FEUP Universidade do Porto
Faculdade de Engenharia

3 ao Quadrado - Agenda Web

Relatório de Projecto

Gestão de Projectos de Software - Grupo A - LEIC 2001/2002
http://gnomo.fe.up.pt/gps01A

João Montenegro - ei97023@fe.up.pt

André Teixeira - ei97024@fe.up.pt
César Rodrigues - ei97006@fe.up.pt
Nuno Dias - ei95030@fe.up.pt
Rui Soares - ei97026@fe.up.pt
Carlos Ribeiro - ei97043@fe.up.pt
Filipe Pinto - ei96036@fe.up.pt
Rúben Pereira - ei96033@fe.up.pt
Vânia Gonçalves - ei97011@fe.up.pt

Versão 2.0

14 de Dezembro de 2001

Conteúdo

Lista de Figuras

1 Introdução

1.1 Objectivos

Pretende-se com este relatório de projecto definir a arquitectura do sistema a implementar, bem como alguns aspectos que se prendem com a tecnologia a utilizar no desenvolvimento do sistema, e uma divisão em fases do trabalho de programação. Assim, há alguns aspectos a ter em conta, que se pretende que fiquem claros após a abordagem que se segue.

1.2 Âmbito do projecto

Como já foi dito no Relatório de Especificação de Requisitos, este projecto de reengenharia visa o desenvolvimento de uma aplicação (Agenda Web), feita tendo como ponto de partida uma outra já implementada.

1.3 Arquitectura

Para definir a arquitectura irá ser apresentado um conjunto de diagramas em Unified Modelling Language (UML). Assim, para se definir a arquitectura lógica vai ser utilizado um diagrama de pacotes; para a arquitectura física vão ser apresentados diagramas de componentes, modelo relacional da base de dados e diagrama de distribuição; para o modelo de objectos vai ser utilizado um diagrama de classes, e para o modelo dinâmico vão ser apresentados diagramas de sequência e de actividades. Com isto, pretende-se que sejam tornadas claras as opções feitas neste projecto de reengenharia.

1.4 Tecnologia

Quanto à tecnologia a utilizar na implementação (*JavaBeans* e *Java Server Pages* (JSP)), há também algumas questões que devem ser tidas em conta. O facto de ser uma tecnologia ainda em fase de estudo pelo grupo de trabalho, pode vir a influenciar o desenvolvimento do projecto, nomeadamente o tempo de programação.

2 Arquitectura Lógica

Na especificação da arquitectura lógica do sistema, pretende-se efectuar uma decomposição hierárquica do sistema em módulos lógicos, bem como a especificação das dependências entre esses módulos.

2.1 Diagrama de Pacotes

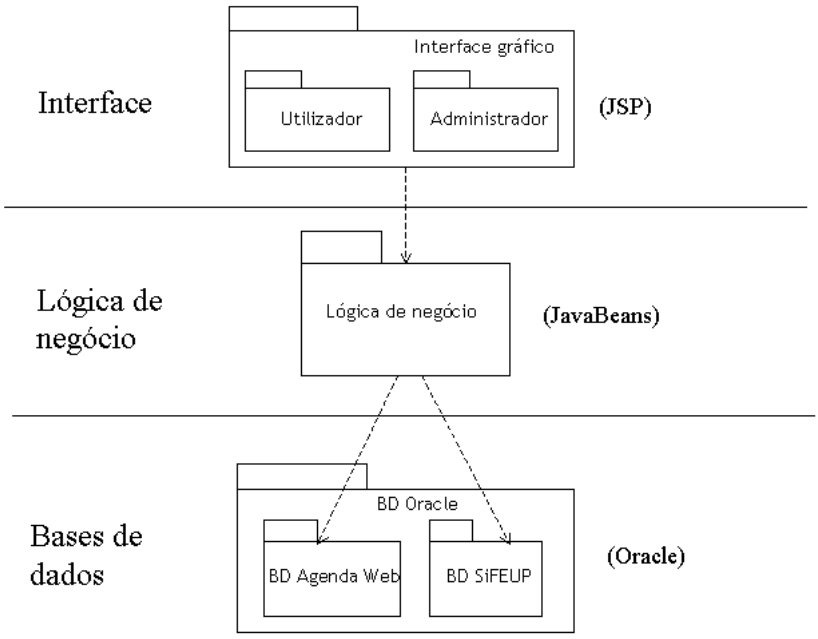

Figura 2.1: Diagrama de Pacotes.

2.2 Descrição do diagrama

2.2.1 Pacote Interface gráfico

Este pacote encontra-se na camada de mais alto nível e pretende representar o interface da aplicação com o utilizador. No referido pacote faz-se uma distinção entre o utilizador e o administrador, porque há diferenças entre as operações feitas pelo administrador e as operações feitas pelo utilizador. Estas diferenças estão, no essencial, descritas no modelo de casos de uso presente no relatório de especificação de requisitos (secção 4.) já publicado. Pretende-se englobar no pacote **Utilizador** os interfaces com o colaborador da organização e com o dono da agenda, isto porque, como as operações efectuadas por ambos são equivalentes, também os referidos interfaces serão semelhantes. Quanto à tecnologia, os interfaces serão implementados em HTML com recurso a JSP.

2.2.2 Pacote Lógica de negócio

Este pacote situa-se numa camada intermédia e pretende representar uma classe de objectos que servirão para criar uma ligação entre o interface e a camada de mais baixo nível. É utilizado nesta camada o modelo *JavaBeans*. Este é um modelo de componentes, independente da plataforma, portável e codificado em *java*. Os *beans* são componentes *JavaBeans* independentes. São também módulos de *software* reutilizáveis.

2.2.3 Pacote BD Oracle

Este pacote encontra-se na camada de mais baixo nível e pretende representar as bases de dados utilizadas pela aplicação. Assim, temos neste pacote as bases de dados *Oracle*. Existe a base de dados da aplicação,**BD Agenda Web**, onde serão guardados dados relevantes para o normal funcionamento da aplicação, como por exemplo, dados relacionados com os utilizadores, compromissos disponibilidades, etc.

Na BD *Oracle* encontra-se também o pacote **BD SiFEUP**, que representa o sistema de informação da Faculdade, ao qual será necessário ir buscar informação, nomeadamente sobre os horários das aulas e sobre as pessoas ligadas à organização.

3 Arquitectura Física

Neste capítulo pretende-se especificar a topologia de *hardware* (equipamentos e conexões) do sistema sobre a qual são executados os componentes de *software*. Para tal, vai ser apresentado um diagrama de distribuição. Pretende-se também neste capítulo especificar a estrutura física da implementação, ou seja ilustrar a decomposição do sistema em módulos físicos (tipicamente ficheiros), bem como a especificação de interfaces e dependências entre os módulos. Para tal, vai ser apresentado um diagrama de componentes.

