



Universidade do Porto
Faculdade de Engenharia

FEUP



Nuno Miguel Rodrigues Monteiro Aguiar

HUMAN CAPITAL TEAM TOOL - MÓDULO DE SEGURANÇA

na IBS Portugal

C
GUn

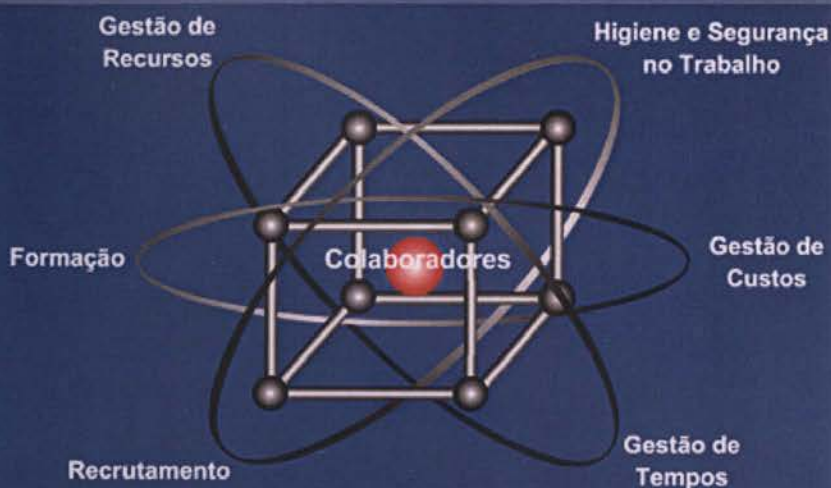
LEIC

Setembro, 2004

Human Capital Team Tool

Módulo de Segurança

Human Capital Team Tool



Orientador da FEUP
Professor Ademar Aguiar

Estágio realizado por
Nuno Miguel Rodrigues Monteiro Aguiar
Nuno.Aguiar@ibs.pt

Orientador da Instituição
Engenheiro Gonçalo Gil Mata



**Faculdade de Engenharia da Universidade do Porto
Licenciatura em Engenharia Informática e Computação**



Human Capital Team Tool - Módulo de Segurança

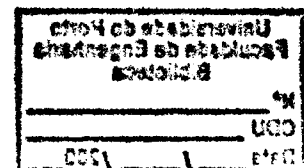
IBS PORTUGAL

Relatório do Estágio Curricular da LEIC 2004

Nuno Miguel Rodrigues Monteiro Aguiar

Orientador na FEUP: Professor Ademar Aguiar
Orientador na IBS: Engenheiro Gonçalo Gil Mata

Setembro de 2004



004 (047.3) LEEC/EEC 5202 2004/AGUm

Universidade do Porto	
Faculdade de Engenharia	
Biblioteca	
Nº	81490 7
CDU	621.01 1042.51
Data	20/03/2006

*Aos meus Pais,
pela sua enorme vontade.*

*À Teresa, que apesar de ausente,
está sempre presente.*

Sumário

O departamento de JAVA da IBS Portugal tem vindo a desenvolver, nos últimos três anos, uma aplicação que suporta a gestão de recursos humanos de uma empresa. A aplicação – a Human Capital Team Tool (HCTT) – abarca um espectro largo de actividades deste processo, ao qual, cada vez mais, as empresas começam a dar importância.

Uma das consequências directas desta abrangência – e porque neste processo de gestão participam todos os colaboradores de uma empresa – é a preocupação com questões de segurança, como por exemplo, as autorizações e os privilégios de acesso dos utilizadores. O HCTT é uma aplicação orientada para a *internet* acedida a partir de um *browser* através da *intranet*¹. Esta utilização é alargada a todos os colaboradores de uma empresa. Foi portanto necessário dotar a aplicação da capacidade de gerir os privilégios de utilização. Esta funcionalidade permite proteger as informações e as operações sensíveis de acessos indesejados. Depois de implementada, é possível dar resposta à problemática: “*quem* tem acesso, e a *quê?*”.

O objectivo deste relatório de estágio é descrever as actividades desenvolvidas no âmbito da implementação destas funcionalidades que se enquadram num processo típico de engenharia de software e que, por isso, aqui se destacam: análise do problema; especificação dos requisitos; implementação das funcionalidades; integração no código já desenvolvido; teste. Na descrição destas actividades, procurou-se um compromisso entre o detalhe e a superficialidade, de forma a que o relatório pudesse ser lido por pessoas com formação em áreas distintas da Engenharia Informática, sem perda do seu valor documental.

Além de todo este processo, houve um primeiro contacto com as metodologias praticadas e as tecnologias utilizadas com os objectivos de, por um lado, integrar o estagiário na equipa de desenvolvimento e, por outro, dar-lhe a informação e os meios necessários para arrancar com o processo de desenvolvimento referido. Desta primeira fase destaca-se o contacto com o HCTT, quer sob o ponto de vista utilização, quer sob o ponto de vista técnico de implementação.

O relatório termina com um conjunto de conclusões importantes. Avalia-se a versão final do módulo de segurança sob parâmetros funcionais, de performance e de arquitectura, analisando o custo da sua manutenção e evolução no futuro. De seguida, efectua-se uma análise dos desvios entre a solução efectivamente implementada e a especificação inicial. Apresentam-se também algumas conclusões sobre o método de desenvolvimento seguido, identificando pontos críticos que afectaram o processo de uma forma um pouco menos positiva, apontando algumas sugestões para melhorá-los.

¹ Um *browser* é uma aplicação que usualmente se utiliza para aceder à *internet* e que muitas vezes se utiliza para aceder a informação que apenas se disponibiliza através da rede interna de uma empresa (*intranet*).

Agradecimentos

O meu sincero obrigado à IBS, pelas condições que me proporcionou, na fase de integração inicial e ao longo de todo o estágio. Especificamente, gostaria de destacar o apoio incondicional e disponibilidade permanente da Engenheira Bárbara Costa, coordenadora do departamento de JAVA, pela atenção, espírito crítico e raciocínio que me ajudaram a questionar correctivamente as minhas opções e a evoluir tecnicamente. Ao mesmo nível, a orientação do Engenheiro Gonçalo Gil Mata, aqui na empresa, revelou-se extremamente valiosa e complementar ao seguimento técnico referido, agora num patamar mais analítico e funcional, mas não menos importante.

O professor Ademar Aguiar, docente da Faculdade de Engenharia, foi a pessoa que me acompanhou durante o estágio, na sua componente académica. Toda a sua atenção, interesse e disponibilidade vieram corrigir alguns trajectos, por vezes mais sinuosos. O seu espírito prático e sugestões pertinentes contribuíram significativa e positivamente para diversas decisões.

Aos meus colegas de trabalho, que me proporcionaram um ambiente agradável e propício a estímulos construtivos e produtivos.

Aos meus amigos, que colmataram eficazmente o desgaste emocional desta prova importante que é o estágio curricular.

E aos meus pais, porque tornaram tudo possível, e a quem nunca conseguirei agradecer o suficiente.

Tabela de Conteúdos

1	INTRODUÇÃO	6
1.1	Organização do conteúdo	6
1.2	A Empresa IBS (International Business Systems)	7
1.3	O Produto Human Capital Team Tool (HCTT)	8
1.3.1	Informação Nuclear	10
1.3.2	Módulos Funcionais	11
1.3.3	Ferramentas	13
2	TECNOLOGIAS	14
2.1	Frameworks	14
2.1.1	UNIIFace	14
2.1.2	Persistence Builder	15
2.2	Linguagens	17
2.3	Ferramentas	18
2.3.1	Ambiente de Desenvolvimento Integrado	18
2.3.2	Registo de Erros/Plataforma de Colaboração	18
2.3.3	Outras ferramentas	18
2.4	Servidores	19
3	O MÓDULO DE SEGURANÇA	20
3.1	Objectivos	20
3.1.1	Segurança de Acesso	21
3.1.2	Segurança de Execução	21
3.1.3	Gestão/Controlo de Autorizações	22
3.2	Metodologia da Equipa de Desenvolvimento	23
3.3	Planeamento de Actividades	24
4	ESPECIFICAÇÃO	25
4.1	Generalidades	25
4.2	Manutenção de Utilizadores	26
4.3	Manutenção de Grupos de Utilizadores	28
4.3.1	Adicionar/Remover Utilizadores	29
4.3.2	Criar/Apagar Grupos	33
4.4	Manutenção de Grupos de Comandos	35
4.4.1	Criar/Apagar Grupos	39
4.4.2	Manutenção Isolada	40
4.4.3	Manutenção XY	41
4.4.4	Manutenção YX	43
4.5	Manutenção de Perfis de Segurança	45
5	IMPLEMENTAÇÃO	48
5.1	Arquitectura Lógica do HCTT	48
5.2	Arquitectura Física do HCTT	50
5.3	Base de Dados	51
5.4	Mecanismos Importantes	53
5.4.1	Processo de Autenticação	53
5.4.2	Segurança de Execução	55
5.4.3	Actualização/Integridade das Autorizações	55
5.5	Implementação dos Casos de Utilização	56
5.5.1	Interface Gráfica	57
5.5.2	Lógica	58
6	CONCLUSÕES	64
7	REFERÊNCIAS	66
8	GLOSSÁRIO	67

Índice de Figuras

Figura 1 – Página <i>web</i> inicial do HCTT.	9
Figura 2 – Árvore da estrutura.	11
Figura 3 – Geração de JSP a partir de processos XML.	15
Figura 4 – Esquema do funcionamento da arquitectura UniFace.	15
Figura 5 – Camadas arquitecturais do <i>Persistence Builder</i>	16
Figura 6 – <i>Frameworks</i> em que se decompõe o <i>Persistence Builder</i>	17
Figura 7 – Diagrama simplificado da arquitectura lógica do HCTT.	22
Figura 8 – Metodologia de desenvolvimento adoptada.	23
Figura 9 – Cronograma de actividades do estágio.	24
Figura 10 – Organização dos casos de utilização.	25
Figura 11 – Página <i>web</i> principal (inicial) do módulo de segurança.	26
Figura 12 – Pacote de casos de utilização “Manutenção de Utilizadores”.	27
Figura 13 – Página <i>web</i> da manutenção de utilizadores.	28
Figura 14 – Pacote de casos de utilização “Adicionar/Remover Utilizadores”.	29
Figura 15 – Página <i>web</i> da manutenção de grupos de utilizadores.	31
Figura 16 – Manutenção simples de grupos de utilizadores.	31
Figura 17 – Manutenção a partir da estrutura orgânica da empresa.	32
Figura 18 – Manutenção a partir de outros grupos de utilizadores.	33
Figura 19 – Pacote de casos de utilização “Criar/Apagar Grupos” (de utilizadores).	34
Figura 20 – Página <i>web</i> de criação de um grupo de utilizadores.	34
Figura 21 – Opções de manutenção do menu “Grupo”.	35
Figura 22 – Árvore de todos os comandos (vista <i>sub-área/tipo, com nós vazios</i>).	36
Figura 23 – Árvore de todos os comandos (vista <i>tipo/sub-área, sem nós vazios</i>).	38
Figura 24 – Página <i>web</i> de manutenção de grupos de comandos.	38
Figura 25 – Pacote de casos de utilização “Criar/Apagar Grupos” (de comandos).	39
Figura 26 – Página <i>web</i> de criação de grupos de comandos.	39
Figura 27 – Pacote de casos de utilização “Manutenção Isolada”.	40
Figura 28 – Página <i>web</i> de manutenção isolada de grupos de comandos.	41
Figura 29 – Pacote de casos de utilização “Manutenção XY”.	42
Figura 30 – Página <i>web</i> de manutenção XY de um grupo de comandos.	43
Figura 31 – Pacote de casos de utilização “Manutenção YX”.	44
Figura 32 – Página <i>web</i> de manutenção YX de um grupo de comandos.	45
Figura 33 – Pacote de casos de utilização “Manutenção de Perfis de Segurança”.	46
Figura 34 – Página <i>web</i> de selecção do perfil de segurança.	46
Figura 35 – Página <i>web</i> de manutenção de perfis de segurança.	47
Figura 36 – Arquitectura lógica do sistema – camadas.	49
Figura 37 – Arquitectura lógica da aplicação HCTT.	50
Figura 38 – Arquitectura física da aplicação HCTT.	51
Figura 39 – Diagrama do modelo relacional de base de dados.	52
Figura 40 – Processo de autenticação: classes participantes.	54
Figura 41 – Processo de autenticação: diagrama de colaboração.	54
Figura 42 – Hierarquia de comandos.	55
Figura 43 – Componentes do módulo de segurança.	56
Figura 44 – Ligação da interface gráfica à lógica de negócio.	58
Figura 45 – Lógica do método <i>ProfileAdapter.loadObject()</i> : classes participantes.	60
Figura 46 – Lógica do método <i>ProfileAdapter.loadList()</i> : classes participantes.	61
Figura 47 – Lógica do método <i>ProfileAdapter.toggleAccess()</i> : classes participantes. ..	62
Figura 48 – Lógica do método <i>ProfileAdapter.updateProfile()</i> : classes participantes. .	63
Figura 49 – Navegador genérico do HCTT.	67

1 Introdução

O estágio curricular é o último grande teste colocado aos alunos da licenciatura em Engenharia Informática e Computação, na Faculdade de Engenharia da Universidade do Porto. Ao fim de quatro anos e meio de estudos, período no qual se adquire e apreende conhecimento precioso nesta área, o aluno é confrontado com uma prova prática efectuada numa empresa, em contexto real de trabalho, que terá a duração de cerca de seis meses.

As actividades deste estágio decorreram no departamento de JAVA da IBS Portugal, em Vila Nova de Gaia. A IBS é uma empresa multinacional, com sede em Estocolmo, na Suécia, cuja principal actividade é o desenvolvimento e consultoria de software informático, para médias e grandes empresas.

No capítulo introdutório deste relatório apresenta-se a instituição e o departamento, o projecto HCTT e a primeira abordagem analítica de alto nível efectuada, que consistiu na distribuição temporal das tarefas a executar. Iremos começar por apresentar a estrutura e organização dos temas abordados neste documento, com vista a permitir orientar uma leitura eficaz.

1.1 Organização do conteúdo

Este documento é composto por **seis capítulos**. Após a presente **introdução**, será feito um enquadramento tecnológico do estágio no capítulo 2, que descreve o contexto técnico no qual a análise e implementação do problema serão baseadas.

No capítulo 3 apresenta-se o problema e os objectivos do **módulo de segurança** e também questões genéricas relacionadas com o estágio, como por exemplo o seu planeamento. Servirá como uma introdução ao capítulo 4, onde se **especificam os requisitos** do módulo, suportado em diagramas de casos de utilização e em metodologias da engenharia de requisitos de software.

Segue-se mais um capítulo técnico onde se descreve a **implementação**, o capítulo 5. São apresentadas a arquitectura lógica e física da aplicação HCTT e também informação técnica detalhada de algumas soluções implementadas.

