram efectuadas algumas alterações, tanto na camada de negócio como na camada de apresentação.

4.2 Cliente PUC para telemóvel

Pretendeu-se desde o início do projecto adaptar da melhor forma possível a comunicação entre este módulo e o servidor central do sistema, bem como uniformizar as representações dos dados em ambos os módulos.

Uma das questões com que se deparou primeiro foi qual a representação dos dados a escolher uma vez que ambos os módulos tinham maneiras diferentes de os representar. Optou-se então por alterar neste módulo a forma de representar os dados de modo a ficarem coerentes com a base de dados central do sistema. Deste modo alterou-se a forma de representação dos contactos, das notas e das tarefas. Devido a esta alteração de representação de dados, tiveram de ser refeitas todas as funcionalidades deste módulo, sendo que a funcionalidade de *email* foi desenvolvida de base uma vez que ainda não encontrava disponível na aplicação.

Por fim existiu um redesenho total do mecanismo de sincronização, uma vez que o servidor central do sistema não possuía suporte algum para esta operação e devido também ao facto de o modelo de dados ter sido todo rescrito.

Depois de tomadas estas decisões, passou-se à fase de desenvolvimento.

Foi ainda proposto um possível redesenho dos menus da aplicação de modo a torná-los mais intuitivos de utilizar, mas uma vez que a tecnologia já possui a forma dos menus predefinida, não foi possível encontrar uma melhor forma de os estruturar.

4.2.1 Menus

A aplicação divide-se em quatro conjuntos de funcionalidades, myCom, myAgenda, opções da aplicação e informação do utilizador.

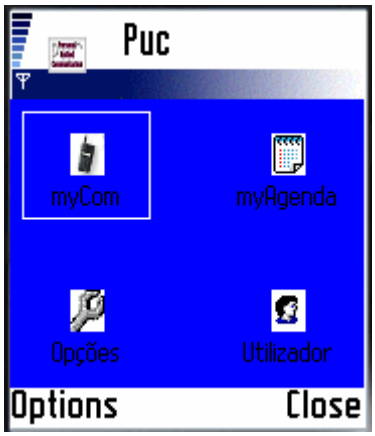

Figura 4.10 – Menu principal da aplicação

Os menus desta aplicação estão estruturados da seguinte maneira:

- myCom:
  - Email – criação, envio e recepção de mensagens de email.
- myAgenda:
  - Calendário – vista diária, semanal e mensal do calendário, bem como criação e listagem de tarefas e sincronização com o servidor.
  - Notas – gestão de notas e sincronização.
  - Contactos – gestão de contactos e grupos de contactos e sua sincronização.
- Opções:
  - Definir modo de calendário – definir a vista a usar no calendário.
  - Definir cor de fundo – definir a cor de fundo da aplicação.
  - Definir cor de texto – definir a cor de texto da aplicação.
  - Definir cor do rectângulo – definir a cor do rectângulo de selecção.

- Utilizador:
    - Definir perfil – definição dos dados pessoais do utilizador.
    - Definir ligação – definição dos parâmetros de ligação ao sistema central.

4.2.2 Representação de dados

De seguida são descritas as alterações que foram efectuadas relativamente à representação dos dados.

4.2.2.1 Contactos

Os contactos como não se encontravam de acordo com o modelo de dados definido para o sistema central, a sua representação teve de ser completamente alterada. Assim os contactos passaram a ser representados pelos seguintes campos:

- Id – identificador do contacto (interno do sistema);
- ContactPersonId – identificador da relação entre contacto e pessoa (interno do sistema);
- OwnerId – identificador da pessoa a que pertence o contacto (interno do sistema);
- EaddId – identificador do endereço electrónico associado ao contacto (interno do sistema);
- ImaddId – identificador do endereço de *instant messaging* associado ao contacto (interno do sistema);
- PhoneId – identificador do número de telefone pessoal associado ao contacto (interno do sistema);
- MphoneId – identificador do número de telemóvel associado ao contacto (interno do sistema);
- BphoneId – identificador do número de telefone do emprego associado ao contacto (interno do sistema);
- Name – nome do contacto;
- Group – nome do grupo a que o contacto pertence;
- Email – endereço electrónico associado ao contacto;
- Im – endereço de *instant messaging* associado ao contacto;

- Phone – número de telefone pessoal associado ao contacto;
- Mphone – número de telemóvel associado ao contacto;
- Bphone – número de telefone do emprego associado ao contacto;
- External – indicação se o contacto é externo ao sistema (interno do sistema);
- State – estado de sincronização do contacto (interno do sistema).

De notar que a representação anterior dos contactos não tinha qualquer semelhança com esta, uma vez que a noção de contacto do servidor central era completamente diferente à implementada na aplicação.