3.1 Diagrama de Distribuição

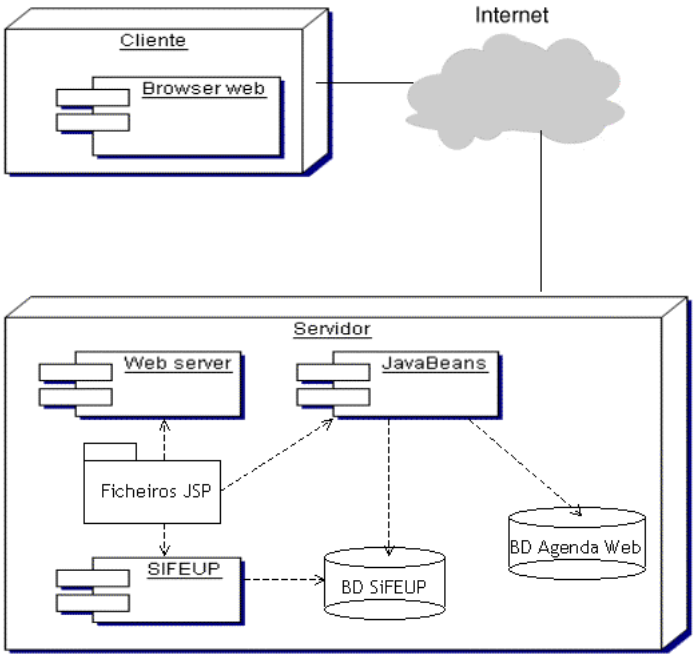

Figura 3.1: Diagrama de Distribuição.

3.1.1 Descrição do diagrama

Como se pode verificar, existem dois grandes nós presentes no diagrama, que são o nó **Cliente** e o nó **Servidor**. Estes nós estão ambos ligados a uma nuvem que representa a Internet. Quer isto dizer que as aplicações do nó **Cliente** acedem às aplicações do nó **Servidor** através da Internet.

Nó Cliente

O nó **Cliente** representa o computador do utilizador final, onde serão feitos pedidos a um servidor através da Internet. Estes pedidos são efectuados utilizando um **Browser Web**.

Browser Web O **Browser Web** representa uma aplicação que corre na máquina cliente, e serve como interface entre o utilizador e o sistema a desenvolver. O *browser* deve ter a capacidade de interpretar o código HTML que vai ser gerado pelo sistema.

Nó Servidor

O nó **Servidor** representa o computador onde correm as aplicações responsáveis por processar os pedidos efectuados pelas aplicações cliente.

JavaBeans Como já foi referido anteriormente, os *beans* são componentes *JavaBeans* independentes. São também módulos de *software* reutilizáveis. Um *bean* é constituído por eventos (acontecimentos e acções a executar), por propriedades (características do *bean*) e por métodos (procedimentos). Os *beans* devem conter os métodos responsáveis por tratar pedidos efectuados pela camada de interface por si só, ou com acesso à camada de bases de dados (devem conter métodos com capacidade para aceder à base de dados - BD SiFEUP).

Web Server O **Web Server** representa a aplicação servidora responsável por responder a pedidos efectuados via Internet por determinado cliente. Este servidor deverá suportar o uso de JSP, visto que terá de interpretar este tipo de páginas, daí a dependência existente entre o **Web Server** e as **Páginas JSP**.

Páginas JSP O pacote **Páginas JSP** representa os ficheiros responsáveis pela geração do código HTML que será visualizado pelo utilizador, quando este efectua algum pedido ao sistema. Estes ficheiros contêm códigoque será interpretado pelo Web Server. Em alguns casos serão utilizados serviços do **SiFEUP**, nomeadamente para autenticação de utilizadores e eventualmente para pesquisas de pessoas da organização, daí a dependênciaentre o pacote de **Páginas JSP** e as aplicações do **SiFEUP**.

SiFEUP O SiFEUP representa as aplicações do sistema de informação da faculdade. Este sistema tem métodos relevantes, nomeadamente para a autenticação e pesquisa de utilizadores. São responsáveis por chamar estes métodos as **Páginas JSP**, daí a ligação entre ambos presente no diagrama.

BD SiFEUP A base de dados *BD SiFEUP* é parte importante do sistema, visto que será responsável pelo armazenamento de quase toda a informação relativa às agendas. Por

exemplo, vai ser nesta base de dados que vão ser armazenados os compromissos criados pelos utilizadores. É também nesta base de dados que serão pesquisados dados relevantes para serem utilizados na aplicação, nomeadamente a informação relativa aos horários e pessoas da organização. É a esta base de dados que os métodos do **SiFEUP** e os **JavaBeans** vêm buscar alguns dados relevantes para a aplicação, daí as dependências existentes no diagrama.

BD Agenda Web Esta base de dados é responsável por guardar informação necessária para o normal funcionamento da aplicação. Serão guardadas informações relativas aos utilizadores da **Agenda Web**, compromissos, disponibilidades, tipos de compromissos, etc.

3.2 Diagrama de componentes - Organização física da aplicação

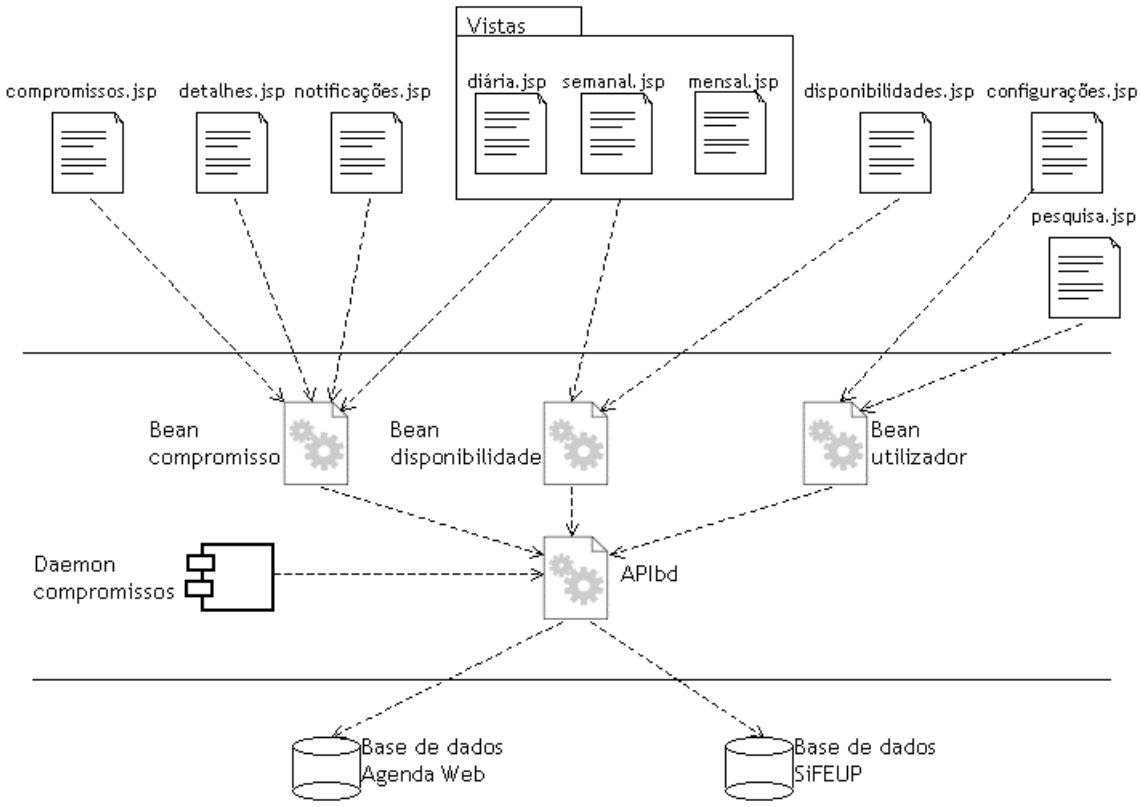

Figura 3.2: Diagrama de componentes da aplicação.