Por fim, no último capítulo, serão apresentadas as **conclusões** e também uma reflexão global sobre o estágio. Procedede-se à análise dos resultados, faz-se uma referência a alguns problemas identificados no processo de desenvolvimento e apresentam-se algumas sugestões de melhoria.

1.2 A Empresa IBS (International Business Systems)

O Grupo IBS é um dos líderes de mercado em soluções de gestão empresarial. Tem vindo a desenvolver produtos para mais de 5.000 clientes oriundos de mais de 40 países. Estes clientes pertencem a um espectro alargado de mercados, mas são principalmente empresas de média e grande dimensão em áreas como Indústria, Distribuição, Serviços, Sector Automóvel, etc. Na vasta lista de empresas clientes podemos encontrar nomes de referência como Electrolux, Maxell, Fuji Film, Mitsubishi, Nintendo, DIM, Pioneer e Volvo.

A missão da IBS passa por combinar as mais avançadas tecnologias de informação disponíveis com o conhecimento profundo detido sobre diferentes processos e procedimentos de negócio praticados em distintos segmentos de actividades, no sentido de fornecer sistemas de gestão empresariais completos e integrados, que garantam a maximização das vantagens competitivas dos seus clientes. A aliança estratégica que detém com a IBM enfatiza o seu carácter inovador, em sintonia com o mercado. A distinção sucessiva pela sua parceira, através de prémios anuais, é um dos sinais de crédito profissional.

O Grupo IBS opera globalmente no mercado, através de uma rede mundial de escritórios e de parceiros de negócio, tendo como principal actividade a oferta de soluções integradas de software e serviços profissionais na área de tecnologias de informação, destinadas a empresas nacionais ou internacionais.

Em Portugal a IBS está representada em Linda-a-Velha e em Vila Nova de Gaia. O estágio foi realizado em Vila Nova de Gaia, que se apresenta como a sede da empresa, a partir de onde são, recentemente, coordenadas as actividades a nível ibérico. Em Portugal, a empresa reúne aproximadamente 200 colaboradores. Os departamentos de análise e desenvolvimento JAVA são os que estiveram directamente envolvidos no estágio.

Na equipa que tem vindo a desenvolver o HCTT trabalham cerca de 10 pessoas, distribuídas pelas áreas de gestão, análise e desenvolvimento. Alguns elementos estão dedicados a tempo inteiro a este projecto e outros trabalham simultaneamente noutros projectos da empresa.

1.3 O Produto Human Capital Team Tool (HCTT)

Actualmente acrónimos como CRM² e ERP³ fazem parte do quotidiano das grandes empresas. Os grandes investimentos na área da informática invadiram todos os sectores nesta “era digital” e as empresas começam agora a voltar atenções para o seu capital humano, identificando-o como uma das suas mais-valias estratégicas. Actualmente, para que uma empresa seja competitiva, é necessário investir mais no capital humano, convertendo este impasse inevitável num elemento catalisador da criatividade construtiva, para superar obstáculos e vencer novos desafios.

Neste contexto de evolução e mudança, a gestão de recursos humanos assume um papel preponderante a nível da eficácia organizacional, que resulta da interligação entre a qualidade das decisões estratégicas e a qualidade da informação sobre os Recursos Humanos, no processo de tomada de decisão. Torna-se no entanto necessário que este processo de gestão se torne perfeitamente integrado em todo o processo de gestão da empresa.

O HCTT foi concebido para dar uma resposta inovadora à necessidade cada vez mais premente de uma gestão eficaz do capital humano, tornando possível gerir de uma forma integrada e global os recursos humanos da organização. Orientada ao mercado de médias e grandes empresas, está dividida em vários módulos funcionais, com uma abrangência multidisciplinar que passa pela gestão de competências, recrutamento, formação, custos, tempos, saúde, higiene e segurança no trabalho, entre outros. Podemos verificar na Figura 1 o aspecto da página inicial do HCTT e os seus diversos módulos.

A ferramenta desenvolvida pela IBS permite organizar o trabalho do gestor de recursos humanos, nas mais diversas áreas, de modo a criar sinergias para ligar as pessoas entre si, permitindo que seja feita uma gestão previsional de forma a criar e gerir cenários futuros. As mutações constantes exigem uma nova dinâmica para a gestão de recursos humanos. O HCTT implementa essa dinâmica de uma forma inovadora. A fundamental importância de uma participação global neste processo de gestão fez com que a aplicação fosse também pensada para uma utilização distribuída pela organização e por todos os seus colaboradores. Uma das consequências directas desta decisão foi a preocupação com questões de segurança.

² *Customer Relationship Management* (gestão da relação com o cliente) – aplicações de software de larga escala que permitem às empresas gerir todos os aspectos da sua relação com o cliente. O objectivo destes sistemas é assistir a construção de relações com os clientes duradouras: converter a satisfação do cliente em fidelidade.

³ *Enterprise Resource Planning* (planeamento de recursos da organização) – sistemas de software desenhados para suportar e automatizar os processos de negócio de médias e grandes empresas. Isto pode incluir manufactura, distribuição, pessoal, gestão de projectos, salários e gestão de finanças.

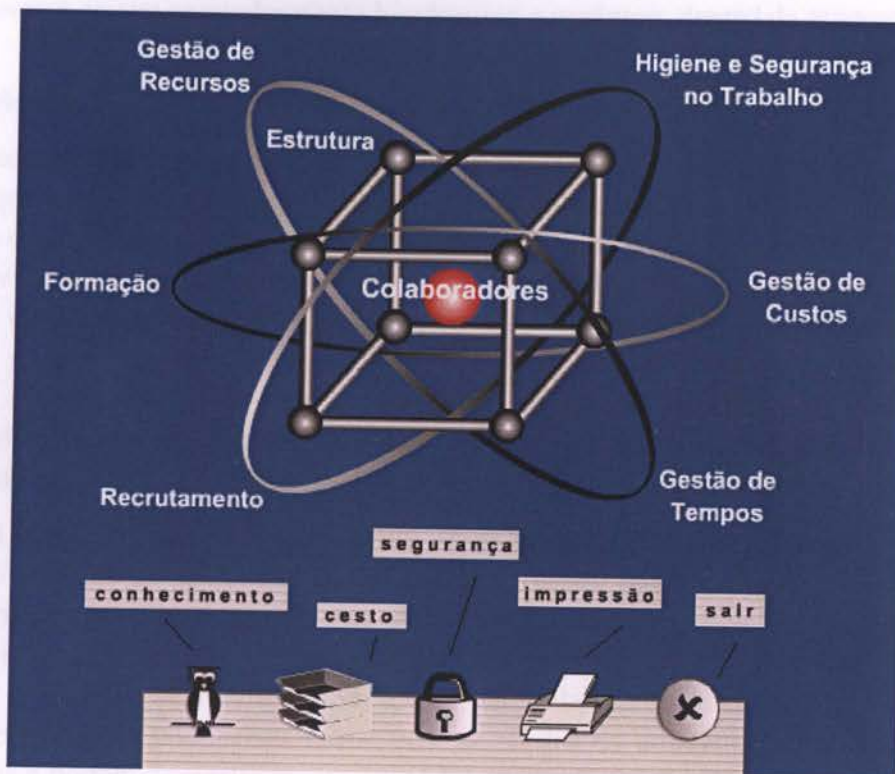


Figura 1 – Página web inicial do HCTT.

O processo de análise e estudo dos requisitos da aplicação, tornou imprescindível e prioritário dotar a aplicação da capacidade de gerir os privilégios de utilização. Sem esta funcionalidade, não seria possível proteger as informações e as operações sensíveis de acessos indesejados. Isto queria dizer que não se poderia oferecer ao gestor de recursos humanos da organização uma ferramenta que lhe permitisse ter o controlo do processo e, ao mesmo tempo, incluir a participação activa dos colaboradores nesse mesmo processo. A realidade do módulo de segurança virá quebrar as limitações existentes a este nível, complementando as funcionalidades básicas de segurança já existentes.

O HCTT é uma aplicação do tipo *cliente-servidor*⁴, orientada para a *intranet/internet*. Numa estrutura típica de três camadas (*“3-tier”*), a aplicação apresenta uma segmentação flexível que tem vindo a permitir o acompanhamento atento da evolução tecnológica na área, cuja integração tem pautado o seu desenvolvimento. É uma aplicação montada sobre um ambiente *web*, implementada na linguagem JAVA e nas tecnologias JSP, JavaScript, XML e outras.

⁴ Uma aplicação *cliente-servidor* significa que reside fisicamente num computador central (servidor) e permite a utilização distribuída, sendo o acesso feito a partir de outros computadores (clientes).

1.3.1 Informação Nuclear

A informação nuclear do HCTT está concentrada na informação relativa aos colaboradores e ao seu enquadramento hierárquico e orgânico, dentro da empresa.

Colaboradores

Começamos por aceder, no módulo respectivo, à lista de todos os colaboradores da empresa, que está organizada por: número individual de funcionário, nome completo e nome pelo qual a pessoa é conhecida. Podemos abrir cada elemento desta lista, aceder à ficha individual de cada colaborador e editar os dados de cada pessoa. Na ficha individual visualizaremos informação relativa ao perfil, documentos externos e outros dados da pessoa: contactos, dados pessoais, dados profissionais, dados clínicos, processos disciplinares, habilitações literárias, formação profissional, experiência profissional, informação adicional, posição que a pessoa ocupa na estrutura orgânica e passatempos.

Existe uma funcionalidade útil que permite pesquisar colaboradores através do nome ou do número individual. Adicionalmente, podemos ainda reduzir o universo da nossa pesquisa através dos *filtros*, através das mais variadas características dos colaboradores (idade, grau académico, nível salarial, etc.). Os *filtros* são uma ferramenta flexível e podem ser gerais ou particulares. Se forem particulares, “pertencem” apenas a um utilizador e só ele os pode visualizar e utilizar. Todos os utilizadores têm acesso aos filtros gerais e podem, inclusivamente, torná-los particulares.

Estrutura

O módulo da estrutura permite aceder à informação dos vários departamentos da empresa, organizados numa árvore⁵, ilustrada na Figura 2. Existe uma e uma só estrutura principal e, possivelmente, várias secundárias que poderão ser, por exemplo, estruturas de projectos específicos.

Neste módulo não existe o conceito de *pertença (ownership)*, de forma que todas as estruturas e nós são visualizados pelo utilizador. Associados a cada tipo de nó (na árvore da estrutura, são nós que correspondem usualmente a departamentos, funções, e posições) existe um conjunto de funcionalidades e operações disponíveis, à semelhança do que acontece num outro módulo da aplicação, o módulo de Gestão de Recursos.

Acedemos à estrutura e começamos por visualizar informação relativa à missão e responsáveis pela estrutura. As pessoas que aparecem como responsáveis⁶ pela estrutura têm acesso à visualização da informação que aparece hierarquicamente abaixo do local

⁵ O conceito de *árvore* é bastante utilizado no HCTT. Poderá encontrar-se uma definição um pouco mais formal de *árvore* na nota de rodapé 12, na página 30. Entretanto, na Figura 2, podemos ver um exemplo de uma árvore, que representa uma estrutura. Vemos departamentos, funções e posições, dependendo do nível da árvore, respectivamente.

⁶ O conceito de *responsável* corresponde a uma restrição de segurança e está definido na secção 3.1.2, na página 21.

onde eles estão definidos como responsáveis, ou seja, é-lhe permitido visualizar informação acerca de departamentos, funções e posições.

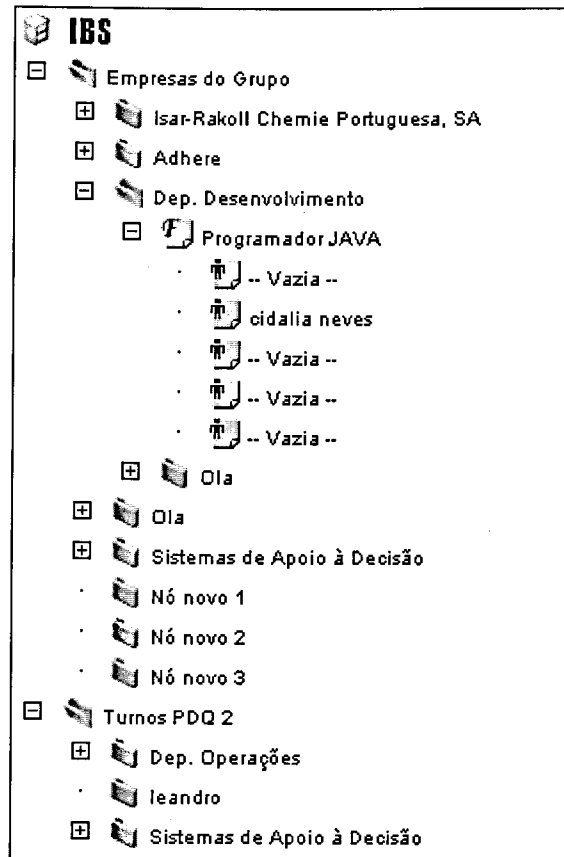


Figura 2 – Árvore da estrutura.

1.3.2 Módulos Funcionais

Vimos já na Figura 1 que a aplicação HCTT disponibiliza vários módulos aplicativos que gerem a informação nuclear sobre perspectivas diferentes. Cada módulo pretende suportar cada área específica das actividades diversas integradas no processo de gestão de recursos humanos.

Recrutamento

O recrutamento e selecção de pessoas devem ser vistos como duas fases de um mesmo processo: a introdução de colaboradores na organização. O objectivo do recrutamento é atrair com selectividade, mediante várias técnicas de divulgação, candidatos que possuam os requisitos mínimos da posição a ser preenchida, enquanto que o objectivo da selecção é a de escolher, entre os candidatos recrutados, aqueles que reúnam maiores possibilidades de se ajustar à posição que queremos preencher.

Neste módulo podemos introduzir, visualizar e gerir informação sobre os processos de selecção e também informação relativa aos candidatos. Dentro deste módulo há uma

ferramenta – o *Analyser* – que nos permite obter indicadores sobre os processos de recrutamento, selecção e candidatos.

Gestão de Recursos

A gestão de recursos é a actividade que consiste na integração, coordenação, manutenção e retenção dos colaboradores dentro da organização.

No HCTT, a gestão de recursos aparece-nos com uma imagem muito semelhante à apresentada pela estrutura. Conseguimos visualizar informação relativa a estruturas, nós, funções, posições e plano de desenvolvimento global.

É possível realizar todas as tarefas de gestão de pessoas: introdução das pessoas na respectiva posição para a qual foram admitidas, análise do plano individual de formação da pessoa, comparação do perfil da pessoa, visualizar a localização dessa pessoa na estrutura orgânica, análise da possibilidade de mudança de função e também proceder à avaliação da pessoa.