De notar ainda que a representação dos grupos de contactos não foi alterada, visto já se encontrar compatível com a representação presente no servidor central.

4.2.2.2 Notas

Quanto à representação das notas a única alteração que se efectuou foi a adição de um novo campo que contém o texto de descrição da nota, uma vez que até aí as notas só possuíam o seu título e nada mais.

De modo a não tornar esta aplicação mais confusa quanto ao uso, optou-se por não utilizar pastas para agrupar as notas como acontece no servidor central, sendo as notas apresentadas todas juntas numa mesma lista.

4.2.2.3 Tarefas

Quanto às tarefas, de modo a que ficassem compatíveis com a representação do servidor central bastou apenas modificar a representação da duração da tarefa, uma vez que anteriormente era representada pela hora de fim dessa tarefa e alterou-se para a duração efectiva da tarefa. Inicialmente a tarefa possuía também informação acerca do seu tipo (de entre 3 valores possíveis), campo que foi modificado para conter o título da tarefa de modo a poder ter uma descrição mais específica do tipo de tarefa, como é feito na representação das tarefas no sistema central.

4.2.3 Sincronização

Depois de efectuadas as alterações descritas em cima, iniciou-se o desenvolvimento da actualização dos dados no servidor a partir dos existentes no telemóvel e vice-versa. De entre os modos de comunicação para esta operação surgiram as possibilidades da utilização de

webservices [15] ou então da utilização da criação manual de ligações HTTP sendo os dados transmitidos usando o formato XML [12]. Optou-se pela criação manual de ligações HTTP (interface *javax.microedition.io.HttpConnection*) com o servidor em que os dados são enviados em comandos POST para as mesmas *actions* utilizadas na interface web de maneira a efectuar a reutilização do código já feito na parte do servidor e minimizar ainda a dificuldade de manutenção da aplicação. Para além da vantagem de reutilização de código existe ainda a vantagem de se reduzir a quantidade de informação transmitida diminuindo assim significativamente o tempo e os custos desta operação, caso se optasse pela opção da utilização de *webservices* [15].

Suponhamos que se adicionava uma nova nota na interface web e de seguida se efectuava a sincronização das notas. Caso se tivesse optado pela utilização de *webservices* [15] a informação transmitida, segundo a especificação do protocolo SOAP [13], para além do cabeçalho presente em todas as ligações HTTP, seria a seguinte:

Pedido realizado pelo telemóvel:

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:GetNewNotes xmlns:m="Some-URI">
       </m:GetNewNotes>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Informação enviada pelo servidor:

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
       <m:GetNewNotesResponse xmlns:m="Some-URI">
           <id>23</id>
           <name>Nota</name>
           <text>Nota de teste</text>
       </m:GetNewNotesResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Tabela 4.1 – Informação transmitida utilizando webservices

Com a opção tomada, o pedido feito pelo telemóvel é um simples pedido HTTP e a informação enviada pelo servidor, para além do cabeçalho de qualquer ligação HTTP, é:

```
<note>
      <id>23</id>
      <name>Nota</name>
      <text>Nota de teste</text>
</note>
```

Tabela 4.2 – Informação transmitida utilizando ligações HTTP

Como é possível ver através deste exemplo a quantidade de informação transmitida através de *webservices* [15] é muito superior à quantidade necessária para a opção implementada. Neste caso a quantidade de informação transmitida por *webservices* [15] ronda os 556 caracteres enquanto que na opção implementada ronda os 67 caracteres, cerca de oito vezes menos informação transmitida, e consequentemente gastos oito vezes inferiores comparando com o possível uso de *webservices* [15].

Para o servidor, esta operação de sincronização não é diferente de ter vários acessos feitos sequencialmente por um utilizador Web, uma vez que são utilizadas as mesmas *actions* e EJBs sendo apenas diferentes as vistas que são transmitidas ao utilizador. No sentido inverso (do servidor para o telemóvel) a informação é transferida em ficheiros XML [12], que uma vez que chegam ao telemóvel são interpretados através de um *parser*, neste caso o kXML, e guardados na base de dados.

Decidiu-se ainda separar a sincronização dos diferentes tipos de dados de modo a que o utilizador opte por actualizar apenas a informação que ache importante num dando momento, diminuindo assim o tempo e custo da operação.

Outra opção por nós tomada é o facto de, em caso de conflito, os dados presentes no telemóvel se sobreporem sempre aos dados presentes no sistema central. Tomou-se esta opção devido ao facto de o telemóvel estar sempre próximo do utilizador e pressupõe-se assim que a informação aí presente é a mais correcta.