3.2.1 Descrição do diagrama

No diagrama anterior são apresentados os componentes (ficheiros) da aplicação, bem como as relações existentes entre eles.

Documentos JSP

Os ficheiros de código **JSP**, presentes na camada superior do diagrama, serão responsáveis pelo interface da aplicação. Ao ser interpretado, o código presente nestes ficheiros, irá gerar o código **HTML** que será apresentado ao utilizador.

Bean compromisso

O **Bean compromisso** contém métodos que têm como objectivo efectuar operações com os compromissos. Assim, estarão presentes neste bean, por exemplo, métodos para marcar, aceitar ou rejeitar um compromisso.

Bean disponibilidade

O **Bean disponibilidade** tem métodos para lidar com as disponibilidades. Será possível encontrar neste bean métodos para marcar ou visualizar disponibilidades em determinada agenda.

Bean utilizador

O **Bean utilizador** contém métodos que têm por objectivo efectuar operações sobre os utilizadores da **Agenda Web**. Neste bean será possível encontrar, por exemplo, métodos para pesquisar utilizadores da aplicação.

APIbd

A **APIbd** é uma biblioteca de métodos para acesso à base de dados da aplicação. Assim, todos os métodos presentes nos **(**beans), que necessitem de aceder à base de dados terão que fazê-lo através desta biblioteca.

Daemon compromissos

O **Daemon compromissos** é um programa que está sempre a correr numa determinada máquina, com o objectivo de enviar notificações aos utilizadores da **Agenda Web**. Pretende-se que com uma certa periodicidade, este programa consulte a base de dados, procurando as situações em que será necessário enviar uma notificação ao utilizador, e envie essa notificação.

Base de dados Agenda Web

A **Base de dados da Agenda Web** é responsável por guardar por exemplo, toda a informação relativa a compromissos e disponibilidades presentes nas agendas disponíveis.

Base de dados SiFEUP

Esta base de dados é a utilizada pelo SiFEUP, para guardar informação relacionada com os docentes e não docentes da Faculdade de Engenharia. É nesta base de dados que estão armazenados, por exemplo, os dados dos horários de alunos e professores.

3.3 Diagrama de componentes - base de dados Agenda Web

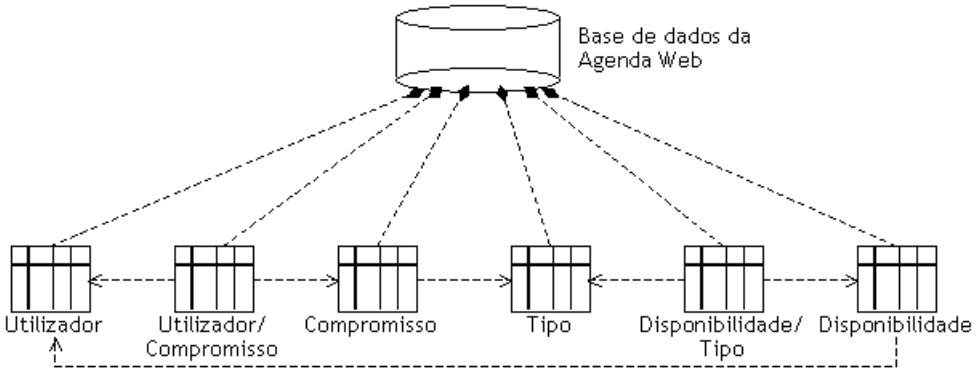

Figura 3.3: Diagrama de componentes da base de dados da **Agenda Web**.

3.3.1 Descrição do diagrama

No diagrama apresentado vê-se a constituição (tabelas e relações entre elas) da base de dados utilizada pela aplicação para guardar compromissos, disponibilidades e outra informação necessária para o bom funcionamento da **Agenda Web**.

Utilizador

Na tabela **Utilizador** é guardada a informação relativa aos utilizadores da **Agenda Web**. Trata-se de informação pessoal do utilizador e outra informação relacionada com o normal funcionamento da aplicação.

Compromisso

Na tabela **Compromisso** é armazenada a informação de cada compromisso marcado em determinada agenda. É guardada informação como, o momento de início, a duração, o local, aspectos que se prendem com a periodicidade do compromisso, etc.

Utilizador/Compromisso

Esta tabela serve como ligação entre as duas anteriores (**Utilizador** e **Compromisso**). Torna-se necessária a utilização desta tabela, porque existe uma ligação do tipo "Muitos-para-Muitos" entre as duas tabelas anteriores, ou seja, um **Utilizador** pode marcar muitos compromissos, e um **Compromisso** pode ter vários participantes.

Disponibilidade

Na tabela **Disponibilidade** é armazenada a informação de cada disponibilidade existente em determinada agenda. É guardada informação como, o momento de início, a duração, aspectos que se prendem com a periodicidade da disponibilidade, etc.

Tipo

Esta tabela guarda somente os tipos de compromisso ou disponibilidade existentes no sistema.

Disponibilidade/Tipo

Esta tabela serve para fazer uma ligação entre as tabelas **Disponibilidade** e **Tipo** (função semelhante a **Utilizador/Compromisso**). A necessidade da existência desta tabela advém do facto de existir uma relação do tipo "Muitos-para-Muitos" entre as tabelas já referidas, ou seja, uma **Disponibilidade** pode ser de vários tipos, e um **tipo** pode estar presente em várias disponibilidades.

3.4 Modelo relacional da base de dados Agenda Web

3.4.1 Descrição do modelo

Como se pode verificar após a inspecção do diagrama, existem quatro tabelas que guardam a informação mais básica utilizada no normal funcionamento da aplicação, que são: **Utilizadores**,