É no módulo de gestão de recursos que se desenvolve todo o processo de gestão do capital humano dentro da organização.

Formação

A formação profissional é um processo educacional que visa preparar e formar a pessoa para o exercício de uma profissão. Muitas vezes está relacionado com o processo através do qual a pessoa amplia os seus conhecimentos e permite o desenvolvimento e aperfeiçoamento para o crescimento profissional, em termos da sua carreira na empresa.

No módulo de formação podemos encontrar informação relativa aos cursos de formação e também relativa aos formandos. Os cursos de formação estão organizados por várias áreas de formação e dentro dessas áreas é possível constituir turmas, inscrever e avaliar os formandos.

Em relação aos formandos é possível diagnosticar as necessidades de formação e aceder ao plano individual de cada pessoa.

Dentro deste módulo existe um sub-módulo que diz respeito ao planeamento da formação, um elemento muito importante na tarefa de gestão da formação.

Higiene e Segurança no Trabalho

Segundo a OMS (Organização Mundial de Saúde), o objectivo da medicina do trabalho é prevenir o mais completo bem-estar físico, psíquico e social de cada pessoa no seu local de trabalho. Tal significa promover e manter a segurança no seu meio de trabalho, promover a adaptação da pessoa ao posto de trabalho, avaliar com vista à prevenção de doenças e por fim educar e informar sobre os aspectos de saúde.

A legislação em vigor desde 1999, torna esta preocupação com as questões da Segurança, Higiene e Saúde no Trabalho (SHST) obrigatórias a todas as organizações.

O módulo de SHST do HCTT apresenta-se como uma eficaz ferramenta de gestão deste processo, nas organizações. É possível através deste módulo aceder a informação sobre os postos de trabalho, riscos, factores de risco, actividades e acções.

1.3.3 Ferramentas

O HCTT disponibiliza ainda algumas ferramentas que, ainda que não tenham a dimensão e a abrangência funcional para serem consideradas módulos aplicativos complementam a funcionalidade de todos eles.

Base de Conhecimento

O HCTT possui um conceito de *modelo* para as mais variadas áreas da aplicação. Estas ferramentas denominam-se por *templates* para: acções, tarefas, missões, etc. A introdução de informação torna-se mais fácil, sendo necessário apenas *importar um template* que já contém informação pré-definida.

Cesto

O *Cesto* constitui um pequeno esquema de comunicação entre utilizadores. Determinadas funcionalidades da aplicação requerem que mensagens sejam enviadas para outros utilizadores. Acedendo a esta ferramenta, o utilizador pode consultar as mensagens que lhe foram enviadas. Funciona como um repositório de tarefas pendentes.

Segurança

Esta ferramenta permitia apenas criar e manter utilizadores. É esta ferramenta que foi complementada ao longo deste estágio, vindo depois a constituir o módulo de segurança, incluindo as funcionalidades avançadas que serão introduzidas no capítulo 3.

2 Tecnologias

Neste capítulo, é feita uma enumeração de todos os aspectos tecnológicos relevantes para o problema que irá ser analisado e solucionado. São apresentadas as linguagens de programação e as ferramentas utilizadas ao longo do processo de desenvolvimento. São feitas também referências às *frameworks* e às tecnologias dos servidores utilizados.

A forte influência da aliança estratégica entre a IBS e a IBM reflecte-se um pouco nas tecnologias utilizadas e nas soluções adoptadas. Esta aliança, que se verifica a um nível mundial, tem efeitos a nível comercial e a nível interno, particularmente ao nível do desenvolvimento.

2.1 Frameworks

O conceito de *framework* é relativamente complexo e poderoso. Uma *framework* é um conjunto de classes que definem uma arquitectura reutilizável. Esta arquitectura, embora abstracta, orienta a solução que deverá ser detalhada e implementada para o problema específico [2][7]. De uma forma geral, o uso de *frameworks* contribui para acelerar o processo de resolução de problemas recorrentes, melhorando a qualidade, fiabilidade, flexibilidade e performance das soluções.

2.1.1 UNIIFace

O UNIIFace é uma *framework* desenvolvida pela própria IBS com o intuito de simplificar e uniformizar a criação de interfaces gráficas. Permite solucionar dois problemas distintos:

- A padronização da interface gráfica através da geração automática de ficheiros JSP⁷.
- Separar a camada de apresentação dos JSPs da parte de lógica JAVA.

A geração das *Java Server Pages* (JSP) é feita com o auxílio da aplicação *GuiGen* (incluída também no “pacote” UNIIFace), a partir de um documento XML (chamado processo). Este documento define todos os elementos gráficos que a página terá (botões, caixas de texto, textos, listas, etc.) assim como os seus comportamentos. Um ficheiro DTD⁸ define tudo o que é possível especificar no documento XML e quais as regras que devem ser seguidas.

⁷ Java Server Pages – tecnologia que permite a geração dinâmica de páginas *web*.

⁸ *Document Type Definition* – define o conteúdo de um documento SGML (Standard Generalized Markup Language) ou XML (Extensible Markup Language).

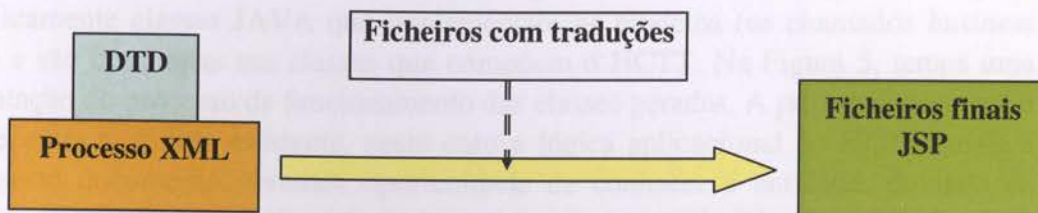


Figura 3 – Geração de JSP a partir de processos XML.

Os JSPs gerados desta forma separam totalmente a parte gráfica da parte de processamento, o que constitui precisamente uma das preocupações principais na arquitectura do HCTT. A *framework* utiliza um conjunto de classes JAVA, às quais se dá o nome genérico de *Adapters*, cuja função é fazer uma interface entre as operações efectuadas nas páginas *web* (carregar em botões, editar campos, processar a informação mostrada, etc.) e todo o processamento implicado na lógica interna da aplicação. Isto quer dizer que o núcleo de lógica de uma determinada aplicação pode ser desenvolvida independentemente da interface gráfica, apoiando apenas nos pressupostos definidos pela *framework* para o posterior acoplamento. Isolando estas duas camadas aplicacionais confere-se flexibilidade à arquitectura para incorporar evoluções ou alterações tecnológicas futuras⁹.

A Figura 4 sintetiza a interacção entre os vários elementos na visualização de uma página:

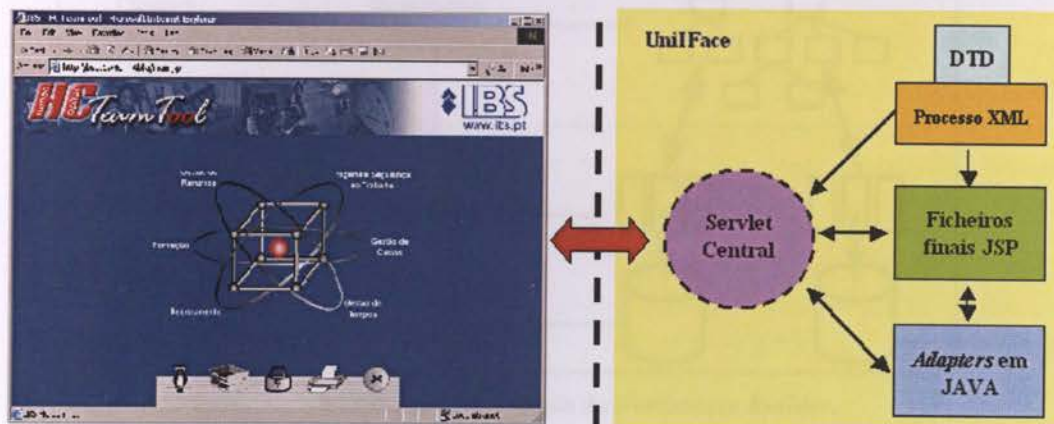


Figura 4 – Esquema do funcionamento da arquitectura UniFace.

2.1.2 Persistence Builder

O Persistence Builder é uma ferramenta integrada no IDE *VisualAge* da IBM que permite gerar a camada de persistência JAVA de forma semi-automática. Através de um

⁹ Se quisermos mudar a interface gráfica para uma outra tecnologia ou apresentação, não precisaremos de mudar a lógica interna da aplicação.

ambiente visual, é possível criar o modelo de classes do problema manualmente, introduzindo toda a informação necessária. Após este passo é possível gerar as respectivas tabelas que permitem guardar a informação numa base de dados relacional, tornando-a persistente. Além das tabelas que são criadas, é possível gerar automaticamente classes JAVA que implementam os modelos (os chamados *business objects*) e são integradas nas classes que compõem o HCTT. Na Figura 5, temos uma representação do processo de funcionamento das classes geradas. A primeira camada é a interface entre o código existente, neste caso a lógica aplicacional do HCTT (mais à frente, neste documento, teremos oportunidade de conhecer a entidade, do lado da aplicação, responsável por esta conexão à camada *persistence builder*, os *controllers*). A segunda camada corresponde às classes que foram geradas a partir do modelo desenhado e que recebem a informação que vem da base de dados. Do ponto de vista da aplicação, estes *business objects* “são” a base de dados. A última camada corresponde a uma outra categoria de classes JAVA, os *service objects*, responsáveis pela ligação física e interacção com a base de dados, através da norma JDBC [16].

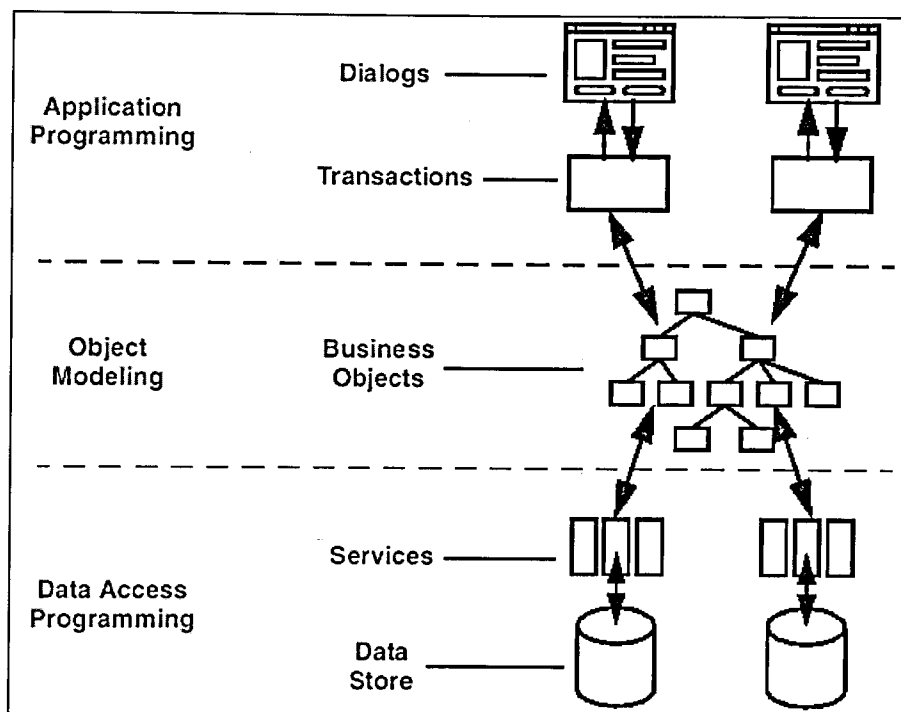


Figura 5 – Camadas arquiteturais do *Persistence Builder*.

Na realidade o Persistence Builder é um conjunto de outras *frameworks* (Figura 6) que resolvem problemas recorrentes no desenvolvimento de *software* de média/grande dimensão com necessidade de persistência. O objectivo é diminuir a quantidade de código escrito e aumentar a robustez e eficiência da aplicação final.

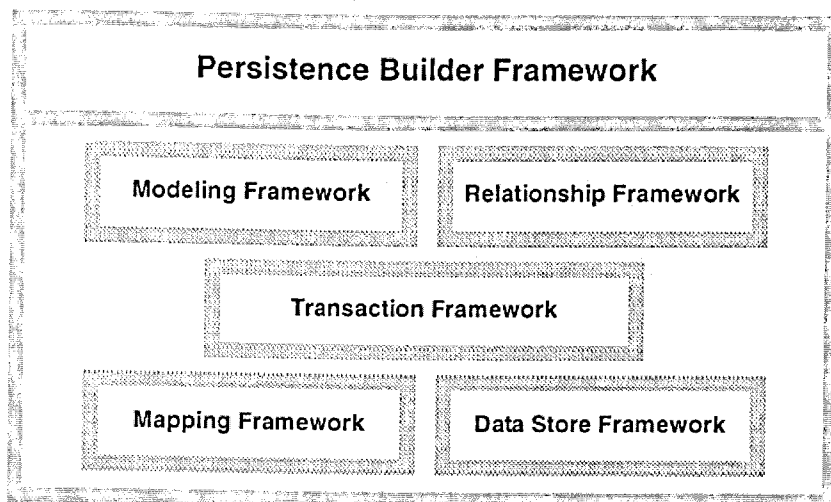


Figura 6 – Frameworks em que se decompõe o Persistence Builder.

Estas *frameworks* auxiliam diferentes etapas do desenvolvimento mas todas elas se debruçam sobre os mesmos problemas: criar uma base de dados que suporte uma aplicação e todos os mecanismos que permitam interagir com ela.

2.2 Linguagens

A linguagem fundamental do HCTT é JAVA, uma linguagem orientada por objectos e com uma crescente popularidade devido ao seu carácter não proprietário, ao contrário da linguagem C#, por exemplo, utilizada na tecnologia .NET. Toda a lógica interna da aplicação é desenvolvida nesta linguagem. Dado que o HCTT é orientado para um ambiente *web* esta linguagem é utilizada convencionalmente e sob formas diferentes:

- **Servlets** - Uma versão JAVA do conceito de *Common Gateway Interface* (CGI) com todas as vantagens inerentes à linguagem JAVA. São programas especialmente concebidos para suportar a criação de páginas web dinâmicas, fornecendo mecanismos para lidar com sessões, *cookies*, etc.
- **JSP's** - Advêm de uma especificação da *Sun Microsystems* para combinar JAVA com HTML de modo a gerar páginas de *internet* dinâmicas. Uma das características principais desta solução é permitir afastar a componente de geração dinâmica (lógica) do aspecto gráfico (*design*).

Embora as páginas *web* sejam geradas pelos JSPs, o conhecimento de HTML e de JavaScript é útil em etapas de análise/resolução de problemas.