De seguida é mostrado o diagrama de sequência do processo de sincronização das notas, em que as operações são efectuadas pela mesma ordem para qualquer tipo de informação a sincronizar.

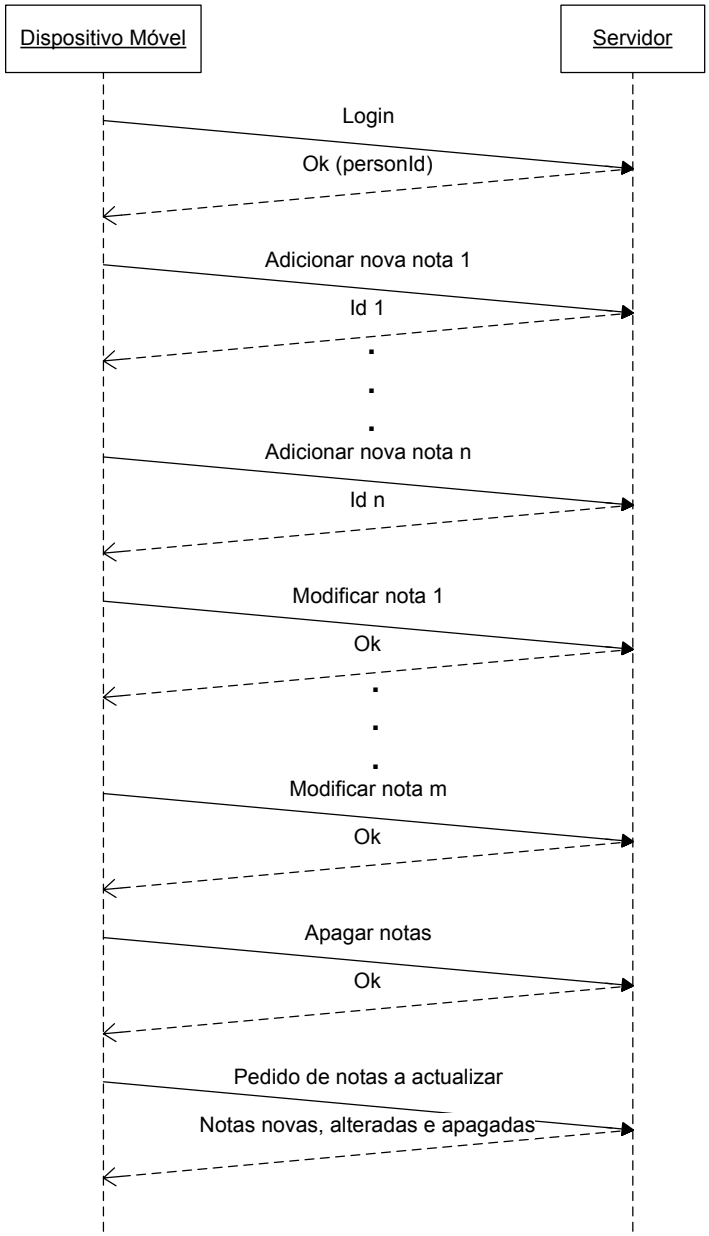

Figura 4.11 – Diagrama de sequência para a sincronização de notas

5 Conclusões

Este projecto foi bastante interessante, principalmente porque permitiu adquirir conhecimentos acerca de várias tecnologias utilizadas nas várias fases de desenvolvimento de um projecto e que poderão ser úteis no futuro ao longo da vida profissional. Ao longo do desenvolvimento deste projecto também adquirimos experiência em nos habituarmos a cumprir prazos, resolver situações imprevistas, planear e dividir bem as nossas tarefas, entre outras coisas.

Deparámo-nos também com os problemas normais que ocorrem no desenvolvimento de um projecto deste tamanho e que ainda não nos tinham acontecido durante o curso. Referimo-nos a cumprimento de prazos e questões mais técnicas como manutenção do código, gestão de versões, etc.. Outra das dificuldades foi trabalhar com código desenvolvido por terceiros, e sua consequente compreensão.

Como já referido, este trabalho foi muito enriquecedor porque para além de ter a componente académica tem também uma componente comercial importante, o que sensibilizou também para outras questões como o custo de utilização, facilidade de utilização, valor do produto percebido pelo consumidor (o que pode levar o consumidor a optar pelo produto em detrimento de outros), etc. Este projecto levou-nos a perceber que para os clientes o importante é a interface, não interessa a estruturação do sistema nem se é ou não facilmente extensível, para o utilizador o que interessa é ter uma interface agradável, intuitiva e funcional.

De notar que qualquer implementação sobre a aplicação é bastante fácil de efectuar, sendo a própria aplicação acessível em termos de aprendizagem, devido aos vários padrões arquitecturais utilizados e flexibilidade estrutural existente em toda a aplicação.