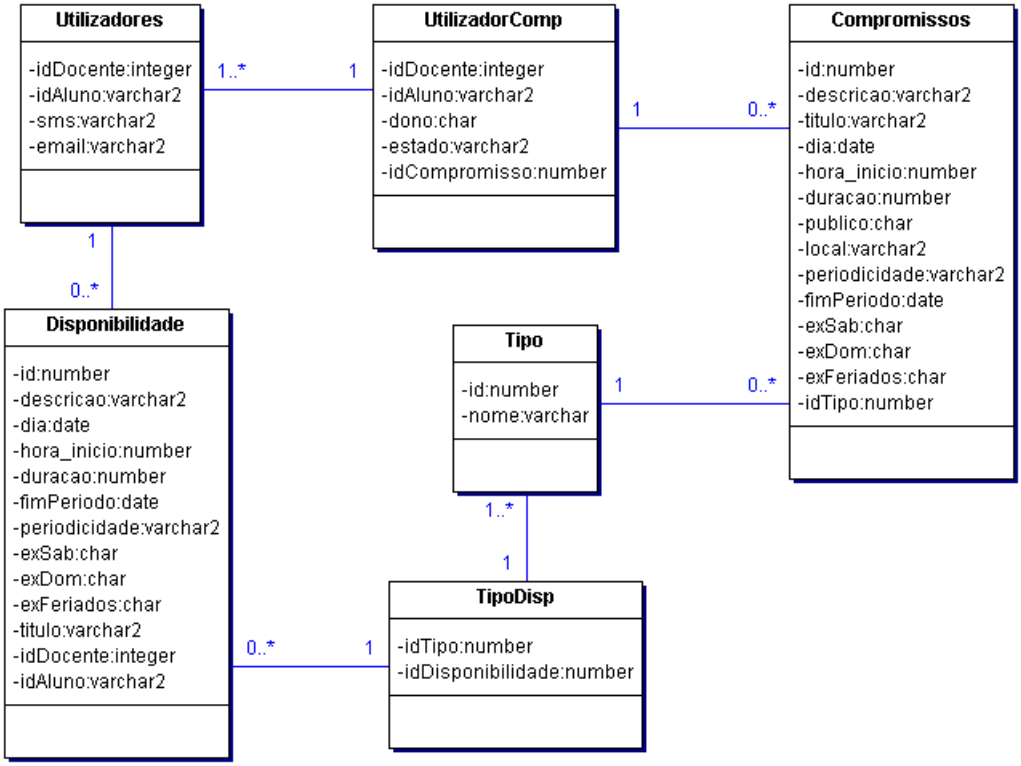

Figura 3.4: Modelo relacional da base de dados da **Agenda Web**.

Compromissos, **Disponibilidade** e **Tipo**. Existem depois, duas tabelas que têm o objectivo de implementar uma ligação entre outras tabelas. Estes são os casos onde existem relações do tipo "Muitos-para-Muitos", como se pode verificar no diagrama de classes (ver secção 5). As tabelas nestas condições são **UtilizadorComp** e **TipoDisp**. Através das relações entre as tabelas, presentes no modelo, pode-se verificar que um utilizador pode ter associadas 0 ou várias disponibilidades; um utilizador pode ter associados 0 ou vários compromissos, e um compromisso pode ter associados um ou mais utilizadores (esta ligação é feita através de informação presente na tabela **UtilizadorComp**); um tipo pode estar associado a 0 ou mais compromissos; um tipo pode estar associado a 0 ou mais disponibilidades e uma disponibilidade pode ser de 1 ou mais tipos.

Tabela Utilizadores

Na tabela **Utilizadores** existem dois campos (**idDocente** e **idAluno**) para representar o utilizador. Isto porque na base de dados do *SiFEUP* os alunos e docentes estão representados em tabelas diferentes e com tipos diferentes. Na prática, só um dos campos vai estar preenchido para cada utilizador, caso este seja aluno ou docente.Os campos **sms** e **email** contêm, respectivamente, o número de telefone para onde enviar a mensagem *sms* e o endereço de *e-mail* para onde enviar as mensagens de correio electrónico.

Tabela utilizadorComp

Na tabela **utilizadorComp**, além dos campos referentes às ligações com as tabelas **utilizadores** e **compromissos**, existem mais dois campos: **estado** e **dono**. O primeiro refere-se ao estado em que se encontra o compromisso para aquele utilizador (pendente, aceite, etc). O campo **dono** significa se quem marcou o compromisso foi este utilizador (a que se refere a linha, já que esta tabela relaciona utilizadores com compromissos).

Tabelas Compromissos e Disponibilidade

As tabelas **Compromissos** e **Disponibilidade**, em termos de campos, são idênticas, sendo a única diferença os campos que relacionam a tabela **Disponibilidade** com **Utilizadores** (**idAluno** e **idDocente**. Entre os vários campos existentes, salienta-se o conjunto de campos que indicam a data, hora e duração destes acontecimentos (**dia**, **hora_inicio**, **duracao**). O campo **periodicidade** é utilizado, caso o **compromisso/disponibilidade** seja periódico, para indicar a periodicidade com que este acontecimento se repete, até atingir uma data final especificada (**fimPeriodo**). Nesta periodicidade, podem existir excepções, nomeadamente ao sábado (**exSab**), domingo (**exDom**) e feriados (**exFeriados**).Contêm, também uma descrição (**descricao**), um título (**titulo**) e, claro, um identificador (**id**).

Tabela Tipo

A tabela **Tipo** contém apenas dois campos, sendo eles o **id** e o **nome**.

Tabela TipoDisp

A tabela **TipoDisp** contém os campos necessários para a relação entre as tabelas **Disponibilidade** e **Tipo**.

4 Modelo de Objectos

Nesta secção descreve-se os objectos que serão implementados para cada pacote lógico existente na arquitectura lógica.

4.1 Diagrama de classes

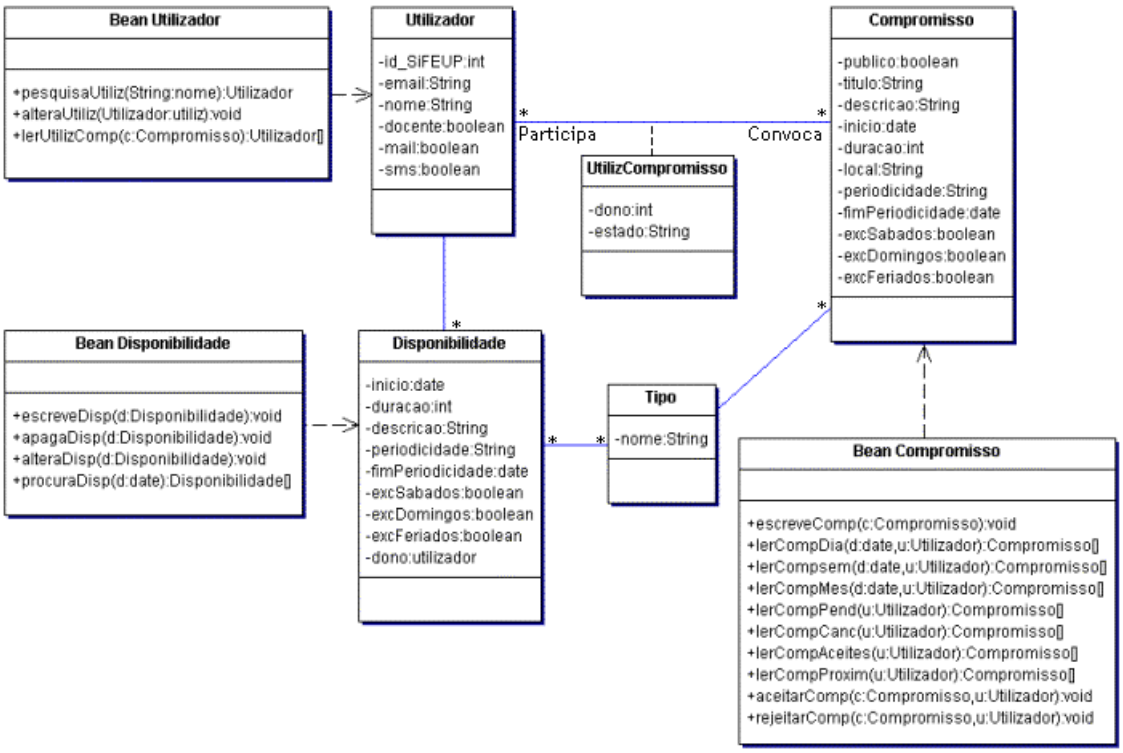

Figura 4.1: Diagrama de Classes.