Como seria de esperar, para interagir com a base de dados é indispensável o uso de SQL, a linguagem quase universal de interacção com as bases de dados relacionais.

2.3 Ferramentas

Dada a dimensão e também complexidade do projecto são utilizadas algumas ferramentas poderosas que visam apoiar e acelerar a implementação. São utilizadas aplicações para a produção do código, para o controlo de versões dos ficheiros, ferramentas de análise, etc. De seguida apresenta-se uma descrição sumária de cada uma delas.

2.3.1 Ambiente de Desenvolvimento Integrado

Um IDE (*Integrated Development Environment*) é uma aplicação gráfica que integra e potencia várias ferramentas utilizadas durante a codificação de programas (editor de código, compilador, *debugger* [19], programação de interfaces gráficas, etc.). O IDE utilizado foi o Eclipse [12], que constitui uma das mais recentes e promissoras apostas da IBM e outras empresas de software em consórcio. É um IDE bastante revolucionário baseando-se numa arquitectura de *plugins* em que tudo pode ser personalizado e estendido à medida das necessidades das equipas.

2.3.2 Registo de Erros/Plataforma de Colaboração

O registo e análise de erros é executado com o auxílio da plataforma de colaboração Lotus Notes. Utiliza-se uma aplicação implementada especificamente para esta plataforma que constitui uma base de dados onde se regista e se segue de perto os vários estados de cada erro reportado. O Lotus Notes é também utilizado como plataforma de colaboração para, por exemplo, troca de *emails* e marcar reuniões.

2.3.3 Outras ferramentas

Estas ferramentas auxiliam muitas vezes a detecção e resolução de problemas encontrados na aplicação HCTT. Permitem-nos observar a forma resumida e centralizada (p.e. através de gráficos) todos os dados relevantes:

- **JProbe Suite** – Conjunto de ferramentas extremamente completo, que permite uma monitorização da execução de baixo-nível do código JAVA.
- **IronTrack SQL** – É uma ferramenta que captura e analisa toda a actividade na base de dados. É possível monitorizar todas as perguntas SQL enviadas para a BD e respectivos tempos.
- **SQL Profiler** – É uma ferramenta que actua de modo idêntico à anterior complementando-a em certos aspectos específicos.

2.4 Servidores

Dado que a lógica da aplicação é 100% desenvolvida em JAVA, o acesso ao DB2 é feito recorrendo ao *standard* industrial JDBC. Graças a esta tecnologia é possível manter o carácter portátil do JAVA, assim como isolar a aplicação do tipo de base de dados utilizada possibilitando a migração para outros tipos de bases de dados.

O ambiente de desenvolvimento do HCTT é suportado por dois tipos de servidores distintos, que podem ou não estar na mesma máquina. O servidor aplicacional é o *IBM WebSphere Application Server* e o servidor de base de dados é o *IBM DB2/400*.

Por vezes, como as estações de trabalho se baseiam em ambientes *Windows*, é utilizada a tecnologia ODBC para aceder à informação através da aplicação *Microsoft Access*.

No ambiente de desenvolvimento, utiliza-se ainda o servidor *Tomcat* [9]. Este servidor tem as vantagens de ser um projecto *open-source*, ser facilmente integrado no *Eclipse* e ser possível fazer depuração do código [18]. O *Tomcat* é um servidor que é utilizado na implementação de *servlets* e *JSPs* e disponibiliza uma funcionalidade similar à do *IBM WebSphere*, no âmbito do HCTT.

3 O Módulo de Segurança

Nos primeiros dois capítulos, apresentaram-se as três importantes componentes do cenário circundante ao objecto principal do estágio, o módulo de segurança: a empresa IBS, o produto HCTT e as tecnologias utilizadas para a sua implementação. Após este enquadramento inicial prosseguir-se-á para uma definição detalhada do problema que consiste na especificação das características pretendidas para o módulo em questão.

Evidencia-se primeiramente a importância do tema e depois subdivide-se o conceito *segurança* em três aspectos distintos. É dado especial destaque à *segurança de acesso* e à *segurança de execução*, definindo-se conceitos fundamentais como *responsável (ownership)*, *comando*, *grupo* e *perfil de segurança*. Esta primeira secção, que expõe os objectivos do estágio, termina com a manutenção das autorizações e definem-se requisitos para tornar esta tarefa eficaz.

Adicionalmente apresentam-se mais duas secções, com aspectos específicos do estágio curricular, tais como a metodologia da equipa de desenvolvimento e o planeamento de actividades que serviu de suporte à realização de todas as tarefas.

3.1 Objectivos

O HCTT é uma aplicação dotada de um espectro funcional consideravelmente vasto pois a Gestão de Recursos Humanos é um processo complexo e abrange todos os colaboradores de uma empresa, implicando um conjunto bastante diverso de responsabilidades. É uma ferramenta orientada à gestão partilhada de informação sensível, distribuída pelos vários níveis da estrutura empresarial e, como tal, exige uma atenção especial no que respeita ao controlo de acesso a esses dados.

Ao permitirmos restringir o conjunto de operações e acesso à informação pelos utilizadores estamos a fornecer à direcção de Recursos Humanos uma ferramenta transversal, poderosa e eficaz para a correcta divisão de responsabilidades.

Esses mecanismos de segurança subdividem-se em três aspectos distintos: *autenticação*, *segurança de acesso* e *segurança de execução*, cada um com características particulares de implementação técnica e interacção com o utilizador.

O mecanismo de *autenticação* foi já implementado e consiste num esquema clássico de registo de utilizadores e palavras-passe que são depois fornecidas, individualmente, aos colaboradores da empresa. Para aceder ao HCTT, o utilizador necessita de se autenticar, introduzindo o nome do seu utilizador (*username*) e a sua palavra-passe (*password*).

Depois de autorizado a aceder à aplicação, o utilizador terá apenas *acesso* a um conjunto potencialmente restrito de módulos funcionais, áreas da aplicação (*páginas web*) e informação. É este o âmbito da *segurança de acesso*: restringir o acesso a funcionalidades e informação.

A *segurança de execução* refere-se ao controlo da execução das *operações* disponibilizadas pelo HCTT. Na maior parte das situações, existem várias funcionalidades acedidas no mesmo *sítio* da aplicação. Deste conjunto, existem operações que o utilizador pode utilizar e outras que não pode utilizar. Neste caso, teremos que conceder *acesso* à página *web*, e implementar um esquema de autorização que tenha efeitos no momento em que o utilizador *executa* a operação, ou seja, que verifique se o utilizador tem ou não a permissão requerida para a executar.

3.1.1 Segurança de Acesso

A segurança de acesso compreende duas vertentes. A primeira prende-se com a restrição da informação apresentada. Esta restrição apoia-se no conceito de *responsável* (*ownership*) que é utilizado em vários módulos da aplicação. Por exemplo, no módulo de recrutamento, o utilizador só conseguirá visualizar processos de recrutamento pelos quais é responsável; e no módulo de gestão de recursos, o utilizador visualiza apenas as estruturas e nós pelos quais é responsável e todos os seus descendentes.

A segunda vertente refere-se ao controlo de acesso aos módulos e outras áreas funcionais. Na página principal do HCTT, o utilizador selecciona o módulo ao qual deseja aceder. Só poderá utilizar aqueles aos quais tem acesso, caso contrário surgirá um aviso indicando que não tem permissão para aceder a esse módulo. Similarmente, deverá ser possível negar o acesso a determinadas páginas *web* da aplicação, a um ou mais utilizadores.

3.1.2 Segurança de Execução

É necessário termos um outro nível de segurança que actua no momento em que o utilizador tenta efectuar uma operação no HCTT, pois a segurança de acesso não permite controlar operações específicas, mas sim conjuntos de operações¹⁰. Esta questão tinha já sido abordada desde o início do projecto e avançaram-se decisões arquitecturais que permitissem preparar a solução a adoptar no futuro.

Surge então o conceito de *comando*. Qualquer operação que seja possível executar na aplicação, tem uma representação interna lógica única – o *comando*. Mais ainda, quando um utilizador executa determinada operação, essa entidade está necessariamente envolvida. A Figura 7 apresenta um esquema básico de arquitectura e pretende mostrar que o fluxo operacional da aplicação passa sempre pela execução de 1 comando.

¹⁰ Ao impedir o acesso a determinada área, impede a execução das funcionalidades que aí são disponibilizadas.

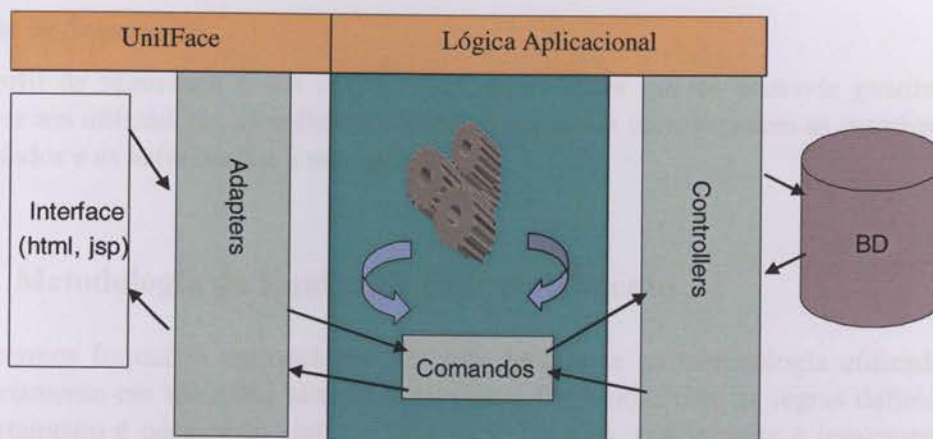


Figura 7 – Diagrama simplificado da arquitectura lógica do HCTT.

Quando um *comando* vai ser utilizado será necessário verificar primeiro se o utilizador tem ou não permissão para *o executar*. De acordo com essa informação prosseguir-se-á ou não com a operação que se iniciou. Se o comando não puder ser executado, é mostrado um aviso ao utilizador na interface gráfica dizendo que não está autorizado a executar a operação.

3.1.3 Gestão/Controlo de Autorizações

A gestão/controlo de autorizações é um problema complexo pois pretende dar resposta a situações nas quais existe um número grande de utilizadores e de comandos. A visualização de informação, no HCTT, é baseada em listas com pouco mais de 20 elementos. Esta característica limitativa da interface gráfica levanta a preocupação de arranjar outras soluções que permitam seleccionar, pesquisar e manipular um grande número de elementos de uma forma rápida e eficaz. De seguida apresentam-se alguns conceitos com vista a fazer uma abordagem a esta questão, avançando-se já com algumas soluções.

Grupos de Utilizadores

Um grupo de utilizadores é um conjunto de utilizadores aos quais se deseja aplicar as mesmas operações em simultâneo. É essencial que a forma de os definir e manter tenha em conta o elevado número de utilizadores da aplicação, para não comprometer a eficácia da solução.

Grupos de Comandos

Paralelamente, um grupo de comandos é um conjunto de comandos, associados de forma a que se possam manipular simultaneamente. O elevado número de comandos obriga a que estes sejam apresentados de uma forma organizada, estruturada e equilibrada.

Perfis de Segurança

O perfil de segurança é um conjunto de autorizações que se pretende guardar e/ou aplicar aos utilizadores da aplicação. Estas autorizações compreendem as autorizações a comandos e as autorizações a módulos.

3.2 Metodologia da Equipa de Desenvolvimento

Em termos formais a metodologia adoptada baseou-se na metodologia utilizada pelo departamento em situações semelhantes a esta. De acordo com as regras definidas no departamento é necessário elaborar uma especificação, previamente à implementação, que é composta pelas seguintes partes:

- **Modelo relacional** – Contém informação sobre todas as alterações ao modelo relacional da BD necessárias para a correcta implementação da funcionalidade.
- **Interfaces** – Pretende-se que sintetize todas as sequências de ecrãs envolvidas no uso da funcionalidade.
- **Descrição funcional** – Contém uma descrição sumária da funcionalidade e para cada parte em que esta se divida, uma análise funcional com ligações para as informações relacionadas dos documentos anteriores.

De uma forma global o esquema da Figura 8 sintetiza a metodologia adoptada:

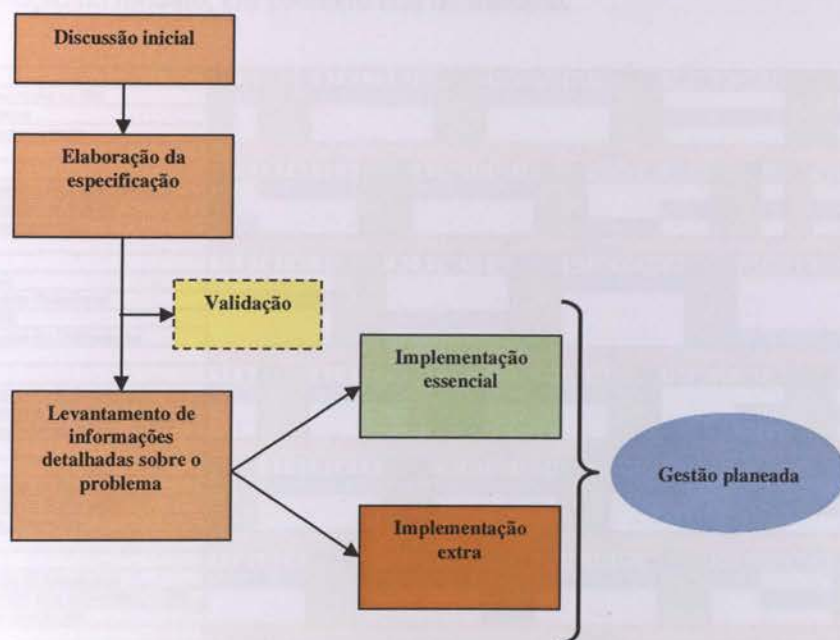


Figura 8 – Metodologia de desenvolvimento adoptada.

3.3 Planeamento de Actividades

Na Figura 9, abaixo apresentada, podemos visualizar o planeamento de actividades, apresentado de uma forma macro. O estágio iniciou-se com um período de adaptação e integração na empresa, na equipa e no projecto que durou cerca de duas semanas. Tomou-se contacto com a metodologia de trabalho da equipa, as tecnologias utilizadas ao longo do processo de desenvolvimento, a arquitectura do HCTT e as suas particularidades técnicas.

Posteriormente, surgem as fases de análise e do desenho da implementação. Antes de iniciar a implementação, efectuou-se um estudo da ferramenta Persistence Builder (já introduzido na secção 2.1.2) e das tecnologias JAAS [13] e JUnit [14]. Estas tecnologias não foram referidas no capítulo 2 pois não estavam a ser utilizadas pelo departamento. O objectivo foi analisar a sua aplicabilidade e relevância na implementação do módulo de segurança.