5.1 Trabalho Futuro

Existem alguns aspectos funcionais que poderiam tornar o sistema mais completo para o utilizador. Estes aspectos são sobretudo novas funcionalidades que são listadas em seguida.

5.1.1 Módulo myCom

- Permitir a ordenação por online/offline da lista de contactos do serviço de mensagens instantâneas para além da ordenação por grupo já implementada;
- Permitir guardar uma mensagem de *email* (na pasta "Draft") caso não se pretenda enviá-la no momento;

- Permitir enviar mensagens de *email* com anexos;
- Actualizar a versão do JWMA [17] utilizada no serviço de *email* para uma versão mais recente, realizando uma actualização da arquitecta para a utilizada no resto do sistema;
- Gravação de mensagens para resposta automática (a definir nas configurações);
- Permitir acesso a vários servidores de email.

5.1.2 Módulo myAgenda

- Implementar a funcionalidade de procura nas várias secções do sistema como sejam os contactos, o calendário, as notas e os favoritos;
- Criar uma nova visualização dos grupos de contactos usando uma vista baseada em grupos (como acontece actualmente com as notas e favoritos na interface web) e sendo depois o utilizador a decidir qual das vistas pretende utilizar;
- Permitir que um grupo de contactos possa pertencer a um outro grupo e permitir ainda que um contacto possa pertencer a mais que um grupo;
- Criar a noção de grupos em que utilizadores do mesmo grupo possam marcar/alterar/apagar eventos no calendário de outro utilizador desse grupo;
- Criação da funcionalidade de definir alarmes para os eventos de modo a que o utilizador possa ser notificado aquando da ocorrência desses mesmos eventos;
- Criar a opção de apagar automaticamente os eventos passados, apagando eventos passado um dia, uma semana, um mês, um ano, ou então nunca os apagar.

5.1.3 Módulo de dispositivos móveis

- Criar a funcionalidade "Ir Para Data" na aplicação para dispositivos móveis de modo a facilitar a navegação do utilizador no calendário desta aplicação;
- Estudar outras possibilidades para o mecanismo de sincronização de modo a tornar este mecanismo compatível com uma gama mais vasta de dispositivos (tal como o SyncML).

6 Referências

[1] Java 2 Platform, Enterprise Edition (J2EE) – http://java.sun.com/j2ee/

[2] Java 2 Platform, Micro Edition (J2ME) – http://java.sun.com/j2me/

[3] JBoss – http://www.jboss.org/

[4] XDoclet: Attribute-Oriented Programming – http://xdoclet.sourceforge.net/

[5] Apache Ant – http://ant.apache.org/

[6] Jabber Protocol – http://www.jabber.org/

[7] PostgreSQL – http://www.postgresql.org/

[8] Java 2 Platform, Standard Edition (J2SE) – http://java.sun.com/j2se/

[9] Enterprise JavaBeans Technology – http://java.sun.com/products/ejb/

[10] The Apache Struts Web Application Framework – http://struts.apache.org/

[11] Tiles – http://struts.apache.org/userGuide/dev_tiles.html

[12] Extensible Markup Language (XML) – http://www.w3.org/XML/

[13] Simple Object Access Protocol (SOAP) – http://www.w3.org/TR/soap/

[14] Web Services Description Language (WSDL) – http://www.w3.org/TR/wsdl

[15] Web Services – http://developers.sun.com/techtopics/mobility/apis/articles/wsa/

[16] Extensible Messaging and Presence Protocol – http://www.jabber.org/

[17] JWMA – http://jwma.sourceforge.net/

[18] Alcatel e-Communication Center – http://www.alcatel.com/

[19] Chandler – http://www.osafoundation.org/

[20] Elefante – http://www.elefante.com.br/

[21] Weblicon Technologies – http://www.weblicon.net/

[22] Jabberbeans – http://jabberbeans.jabberstudio.org/

[23] Jabberd – http://jabberd.jabberstudio.org/

[24] Jive Software XMPP - Smack API – http://www.jivesoftware.com/xmpp/smack/

[25] Alberto Rodrigues Silva, João Patriarca, João Clemente, Paulo Chainho, Paulo Ferreira, *PUC – Sistema de Comunicações Pessoais para as Redes de Próxima Geração*, CITA'2003.

[26] Marco Guimarães, Rui Maia, *Desenvolvimento do serviço myAgenda*, TFC 2002/2003, Fevereiro 2004.

[27] Nuno Domingos, Nuno César, *Desenvolvimento das interfaces para terminais móveis*, TFC 2002/2003, Outubro 2003.

[28] João Pedro Patriarca, *Desenvolvimento do serviço myCom*, MEEC 2001/2002.