4.1.1 Descrição do diagrama

Pode-se verificar no diagrama que existem três grandes classes que correspondem às três entidades principais presentes na aplicação. Associados a estas classes estão **JavaBeans** que contêm métodos para tratar acontecimentos relacionados com as classes referidas.

Classe Utilizador

Com esta classe pretende-se representar a informação que será necessário guardar acerca de cada utilizador da aplicação. A classe **Utilizador** está relacionada com a classe **Compromisso** por uma relação do tipo "Muitos para Muitos". Isto acontece porque na prática um utilizador poderá marcar vários compromissos, e em determinado compromisso poderão participar vários utilizadores. Existe também uma ligação do tipo "Um para Muitos" entre a classe **Utilizador** e **Disponibilidade**, visto que será possível a um determinado utilizador marcar várias disponibilidades na sua agenda.

Classe Compromisso

Pretende-se representar nesta classe a informação que será armazenada na base de dados da aplicação, acerca de cada compromisso marcado. Será possível, através dos atributos presentes, classificar o compromisso (público ou privado, ocasional ou periódico, etc.) e obter toda a restante informação que é relevante para a definição do mesmo.

Classe UtilizCompromisso

Esta é uma classe-associação onde se especifica atributos de um determinado par **Utilizador/Compromisso**. Esta classe associação permite determinar qual o utilizador que marcou determinado compromisso, e o estado actual do mesmo (pendente, recusado, etc.)

Bean Utilizador

Este **JavaBean** contém métodos que realizam operações relacionadas com o tratamento da informação referente ao utilizador. Os métodos que deverão estar presentes são:

- pesquisaUtiliz, que recebe como parâmetro o nome de um utilizador e retorna o objecto **Utilizador** com as informações relativas ao utilizador pesquisado;
- alteraUtiliz, que recebe como parâmetro um objecto **Utilizador** cujas informações deverão substituir as presentes no sistema de informação;
- lerUtilizComp, que recebe como parâmetro um objecto **Compromisso** e retorna todos os utilizadores que participam no mesmo.

Bean Compromisso

Este **JavaBean** contém métodos que realizam operações relacionadas com o tratamento da informação referente aos compromissos. Os métodos que deverão estar presentes são:

- escreveComp, que recebe como parâmetro um objecto **Compromisso**, registando a informação correspondente no sistema de informação;
- lerCompDia, recebe como parâmetros uma data e um objecto **Utilizador**. O objectivo deste método é retornar um conjunto de objectos **Compromisso** que contém informação sobre os compromissos de um utilizador da aplicação para determinado dia;
- lerCompSem, é um método semelhante ao anterior, com a diferença de retornar os compromissos existentes numa determinada semana;
- lerCompMes, é um método semelhante ao **lerCompDia**, com a diferença de retornar os compromissos existentes para determinado mês;
- lerCompPend, recebe como parâmetro um objecto **Utilizador**, e retorna um conjunto de objectos **Compromisso**, que corresponderão aos compromissos que determinado utilizador tem como pendentes (à espera de confirmação);
- lerCompCanc, é um método semelhante ao anterior, com a diferença de retornar os compromissos que foram cancelados;
- lerCompAceites, é um método semelhante ao **lerCompPend**, com a diferença de retornar os compromissos que foram aceites;
- lerCompProxim, é um método semelhante ao **lerCompPend**, com a diferença de retornar os compromissos onde o valor da diferença entre a sua data e a data actual seja menor do que um certo limite;
- aceitarComp, recebe como parâmetros um objecto **Compromisso** e um **Utilizador**, tendo como função registar que o referido utilizador aceitou o respectivo compromisso;
- rejeitarComp, é um método semelhante ao anterior, mas cujo objectivo é rejeitar um compromisso.

Classe Disponibilidade

Com esta classe representa-se a informação que será necessário armazenar sobre uma disponibilidade presente na agenda de determinado utilizador. Com esta informação será possível classificar a disponibilidade como periódica ou ocasional, e também saber a que agenda (utilizador) pertence.

Bean Disponibilidade

Este **JavaBean** contém métodos que realizam operações relacionadas com o tratamento da informação referente às disponibilidades. Os métodos que deverão estar presentes são:

- escreveDisp, que recebe como parâmetro um objecto **Disponibilidade** e regista na base de dados a informação relativa à disponibilidade que se pretende marcar;
- apagaDisp, é um método semelhante ao anterior, mas cujo objectivo é apagar da base de dados a informação relativa a uma disponibilidade;
- alteraDisp, é um método semelhante ao **escreveDisp**, mas cujo objectivo é substituir a informação relativa a determinada disponibilidade, presente na base de dados;
- procuraDisp, recebe com parâmetro uma data, e retorna as disponibilidades existentes na data especificada;

Classe Tipo

Esta classe tem por objectivo representar a informação a guardar, para especificar os tipos de compromisso ou disponibilidade que será possível marcar na agenda. Está ligada à classe **Compromisso** por uma relação do tipo "Um-para-Muitos", visto que um determinado tipo pode estar presente em vários compromissos. A classe **Tipo** está também ligada à classe **Disponibilidade** através de uma relação do tipo "Muitos-para-Muitos". Isto acontece porque, uma disponibilidade pode ter vários tipos, e um tipo pode estar presente em várias disponibilidades.

5 Modelo Dinâmico

Nesta secção, é explicada a fucionalidade de alguns casos de uso mais relevantes, com o auxílio de alguns diagramas.

5.1 Página principal

A página principal é aquela que aparece ao utilizador quando este acede à *Agenda Web*. É a partir daqui que todas as funcionalidades da aplicação estão acessíveis. Para tal, existe um menu (do lado direito) onde se podem efectuar as operações que estão disponíveis (ver figura para uma melhor percepção)