A implementação é faseada em 4 partes: modelo relacional, grupos de utilizadores, grupos de comandos e perfis de segurança. Antecipou-se a conclusão da autorização de acesso a módulos, apesar de pertencer à última fase, logo após a implementação dos grupos de utilizadores. Isto coincidiu com uma redefinição estratégica de prioridades.

No final foi elaborado um documento – o guião de testes – onde se especificou detalhadamente casos de utilização para serem executados, com o objectivo de testar a aplicação. As funcionalidades foram cuidadosamente testadas e no final procedeu-se à instalação do módulo, em contexto real de trabalho.



Figura 9 – Cronograma de actividades do estágio.

4 Especificação

No capítulo 3, definiu-se o objectivo do módulo de segurança: permitir ao administrador de segurança controlar os acessos à aplicação de um modo eficaz. Neste capítulo apresenta-se a especificação dos requisitos do módulo de segurança do HCTT.

Na secção 4.1, organizam-se todos os casos de utilização de acordo com áreas de funcionalidade, definem-se os actores e as pré-condições comuns. Mencionam-se também requisitos partilhados por todos os casos de utilização.

Os pacotes lógicos são tratados separadamente nas secções seguintes. Em cada uma utiliza-se uma estrutura básica de um modo sistemático composta por uma introdução aos requisitos funcionais do pacote lógico, as pré-condições e a interface gráfica. Para cada caso de utilização, apresenta-se o diagrama UML, as pré-condições e a interface gráfica.

4.1 Generalidades

Podemos observar na Figura 10 que os casos de utilização do módulo de segurança foram agrupados em 4 áreas funcionais. Podemos ainda verificar que a manutenção de grupos de utilizadores depende da manutenção de utilizadores. Não podemos manter grupos de utilizadores, se não tivermos utilizadores. Similarmente, a manutenção de perfis de segurança depende dos grupos de utilizadores e dos grupos de comandos.

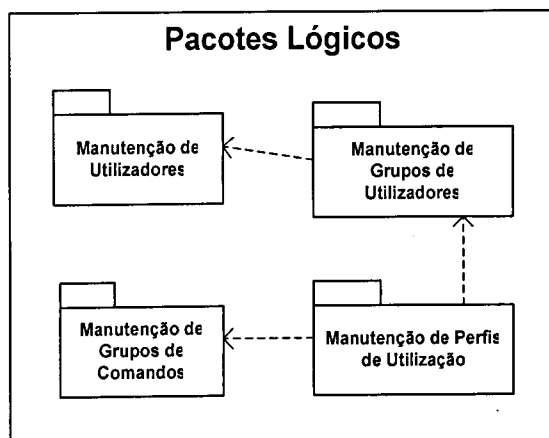


Figura 10 – Organização dos casos de utilização.

Actores

Deverá ser implementado o conceito de utilizador *administrador*, que não tem qualquer restrição de acesso. Este utilizador será o *administrador de segurança* da aplicação HCTT.

Todos os outros utilizadores têm acesso controlado ao módulo e portanto deverão estar autorizados expressamente para executar as operações de segurança.

Para simplificar a apresentação dos casos de utilização, não se faz a distinção entre estes actores: utilizar-se-á sempre o *administrador de segurança*.

Pré-condições dos casos de utilização

Na página inicial do módulo de segurança, torna-se evidente a divisão lógica ilustrada na Figura 10, pois coincide precisamente com a divisão funcional apresentada ao utilizador.

Todos os casos de utilização do módulo têm pré-condições comuns. É sempre necessário o utilizador autenticar-se (processo de *login*) e depois seleccionar o módulo de segurança, quando lhe é apresentada a página principal da aplicação. Ao entrar neste módulo, é apresentada a página da Figura 11.

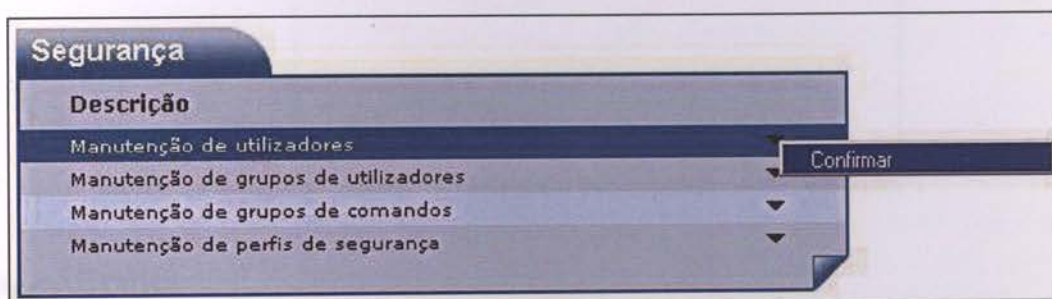


Figura 11 – Página web principal (inicial) do módulo de segurança.

Navegação em listas

Nas páginas onde se apresentam listas de elementos, é necessário disponibilizar um mecanismo de pesquisa e de navegação na lista. Este mecanismo – o navegador genérico do HCTT [15] – está já parcialmente implementado e deverá ser utilizado.

4.2 Manutenção de Utilizadores

A pré-condição deste caso é seleccionar a opção 1 – Manutenção de utilizadores – na página principal do módulo de segurança apresentada na Figura 11.

Este pacote de casos de utilização inclui funcionalidade que permite criar e eliminar utilizadores, bem como editar a sua informação. Desta informação destaca-se o

colaborador da empresa associado ao utilizador, o nível de acesso à aplicação e a *password*.

A Figura 12 apresenta estes casos de utilização em diagrama UML. Este conjunto de funcionalidades estava já implementado no momento deste estudo de especificação.

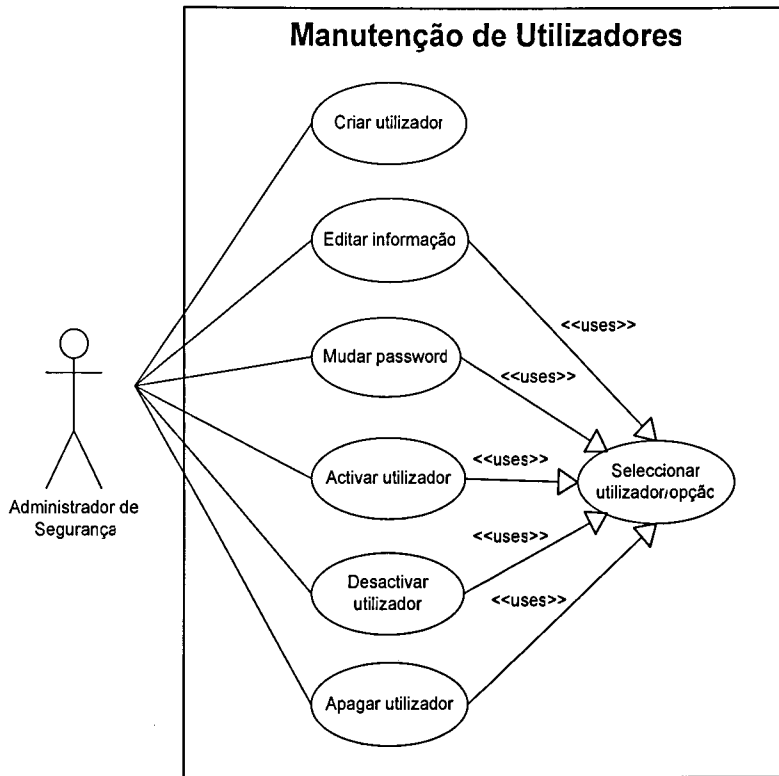


Figura 12 – Pacote de casos de utilização “Manutenção de Utilizadores”.



Interface Gráfica

A Figura 13 apresenta a página da manutenção de utilizadores. O caso de utilização *seleccionar utilizador/opção*, que foi definido no diagrama, corresponde a escolher operações da linha do utilizador correspondente. Na Figura 13, foi seleccionado o utilizador BCOSTA1 (1.^a linha da lista). Surgiram depois, em *context-menu*¹¹, as várias opções possíveis.

¹¹ As *acções de context-menu* são um *standard* gráfico correntemente utilizado na aplicação HCTT. Representam operações sobre elementos de listas. Vejamos o exemplo da Figura 13: ao utilizar o botão do lado direito do rato sobre uma linha da lista surge-nos, num *context-menu*, diversas operações que recebem esse elemento da lista como parâmetro.

Manutenção de utilizadores

✓ Utilizador	Nível	Local	Estado	
BCOSTA1	10	Português (Standard)	Activo	Abrir Mudar password Apagar Activar Desactivar
BCOSTA2	10	Português (Standard)	Activo	
UTILIZADOR_TESTE	10	Português (Standard)	Activo	
CNEVES	0	Português (Standard)	Activo	
NOGUEIRA	10	Português (Standard)	Activo	
FROJO	10	Português (Standard)	Activo	
LUÍS	10	Português (Standard)	Activo	
CELIO	0	Português (Standard)	Activo	
RBRITO	10	Português (Standard)	Activo	
EDISON*	0	Português (Standard)	Activo	
DMARTINHO	0	Português (Standard)	Activo	
BCARDOSO	0	Inglês (EUA)	Activo	
LCASTRO	0	Português (Standard)	Activo	
LA	0	Português (Standard)	Activo	
TEIXEIRA	10	Português (Standard)	Activo	

 Criar

Figura 13 – Página web da manutenção de utilizadores.

4.3 Manutenção de Grupos de Utilizadores

Se recordarmos a organização dos casos de utilização definida na Figura 10, podemos identificar um outro conjunto – o da manutenção de grupos de utilizadores. Este pacote lógico foi sub-dividido em 2 sub-pacotes que se apresentam separadamente em diferentes secções:

- secção 4.3.1 *Adicionar/Remover Utilizadores;*
- secção 4.3.2 *Criar/Apagar Grupos.*

Estes casos de utilização partilham de requisitos e pré-condições comuns.

Pré-condições

Para manter grupos de utilizadores, é necessário escolher a opção 2 da página inicial do módulo (Figura 11).

Requisitos

Recordemos os requisitos definidos de um modo informal na secção 3.1.3 *Gestão/Controlo de Autorizações* (p.22). Após uma análise cuidada do problema,

definiram-se os requisitos, que compreendem três formas distintas de adicionar utilizadores a um grupo, com base:

- na lista dos utilizadores registados no HCTT;
- na estrutura orgânica da empresa;
- nos grupos de utilizadores já definidos.

Deverá ser ainda possível:

- criar grupos de utilizadores;
- apagar grupos de utilizadores;
- remover utilizadores de um grupo;
- remover todos os utilizadores de um grupo.

4.3.1 Adicionar/Remover Utilizadores

A opção de *manutenção de grupos de utilizadores* da página inicial da segurança (Figura 11) conduz-nos à página *web* que reúne todas as funcionalidades respectivas (Figura 15). Depois de seleccionarmos um grupo de utilizadores na árvore lateral, podemos adicionar e remover utilizadores, como mostra o diagrama da Figura 14, abaixo apresentado.

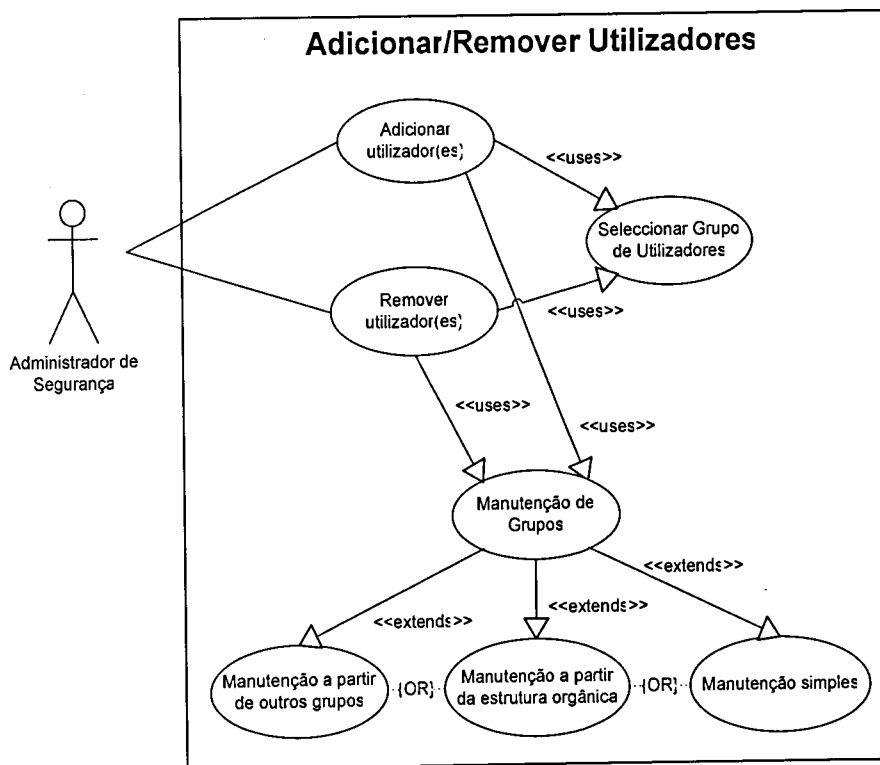


Figura 14 – Pacote de casos de utilização “Adicionar/Remover Utilizadores”.

Existem três formas distintas possíveis para adicionar e remover utilizadores. Estas opções estão ilustradas na Figura 15:

- *Lista de todos os utilizadores* (está designada no diagrama de casos de utilização como *Manutenção simples*) – **opção 1** – surge uma lista com todos os utilizadores registados na aplicação, excepto aqueles que já estão incluídos no grupo ao qual estamos a adicionar. Esta lista não tem uma divisão estrutural, está apenas ordenada crescentemente pelo campo “Utilizador”. Ilustra-se na Figura 16 uma destas listas.
- *Estrutura orgânica* – **opção 2** – podemos seleccionar os utilizadores a partir da árvore da estrutura orgânica da empresa. Ao seleccionarmos um nó da árvore¹², visualizamos todos os utilizadores associados a colaboradores daquele nó. Para cada utilizador, surge a informação se está ou não incluído no grupo que estamos a manter. Esta informação é definida formalmente por *informação de inclusão* (Figura 17). Neste caso podemos verificar na coluna *Incl.?* que nenhum utilizador pertence ao grupo.
- *Outros grupos de utilizadores* – **opção 3** – à semelhança da funcionalidade anterior, aqui também é possível consultar a lista de todos os utilizadores da aplicação com o auxílio de uma divisão estrutural; com efeito, a opção manter a partir de grupos permite-nos navegar na árvore de todos os grupos de utilizadores (excepto o que estamos a manter) e consultar os utilizadores que pertencem a cada um. Nesta lista, tal como no ponto anterior, é possível consultar a composição de cada grupo e adicionar e remover utilizadores a partir dela. Mais uma vez, podemos contar com a ajuda da informação de inclusão no grupo que estamos a manter (Figura 18).

Interface Gráfica

A Figura 15 mostra a página de manutenção de grupos de utilizadores. Nesta página, e relacionado com o caso de utilização que estamos a estudar, temos 3 opções distintas para adicionar e remover utilizadores.

Se optarmos pela manutenção simples (opção 1), surge a lista dos utilizadores da aplicação para podermos adicionar ao grupo que estamos a manter (Figura 16). Na página ilustrada seleccionaram-se alguns utilizadores para adicionar (visível pelos *vistos*¹³ laterais), *clickando* depois no botão “Adicionar utilizadores” para concluir a operação.

¹² A *árvore* é uma estrutura de dados frequentemente utilizada no domínio da informática composta por nós e níveis. Cada nó tem 1 ou mais *filhos* que ocupam o nível $n+1$ do *pai*. Esta é uma definição simplista do conceito de árvore [10].

¹³ Estas operações que aceitam múltiplos elementos da lista como parâmetros são sempre implementadas desta forma, permitindo seleccionar os elementos da lista e depois utilizando botões que são visíveis na base da lista.

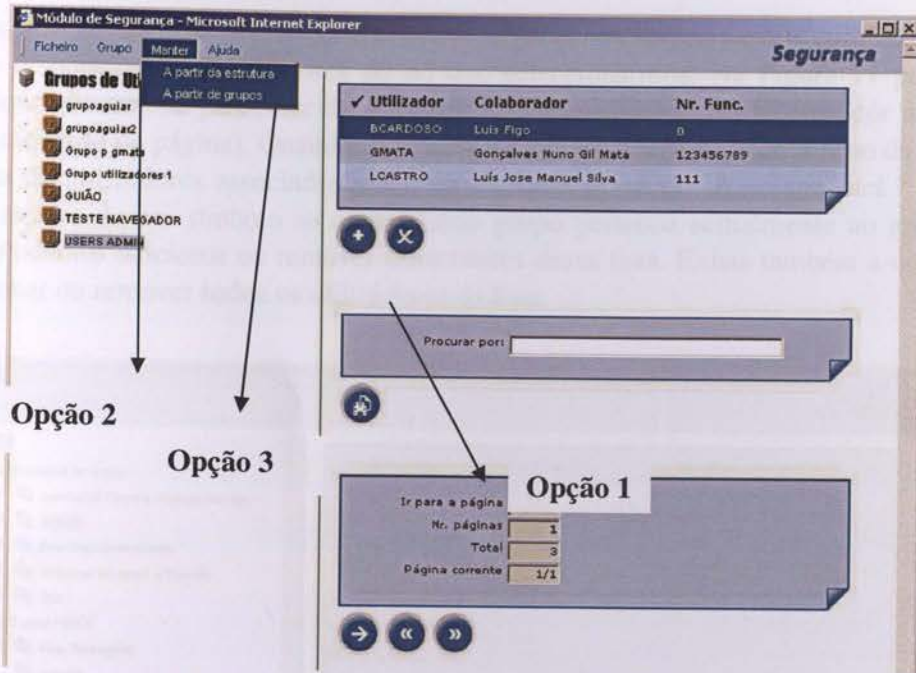


Figura 15 – Página web da manutenção de grupos de utilizadores.

✓ Utilizador	Colaborador	Nr. Func.
SDFS FDGDFG	Ana Sofia Pereira	0
ADMIN	Luis Jose Manuel Silva	111
AGUIAR	Luis Jose Manuel Silva	111
AGUIAR1	Cidália Andrea Neves	1234567
AGUIAR2	Joaquim lebria	0
AGUIAR3	Luis Jose Manuel Silva	111
AGUIAR4	leandro lalalalalala castro	0
BCARDOSO2	Não definido	Não definido
BCOSTA	Frank II Sinatra	0
BCOSTA1	Denzel Washington	0
BCOSTA2	Al Pacino	0
CAMORIM	celso rodrigo1	0
CELIO	Não definido	Não definido
CIDÁLIA	Não definido	Não definido
✓ CIDÁLIA ANDREA NEVES	Cidália Andrea Neves	1234567
CLAUDIA	Al Pacino	0
✓ CNEVES	joana Manuel Couto	12312312
DMARTINHO	Nicholas Cage	0
EDISON*	cidalia neves	0
✓ FROJO	Gonçalves Nuno Gil Mata	123456789
✓ GMATA2	Jose Manuel da Silva joão	11
GMATA3	Gonçalves Nuno Gil Mata	123456789
✓ JOANA	cidalia neves	0

Adicionar utilizadores

Figura 16 – Manutenção simples de grupos de utilizadores.

A opção 2 dá-nos uma vista sobre a estrutura orgânica da empresa e dos utilizadores que estão associados a colaboradores do nó que seleccionarmos. Na Figura 17 podemos verificar que um nó particular da estrutura está seleccionado (texto com cor azul, no lado esquerdo da página). Quando se selecciona um nó, surge do lado direito da página a lista de utilizadores associados aos colaboradores desse nó. A coluna *Incl.?* indica-nos, através de um símbolo se determinado grupo pertence actualmente ao grupo ou não. Podemos adicionar ou remover utilizadores dessa lista. Existe também a opção de adicionar ou remover **todos** os utilizadores da lista.

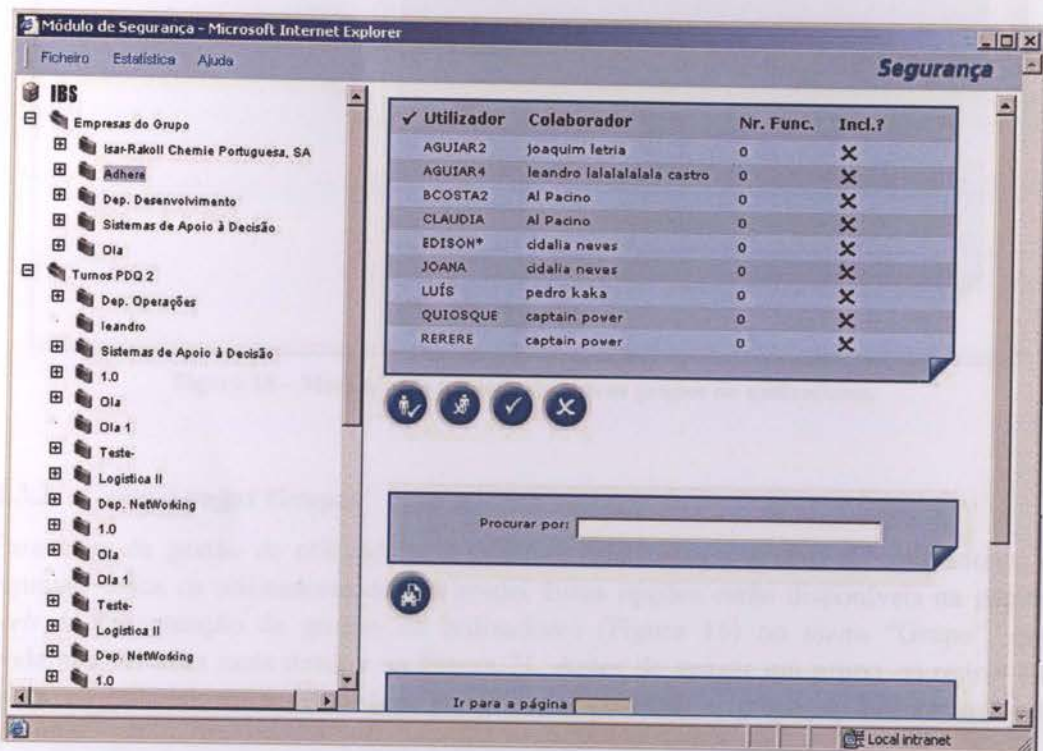


Figura 17 – Manutenção a partir da estrutura orgânica da empresa

A última opção permite-nos seleccionar utilizadores a partir dos outros grupos de utilizadores. Como podemos verificar na Figura 18 surge-nos uma lista quando seleccionamos um nó da árvore lateral (esta árvore é agora de grupos de utilizadores e não a estrutura orgânica da empresa). Da mesma forma, podemos adicionar e remover utilizadores, seleccionados da lista, a um grupo (**todos** ou só alguns).

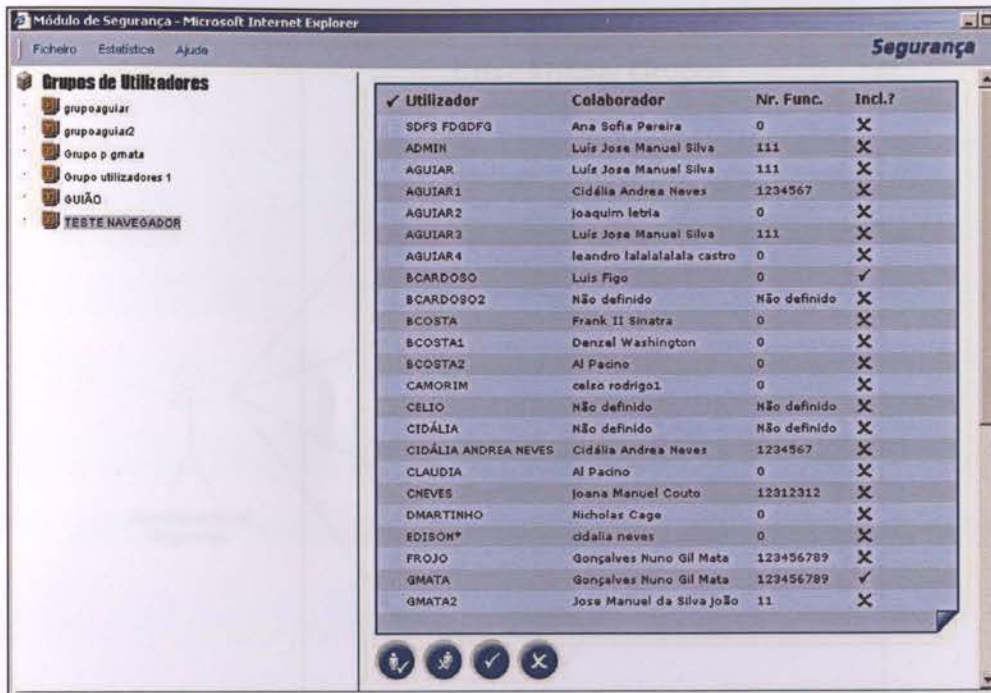


Figura 18 – Manutenção a partir de outros grupos de utilizadores.

4.3.2 Criar/Apagar Grupos

Para além da gestão de utilizadores é possível criar, apagar grupos de utilizadores, e remover todos os utilizadores de um grupo. Estas opções estão disponíveis na página *web* de manutenção de grupos de utilizadores (Figura 15) no *menu* “Grupo”, que podemos ver com mais detalhe na Figura 21. Antes de apagar um grupo ou retirar-lhe todos os utilizadores é necessário, primeiro, seleccionar o grupo de utilizadores em questão.

O diagrama UML da Figura 19 apresenta formalmente este pacote de casos de utilização.



Figura 19 – Pacote de casos de utilização do sistema de manutenção de utilizadores.

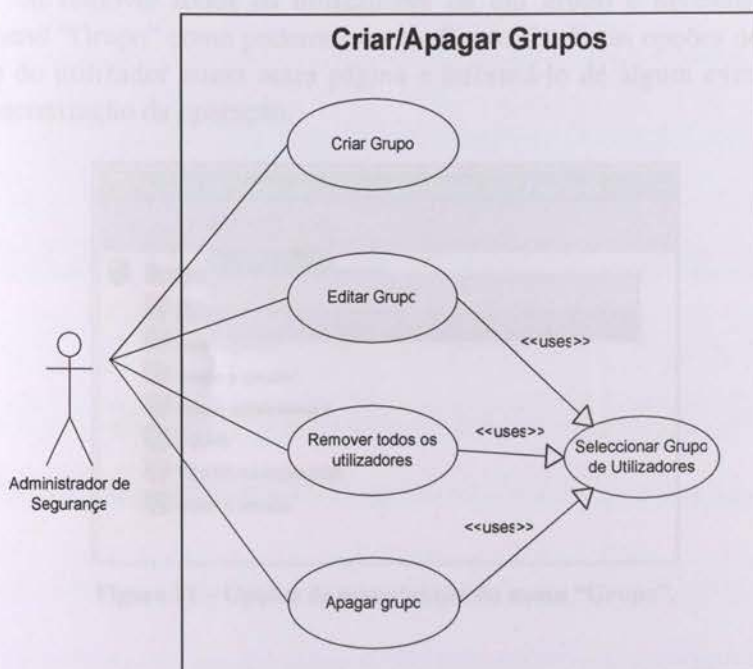


Figura 19 – Pacote de casos de utilização “Criar/Apagar Grupos” (de utilizadores).

Interface Gráfica

A página de criação de grupos de utilizadores (Figura 20) está acessível através da página de manutenção de grupos de utilizadores (Figura 15), no *menu* “Grupo”, através da opção de *menu* “Criar”. Para criação de um grupo de utilizadores, é necessário introduzir obrigatoriamente um nome que o identifica e, opcionalmente, uma descrição. As opções disponibilizadas são *confirmar* e *cancelar*. Ambas as opções permitem voltar à página inicial.

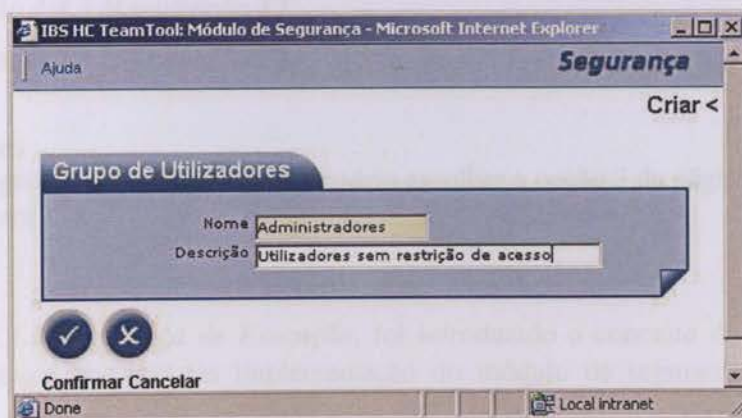


Figura 20 – Página web de criação de um grupo de utilizadores.

Para apagar ou remover todos os utilizadores de um grupo é necessário utilizar as opções do menu “Grupo” como podemos ver na Figura 21. Estas opções deverão pedir a confirmação do utilizador numa outra página e informá-lo de algum eventual erro que impeça a concretização da operação.