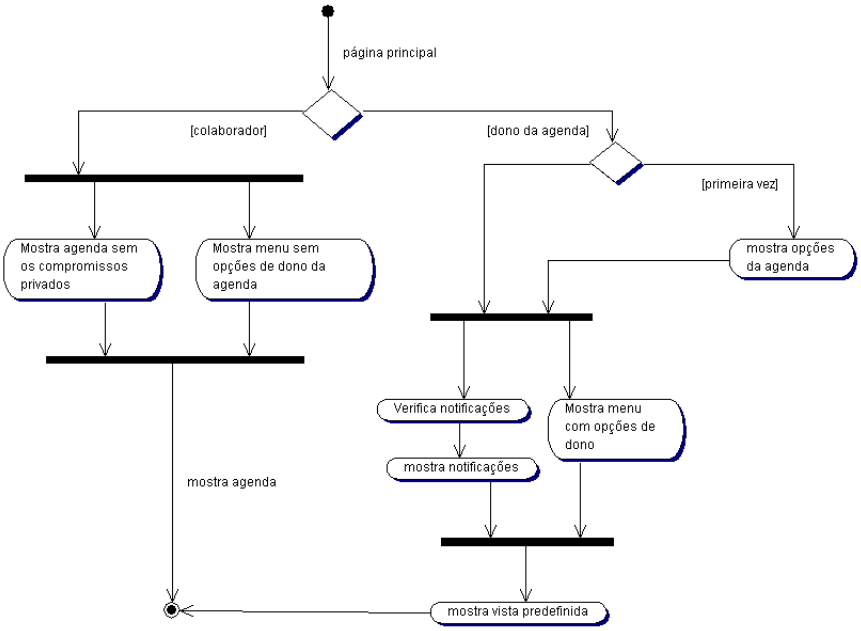

Figura 5.1: Conjunto de acções que ocorrem quando o utilizador acede à página principal.

5.2 Marcar Compromisso

Este diagrama descreve a sequência de operações necessárias para um **Dono de Agenda** marcar um compromisso público ou privado. A distinção entre público e privado é feita no preenchimento do formulário (**preenche formulário**). É também ao preencher o formulário que temos a opção de marcar um compromisso periódico.

Após o preenchimento do formulário é verificada a validade da data especificada. Por validade entende-se se os participantes têm disponibilidade para essa altura. No caso de a data não ser válida, é necessário analisar as disponibilidade e compromissos dos vários participantes e devolver um conjunto de hipóteses válidas para a data do compromisso[1]. Depois da escolha da data, o compromisso é, então, marcado.

O participante tem agora a opção de aceitar ou rejeitar o compromisso. Qualquer destas decisões é comunicada ao **Dono da Agenda** que marcou o compromisso. No caso de o participante rejeitar o compromisso, este é eliminado, tendo, o **Dono da Agenda** que marcou o compromisso, a opção de marcar um novo compromisso com os restantes participantes (caso existam). De notar, também, que esta interação com o participante não existe quando o **Dono da Agenda** está a marcar um compromisso que não tenha participantes (além do próprio, como é óbvio).

[1] Esta funcionalidade é descrita na secção *Verificar Disponibilidades* na página 24

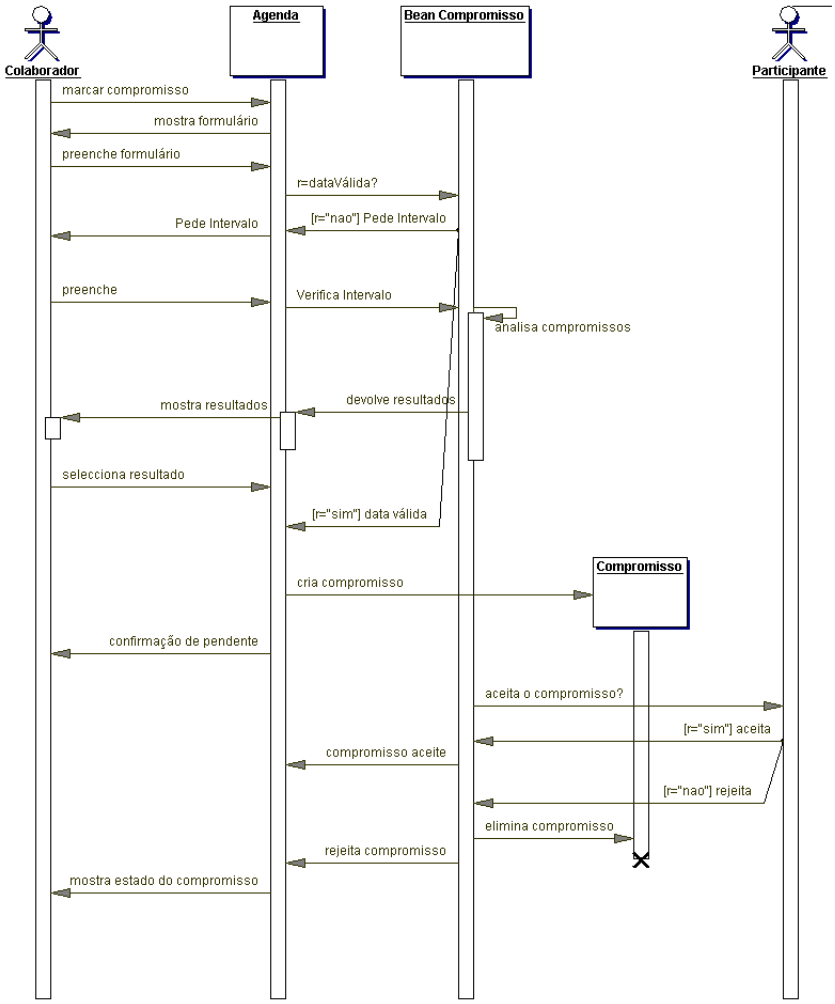

Figura 5.2: Diagrama de Sequência - Marcar Compromisso.

5.3 Marcar Disponibilidade

Este diagrama descreve as operações necessárias à marcação de novas disponibilidades. Após o preenchimento do formulário respectivo, uma nova **disponibilidade** é criada na agenda do **Dono da Agenda**.

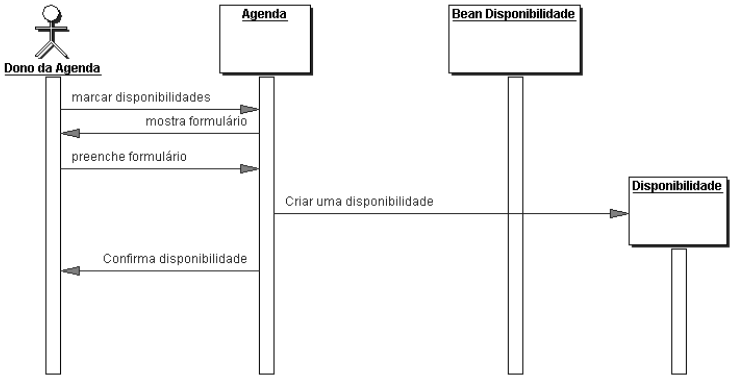

Figura 5.3: Diagrama de Sequência - Marcar Disponibilidade.

5.4 Eliminar Compromisso

O processo de eliminar um compromisso (público ou privado) é distinto, caso seja o próprio criador do compromisso (quem criou o compromisso que se deseja eliminar) a pedir a eliminação, ou um dos outros participantes no compromisso.

Na primeira situação, o criador do compromisso pede a eliminação deste, o que acontece de imediato. No entanto, todos os participantes do compromisso são avisados de tal facto.

No caso em que é um dos participantes a pedir a eliminação, o compromisso é eliminado logo, mas apenas visualmente. Quer isto dizer que, internamente, o compromisso ainda se encontra na base de dados. Ao mesmo tempo é enviada uma mensagem ao criador do compromisso a avisá-lo desta eliminação. Este tem, agora, a opção de criar um novo compromisso. Neste caso, os outros participantes são notificados da eliminação do compromisso original e questionados acerca do novo compromisso[2].