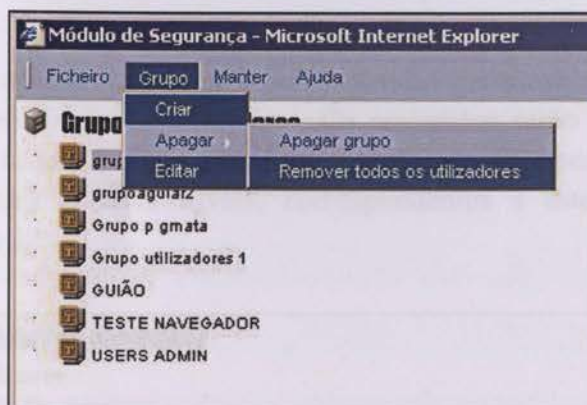


Figura 21 – Opções de manutenção do menu “Grupo”.

4.4 Manutenção de Grupos de Comandos

A manutenção de grupos de comandos, similarmente à manutenção de grupos de utilizadores, compreende a criação e a eliminação do “objecto” grupo de comandos, funcionalidade que é descrita na secção 4.4.1 *Criar/Apagar Grupos*.

Por outro lado, temos também a adição e a remoção de comandos dos diferentes grupos. Tal como vimos anteriormente, existe uma diversidade funcional para adicionar e remover comandos de um grupo. Os três tipos de manutenção são apresentados em três casos de utilização separados:

- secção 4.4.2 *Manutenção Isolada*;
- secção 4.4.3 *Manutenção XY*;
- secção 4.4.4 *Manutenção YX*.

Pré-condições

Para manter grupos de comandos é necessário escolher a opção 3 da página inicial do módulo (Figura 11).

Requisitos

Na secção 3.1.2 *Segurança de Execução*, foi introduzido o conceito de comando e o papel importante que terá na implementação do módulo de segurança. O primeiro

requisito da manutenção de grupos de comandos obrigam os comandos a estarem registados na base de dados da aplicação¹⁴.

Na 3.1.3 *Gestão/Controlo de Autorizações* mencionou-se a limitação da interface gráfica que se utiliza no HCTT, surgindo a preocupação de arranjar uma forma eficaz para manipular o conjunto numeroso de comandos existente. Na tentativa de minimizar esta limitação, optou-se por uma solução de interface que tem já vindo a ser utilizada com sucesso no HCTT: as árvores. Um destes exemplos foi já mencionado (secção 4.3.1, p.29).

Ainda assim, só uma divisão equilibrada de comandos permitirá reunir as condições para uma navegação eficaz no conjunto total. Os comandos serão agrupados segundo três parâmetros: área funcional, sub-área funcional e tipo de operação. A *árvore de todos os comandos* terá então 3 níveis, correspondentes a estes parâmetros, que podemos ver na Figura 22.

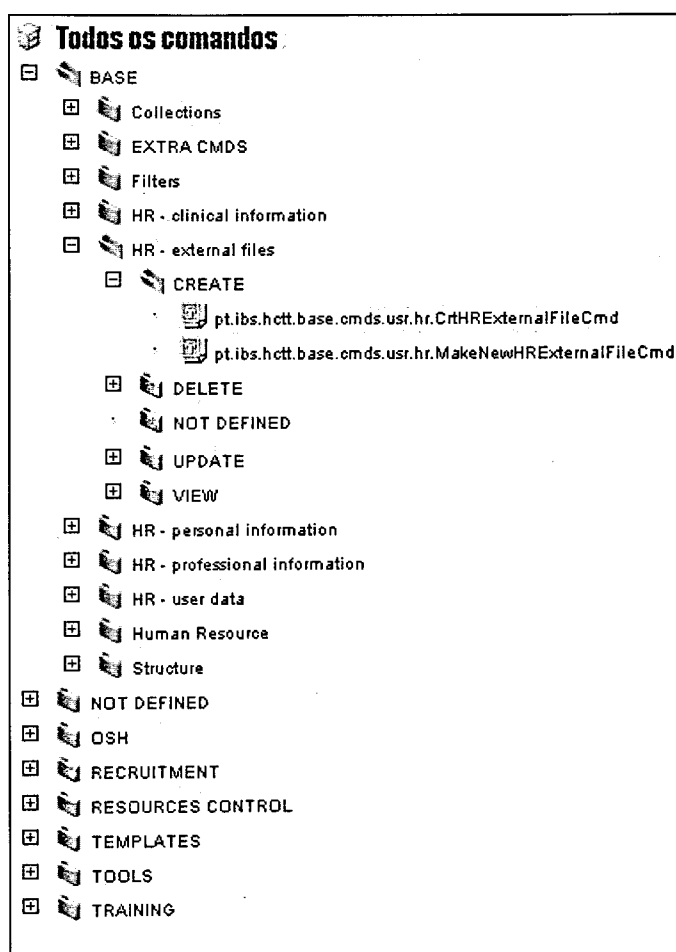


Figura 22 – Árvore de todos os comandos (vista sub-área/tipo, com nós vazios).

Uma boa classificação de comandos é um requisito muito importante. O número de áreas e sub-áreas funcionais deverá ser em número suficiente para garantir um número

¹⁴ A sua organização, que de seguida se introduz, também deverá estar registada.

tratável de comandos em cada ramo da árvore¹⁵, mas estas divisões deverão ter necessariamente um significado funcional adequado¹⁶.

Além disto, é introduzido também um conceito de *vistas*¹⁷ sobre esta árvore. O primeiro nível, é sempre a *área funcional*. O segundo e terceiros níveis podem ser ou a sub-área funcional ou o tipo de operação, mas não os dois ao mesmo tempo. Adicionalmente têm também a opção de os nós vazios aparecerem ou não. Isto quer dizer que a árvore pode ser visualizada de quatro formas diferentes, ou *vistas*: vista sub-área/tipo, com nós vazios; vista tipo/sub-área, sem nós vazios; vista sub-área/tipo, sem nós vazios; vista tipo/sub-área, com nós vazios. Na Figura 22, visualizámos a *árvore de todos os comandos* na 1.^a vista: *sub-área/tipo, com nós vazios*. Vejamos agora as diferenças para a vista *tipo/sub-área, sem nós vazios*, na Figura 23. O nó NOT DEFINED já não aparece, pois não continha qualquer comando e os níveis 2 e 3 são agora *tipo de comando* e *sub-área*, respectivamente.

As operações de manutenção do objecto *comando* não serão disponibilizadas pela aplicação HCTT. Estas operações típicas - criação, eliminação e modificação – são substituídas pela manipulação directa da informação que está na base de dados. Esta operação será uma responsabilidade da IBS. Sempre que é criado um comando novo, é necessário portanto registá-lo na base de dados e tipificá-lo, ou seja, atribuir-lhe uma posição na *árvore de todos os comandos*. Este processo é muito sensível e complexo porque este repositório está em constante mudança¹⁸. Deverão ser compiladas regras sistemáticas para garantir a integridade da informação.

Além da *árvore de todos os comandos*, que oferece um modo de seleccionar comandos para adicionar a um grupo de comandos (a partir do conjunto total de comandos existentes), é necessário também duas outras formas de manutenção, para conferir mais flexibilidade à manutenção de grupos. Uma que permita a manutenção de um grupo a partir do conteúdo dos outros grupos e outra que possibilite a manutenção dos outros grupos a partir do conteúdo de um grupo específico. Nas secções 4.4.3 e 4.4.4, serão avançados detalhes para estes dois requisitos.

¹⁵ O objectivo é uma árvore equilibrada, sem ramos que contenham um número “gigantesco” de comandos.

¹⁶ Se forem criadas divisões sem significado funcional, apenas com o objectivo de se obter ramos com poucos comandos, a pesquisa de comandos não é facilitada. Ao visualizar o nome da área ou sub-área, o utilizador deverá ter logo uma ideia aproximada do conteúdo do nó.

¹⁷ O conceito de *vista* significa uma forma diferente de visualizar uma mesma entidade ou objecto dando ênfase a determinadas características.

¹⁸ Em conjunto com a constante mudança da informação sobre os comandos, o facto de ela estar dispersa em 4 áreas – especificação, classes Java, um ficheiro Excel com a classificação de todos os comandos e scripts SQL de actualização da base de dados – levanta um problema de sincronização delicado.

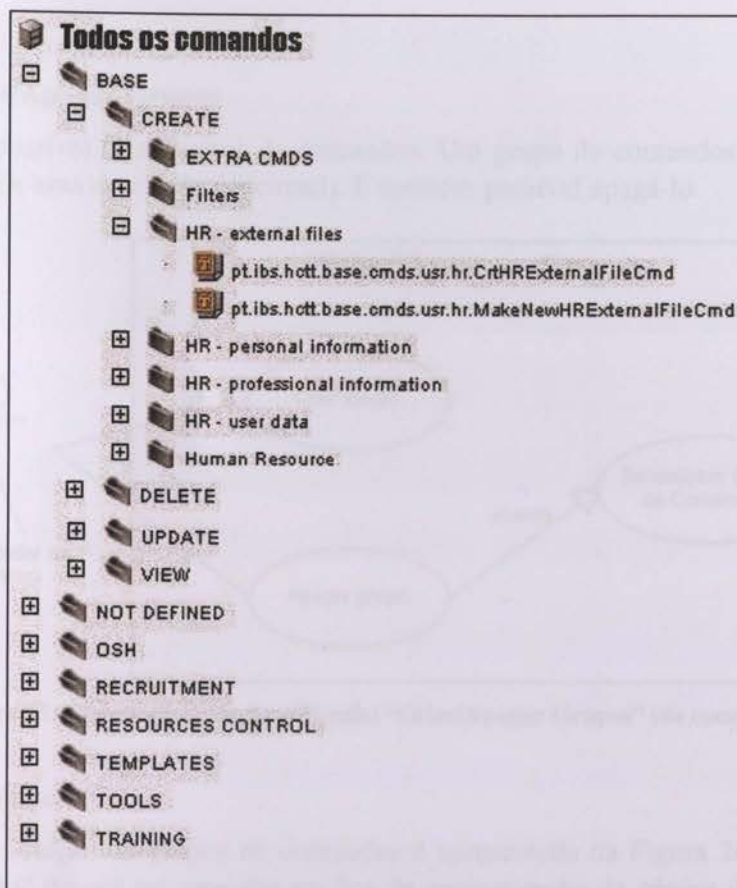


Figura 23 – Árvore de todos os comandos (vista tipo/sub-área, sem nós vazios).

Interface gráfica

Na página de manutenção de grupos de comandos apresenta-se uma lista com todos os grupos de comandos e, em *context-menu*, as diferentes operações de manutenção. Disponibiliza-se também a opção de *criar grupo de comando*, num botão no canto inferior esquerdo, como se pode ver na Figura 24.

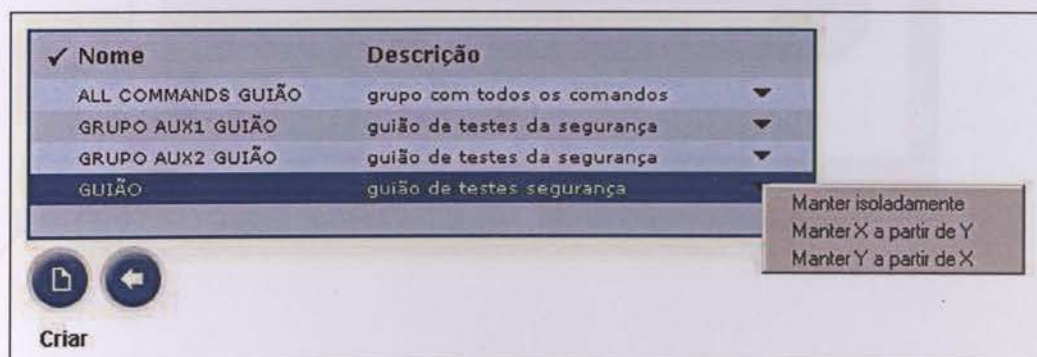


Figura 24 – Página web de manutenção de grupos de comandos.

4.4.1 Criar/Apagar Grupos

Deverá ser possível criar grupos de comandos. Um grupo de comandos tem um *nome* (obrigatório) e uma *descrição* (opcional). É também possível apagá-lo.

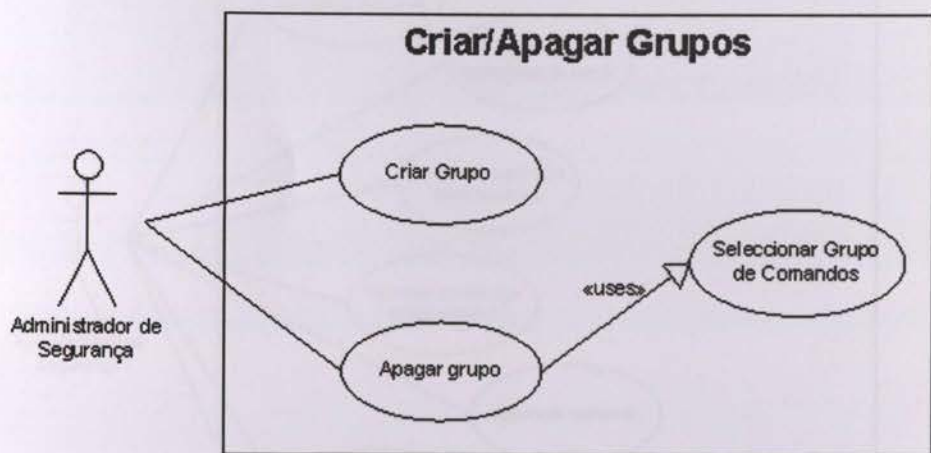


Figura 25 – Pacote de casos de utilização “Criar/Apagar Grupos” (de comandos).

Interface Gráfica

A página de criação de grupos de comandos é apresentada na Figura 26. A opção de “apagar grupo” deverá ser uma das opções de *context-menu* da página de manutenção de grupos de comandos (Figura 24), além das três apresentadas.

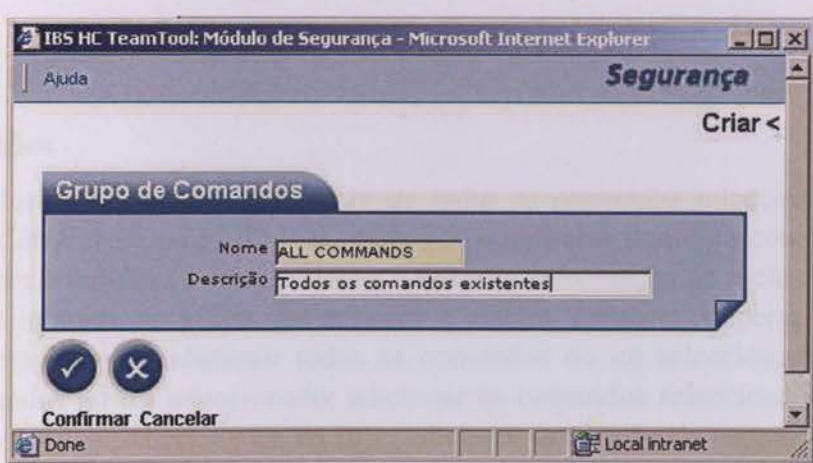


Figura 26 – Página web de criação de grupos de comandos.