Alternativamente, o criador pode optar por não criar um novo compromisso, sendo, então, os participantes do compromisso eliminado, avisados de tal facto.

Em ambos os casos, só após esta decisão é que o compromisso original é realmente eliminado da base de dados, já que, os dados nele contidos são reutilizados na criação do novo compromisso.

[2]O processo de interacção do participante, quando questionado acerca do novo compromisso é idêntico ao descrito em **Marcar Compromisso**.

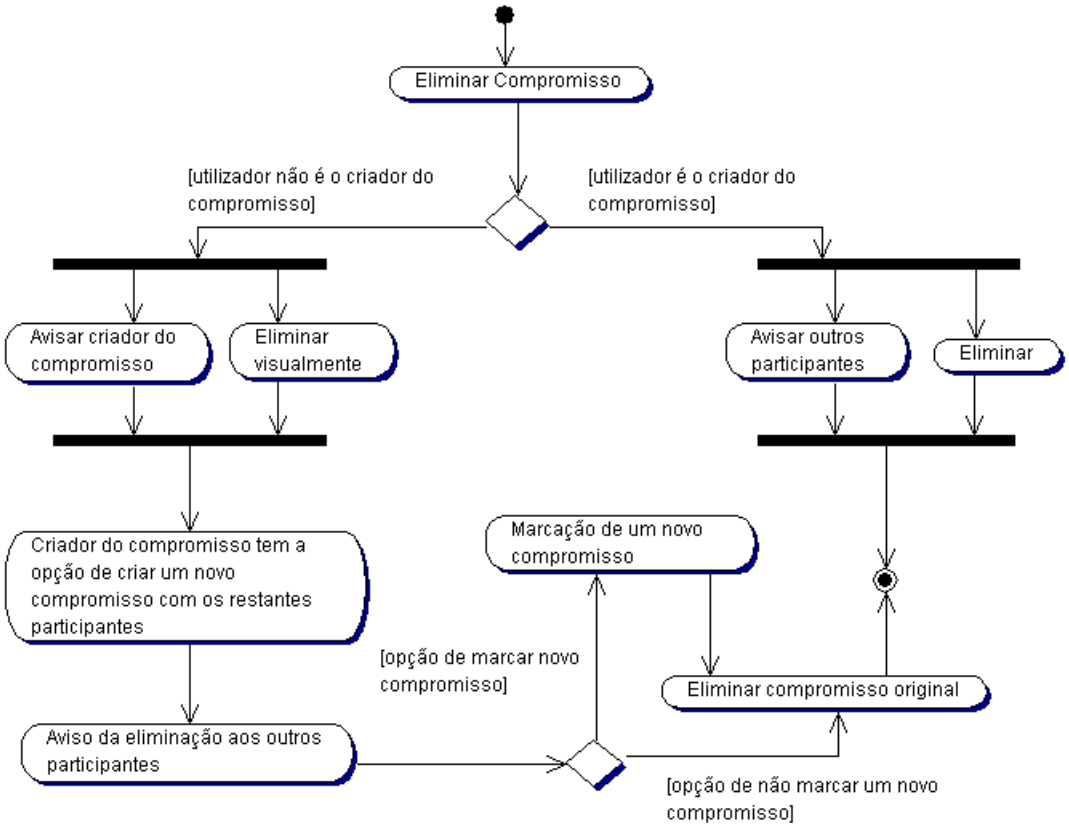

Figura 5.4: Diagrama de Actividades - Processo de eliminar um compromisso.

5.5 Verificar Disponibilidades

Para se obter uma data para a realização de um compromisso com vários participantes, é necessário ter em conta os seus compromissos e disponibilidades, dado que não é possível marcar compromissos que se sobreponham (com a excepção do próprio).

Em primeiro lugar, obtêm-se as disponibilidades existentes no intervalo pedido, dos vários participantes. Se houver uma data/hora coincidente entre todos e tiver uma duração suficiente para caber o compromisso, então, está encontrado o resultado.

No caso de não haver uma data/hora coincidente, obtêm-se os compromissos de todos os participantes, no dado intervalo e reúnem-se todos, "criando" uma só agenda com indisponibilidades e espaços livres (correspondem à intersecção dos espaços livres de todos os participantes). Esta informação é fornecida ao criador do compromissso, para que possa escolher entre as várias alternativas. Se não existir nenhum espaço livre, então, informa-se o criador de que não é possível encontrar uma data/hora em que todos os participantes não estejam ocupados no intervalo que indicou.

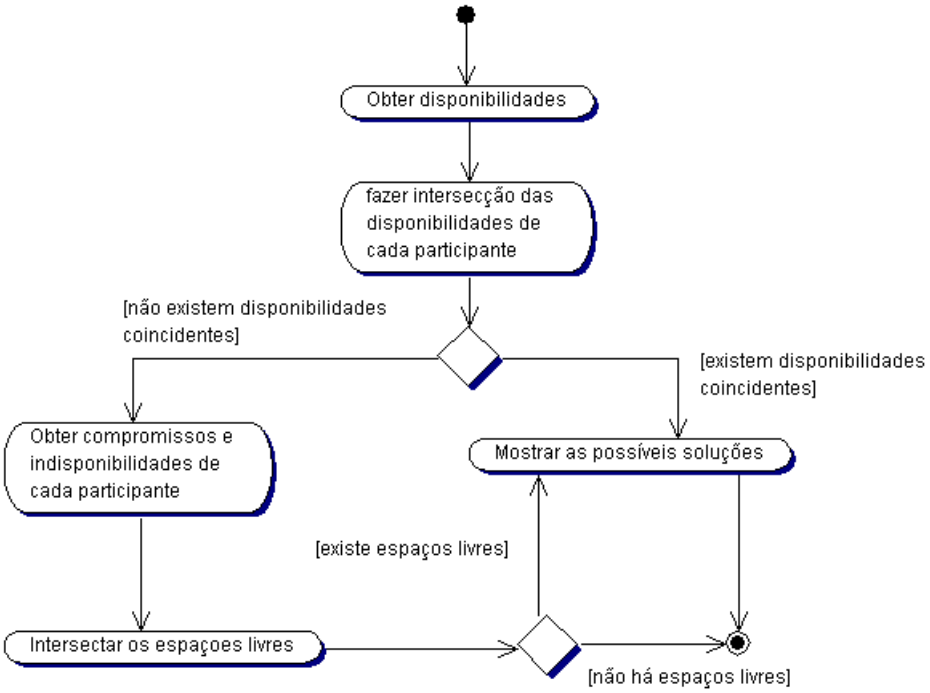

Figura 5.5: Diagrama de Actividades - Verificação de disponibilidades e obtenção de datas/horas livres para a marcação de compromissos.

6 Escolha das Tecnologias

A tecnologia utilizada pelo produto recebido é baseada em componentes, nomeadamente *Enterprise JavaBeans*. A escolha desta tecnologia parecia ser, de facto, a mais apropriada, devido à facilidade de integração dos componentes na base de dados (a utilização da base de dados é transparente).

No entanto, a utilização de *Enterprise JavaBeans* não foi bem sucedida, originando vários problemas ao nível da utilização dos *EJBs* em *Oracle*. Por esta razão, optou-se pelo uso de *Java Beans*, dado que também é uma tecnologia baseada em componentes, embora sem a integração na base de dados como os *EJBs*. De qualquer maneira, a divisão em camadas (interface, lógica de negócio, base de dados) não é posta em causa, dado que serão implementados *beans* específicos para manter esta estrutura.

A base de dados escolhida foi o *Oracle*, visto estar disponível na faculdade e suportar directamente os *JavaBeans*.

Em relação ao interface para o utilizador, este deverá ser acessível através de um *browser web*, tal como é pedido nos requisitos. Desta forma, a tecnologia usada será *Dynamic HTML*, ou seja a utilização de linguagens como *HTML*, *JavaScript* e *StyleSheets*. No entanto, é preferível o uso de uma linguagem com a capacidade de gerar este código dinamicamente. Para isso, optou-se por *Java Server Pages* (JSP) dado que é uma linguagem que está especialmente vocacionada para a utilização de *JavaBeans*.

7 Conclusão

Neste relatório ficou definida a arquitectura do sistema e foram apresentados vários diagramas explicativos do mesmo.

Relativamente ao produto recebido, do qual se deveria partir para desenvolver esta aplicação, optou-se por recomeçar do início e não aproveitar código do produto anterior, por várias razões. Primeiro, o interface, tal como é descrito nos requisitos, deve ser um interface *web*. Depois, os problemas encontrados no uso dos *Enterprise JavaBeans* impossibilitaram a sua utilização[3].

Devido à mudança de tecnologia, houve necessidade de uma grande reestruturação na arquitectura, mais particularmente na camada da base de dados (em *EJBs* a integração na base de dados era transparente) em que foi necessário criar um modelo relacional, além do desenvolvimento de uma *API* para o acesso à base de dados. Assim, este relatório evoluiu para a versão 2.0.

[3] ver *Escolha das Tecnologias* na página 26

8 Glossário

Disponibilidade Uma disponibilidade pode ser de um ou mais tipos e indica a possibilidade de se poder marcar um compromisso do mesmo tipo para essa altura. Por exemplo, uma disponibilidade do tipo reunião e aulas, indica que se podem marcar reuniões ou aulas, na altura definida nessa disponibilidade.

Colaborador da Organização Um colaborador é uma pessoa da organização que tem acesso às agendas dos outros colaboradores, podendo visualizar nessas agendas as disponibilidades e compromissos públicos.

Dono da Agenda O dono da agenda é a pessoa à qual é permitido criar/alterar disponibilidades e compromissos.

Compromisso público É um compromisso marcado por um elemento da organização, sendo sempre visível o seu tipo e descrição.

Compromisso privado É um compromisso marcado por um elemento da organização, sendo que o seu tipo e descrição apenas é visível para o dono da agenda.

Indisponibilidade Uma indisponibilidade numa agenda reflecte a impossibilidade de se marcar compromissos nessa altura, por outro elemento que não o dono dessa agenda.

Utilizador Utilizador representa o conjunto de pessoas que poderão utilizar a Agenda Web, nomeadamente o colaborador da organização e o dono da agenda.

Browser É um software que deve estar presente no computador para permitir que um utilizador possa navegar na Internet (WWW) e consultar páginas HTML (*Hypertext Markup Language*);

JavaBeans É um interface de programação orientado a objectos que permite criar blocos de código reutilizáveis, chamados componentes (*beans*).

Bibliografia

[Faria, 2000] Faria, J. P. (2000). *Modelos de Arquitectura em UML*. Faculdade de Engenharia da Universidade do Porto, LEIC, http://www.fe.up.pt/ jpf/teach/LIA/acetatos/arquitectura.ppt.

A Anexo

A.1 Resumo das alterações efectuadas em relação à versão 1.0

1.3 Arquitectura Onde se lia "(...) para a arquitectura física vai ser apresentado um diagrama de componentes e um diagrama de distribuição (...)", passa a ler-se "(...) para a arquitectura física vão ser apresentados diagramas de componentes, modelo relacional da base de dados e diagrama de distribuição (...)".

1.4 Tecnologia Esta secção foi alterada de forma a reflectir as mudanças relacionadas com a tecnologia utilizada.

2.1 Diagrama de Pacotes Foi alterado o diagrama de pacotes para comportar a mudança de tecnologia;

2.2 Descrição do diagrama Foi alterada a descrição para comportar as alterações efectuadas no diagrama;

3.1 Diagrama de distribuição Foi alterado o diagrama de distribuição para comportar a mudança de tecnologia;

3.1.1 Descrição do diagrama Foi alterada a descrição para comportar as alterações efectuadas no diagrama;

3.2 Diagrama de componentes - Organização física da aplicação Foi aprofundado o diagrama;

3.2.1 Descrição do diagrama Foi alterada a descrição do diagrama (da secção 3.2.) para comportar as alterações efectuadas no mesmo;

3.3 Diagrama de componentes - Organização física da base de dados Foi acrescentado este diagrama para descrever a estrutura da base de dados da aplicação;

3.3.1 Descrição do diagrama Foi acrescentada a descrição do diagrama (da secção 3.3);

3.4 Modelo relacional da base de dados Agenda Foi acrescentado o modelo relacional da base de dados da aplicação;

3.4.1 Descrição do modelo Foi acrescentada a descrição do modelo relacional da base de dados;

4.1 Foi modificado o diagrama de classes tendo em conta a mudança de tecnologia;

4.1.1 Descrição do diagrama Foi modificada a descrição do diagrama de classes para comportar as alterações efectuadas no mesmo;

5 Modelo Dinâmico Devido a uma reestruturação desta secção do relatório, os nomes das sub-secções não correspondem ao utilizados na versão anterior do relatório de projecto. Por isto, não é possível descrever, sub-secção a sub-secção, as modificações aqui efectuadas. Esta reestruturação tem como objectivo dar um maior ênfase nas funcionalidades (casos de uso) e não directamente nos diagramas que são utilizados para as descrever. Relativamente ao conteúdo, foram acrescentadas descrições de funcionalidades (e respectivos diagramas) que não se encontravam anteriormente descritas. Estas são: **Página Principal** e **Eliminar Compromisso**. Por outro lado, a funcionalidade de **Mostrar Agenda**, presente no relatório anterior, foi retirada deste. Quanto ao **Marcar Compromisso**, **Marcar Disponibilidade** e **Verificar Disponibilidade**, foram modificados. Estas alterações são devido a um maior detalhe, atingido por este relatório, assim como a redefinição de alguns conceitos e requisitos, presentes na versão 1.1 do relatório de especificação de requisitos.

6 Escolha das Tecnologias Devido à mudança de tecnologia efectuada, esta secção, foi, como é óbvio, modificada quase por completo.

7 Conclusão Dado que a conclusão de um relatório é algo muito específico ao relatório, esta foi completamente reescrita, reflectindo as modificações que foram feitas à versão 1.0 do relatório de projecto.