4.4.2 Manutenção Isolada

Ao seleccionarmos a opção manutenção isolada da página que se visualiza na Figura 24, podemos navegar na *árvore de todos os comandos*, anteriormente referida, tendo acesso a diversas opções que se encontram ilustradas na Figura 27.

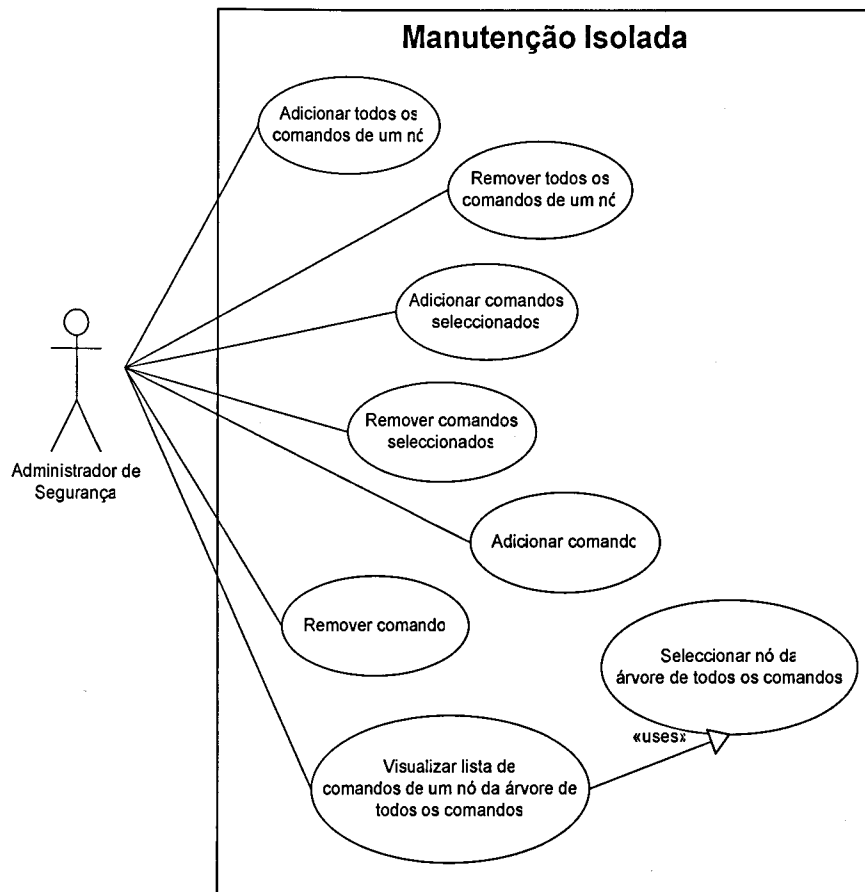


Figura 27 – Pacote de casos de utilização “Manutenção Isolada”.

Interface Gráfica

A Figura 28 ilustra um dos nós da *árvore de todos os comandos* seleccionados, o nó *Base/EXTRA CMDS/UPDATE*. Na lista, podemos observar o nome do comando, a sua descrição, a área e sub-área à qual pertence e também informação de inclusão, ou seja, se está ou não incluído no grupo que estamos a manter. Existem 6 operações básicas disponíveis nesta página: adicionar todos os comandos do nó seleccionado; remover todos os comandos do nó seleccionado; adicionar os comandos seleccionados da lista; remover os comandos seleccionados da lista; adicionar um comando (opção de *context-menu*); remover um comando (opção de *context-menu*). Podemos também verificar na Figura 28 que a navegação na lista dos comandos do nó seleccionado é auxiliada por um navegador e um pesquisador.

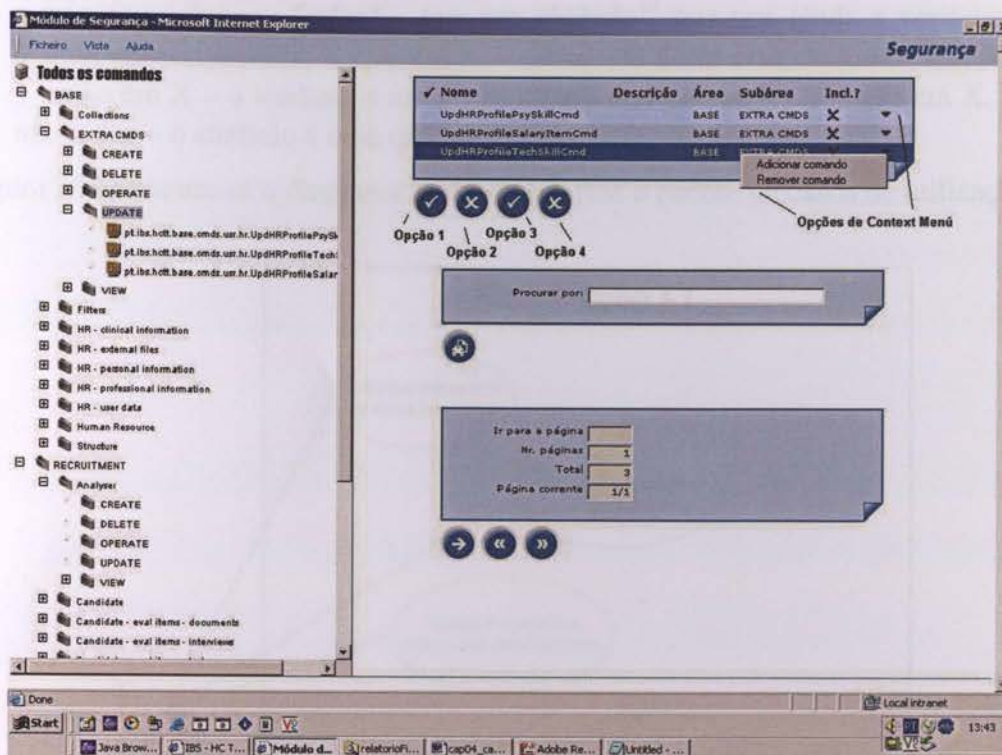


Figura 28 – Página web de manutenção isolada de grupos de comandos.

4.4.3 Manutenção XY

O conceito de manutenção XY define-se pela alteração de um determinado grupo (grupo X) a partir dos comandos de outros grupos já definidos (grupo Y). Nesta situação, navegamos não pela *árvore de todos os comandos* mas sim pela *árvore dos comandos do grupo* que estamos a manter. Esta árvore tem o mesmo formato e estrutura da árvore de todos os comandos, como podemos ver na Figura 30.

A diferença reside no resultado da selecção de um nó da árvore. Anteriormente, quando seleccionávamos um nó, surgia a lista de comandos daquele nó. Agora aparece a lista dos grupos de comandos que contêm comandos desse nó, excepto aquele que estamos a manter. Na Figura 30, podemos ver que o nó BASE/HR-EXTERNAL FILES está seleccionado. Na parte direita da página teremos então *todos os grupos que contêm comandos desse nó*. Aí, e para cada grupo que aparece na lista (grupo Y), podemos obter a seguinte informação:

- considerando apenas os comandos deste nó, quantos comandos estão incluídos em Y – coluna *Total*;
- quantos comandos de a) estão incluídos no grupo que estamos a manter (grupo X) – coluna *Incl.*;

- c) quantos comandos de a) não estão incluídos no grupo que estamos a manter (grupo X) – coluna *Faltam*;
- d) a primeira coluna – *Todos?* – tem um símbolo¹⁹ que nos ajuda a verificar se: nenhum dos comandos Y está em X – o símbolo é um *cruz*; todos os comandos Y estão em X – o símbolo é um *visto*; alguns dos comandos Y estão em X, mas não todos – o símbolo é uma *cruz* e um *visto*.

Na Figura 29 apresenta-se o diagrama UML que define o pacote de casos de utilização.

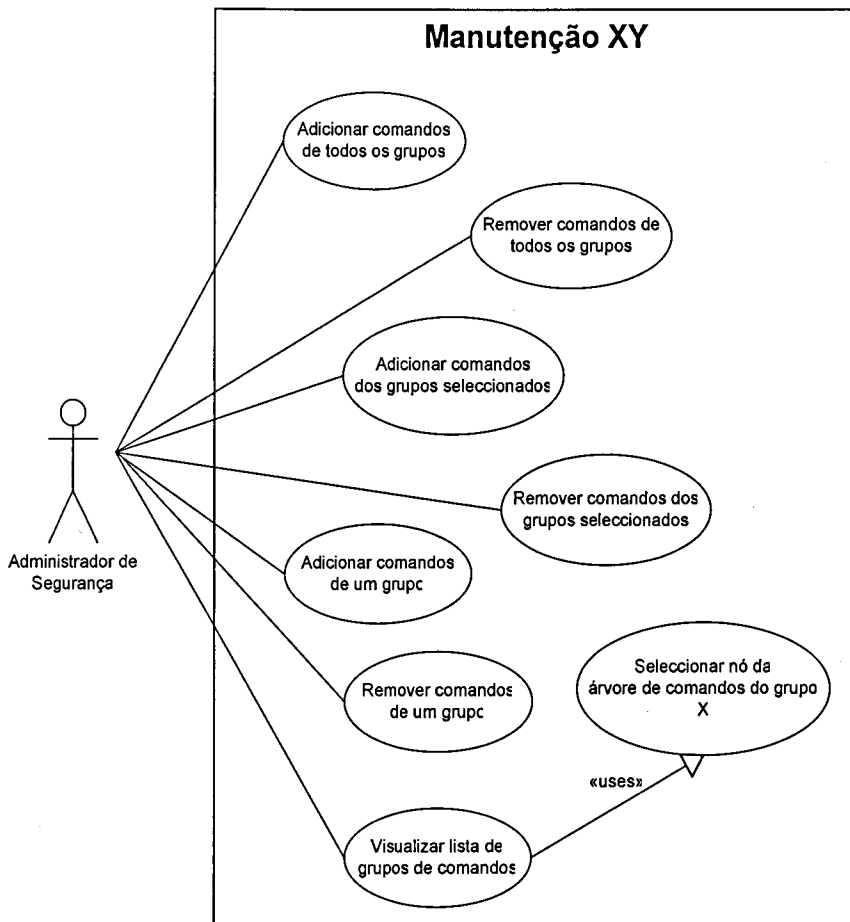


Figura 29 – Pacote de casos de utilização “Manutenção XY”.

Interface Gráfica

As operações disponibilizadas são análogas às apresentadas no caso anterior, mas agora os elementos da lista são *grupos de comandos*, como já vimos. Estas operações consideram **sempre** os comandos do nó seleccionado. Quando dizemos que estamos a adicionar comandos do grupo Y, estamos na verdade a adicionar todos os *comandos de Y* que *pertencem* ao nó da árvore seleccionado. As operações são as seguintes: opção 1 – adicionar comandos de todos os grupos; opção 2 – remover comandos de todos os

¹⁹ É um conceito similar à *informação de inclusão*, introduzido inicialmente na página 31.

grupos; opção 3 – adicionar comandos dos grupos seleccionados; opção 4 – remover comandos dos grupos seleccionados; opção 5 (*context-menu*) – adicionar comandos de um grupo; opção 6 (*context-menu*) – remover comandos de um grupo.

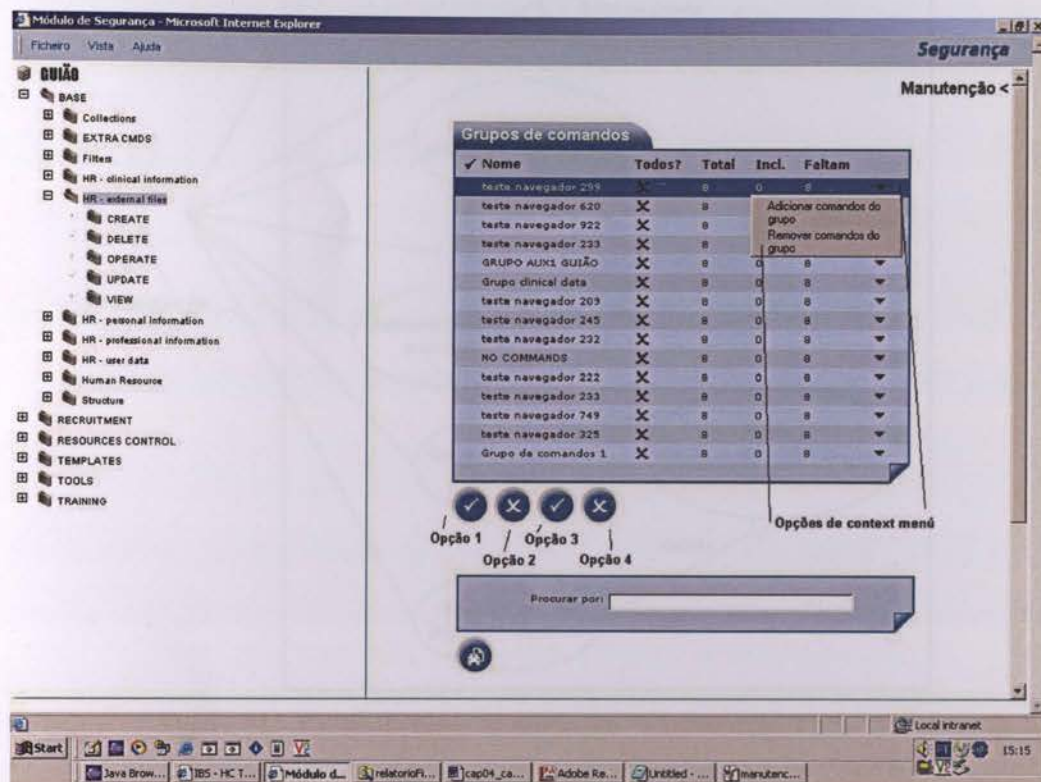


Figura 30 – Página web de manutenção XY de um grupo de comandos.

4.4.4 Manutenção YX

Este conceito é análogo ao da manutenção XY, mas agora modifica-se a composição de outros grupos (grupo Y) a partir dos comandos de um determinado grupo (grupo X). Para cada nó seleccionado, surge-nos uma lista com **todos os grupos de comandos excepto aquele que estamos a manter** (Figura 32). Para cada grupo que aparece na lista (grupo Y) aparece também *informação de inclusão* de comandos:

- do total de comandos deste nó, quantos comandos estão incluídos em X – coluna *Total*;
- quantos comandos de a) estão incluídos em Y – coluna *Incl.*;
- quantos comandos de b) **não** estão incluídos em Y – coluna *Faltam*;
- a coluna *Todos?* contém o *símbolo de inclusão* e é análoga à anterior (secção 4.4.3 alínea d).

O diagrama UML do pacote de casos de utilização apresenta-se de seguida (Figura 31).

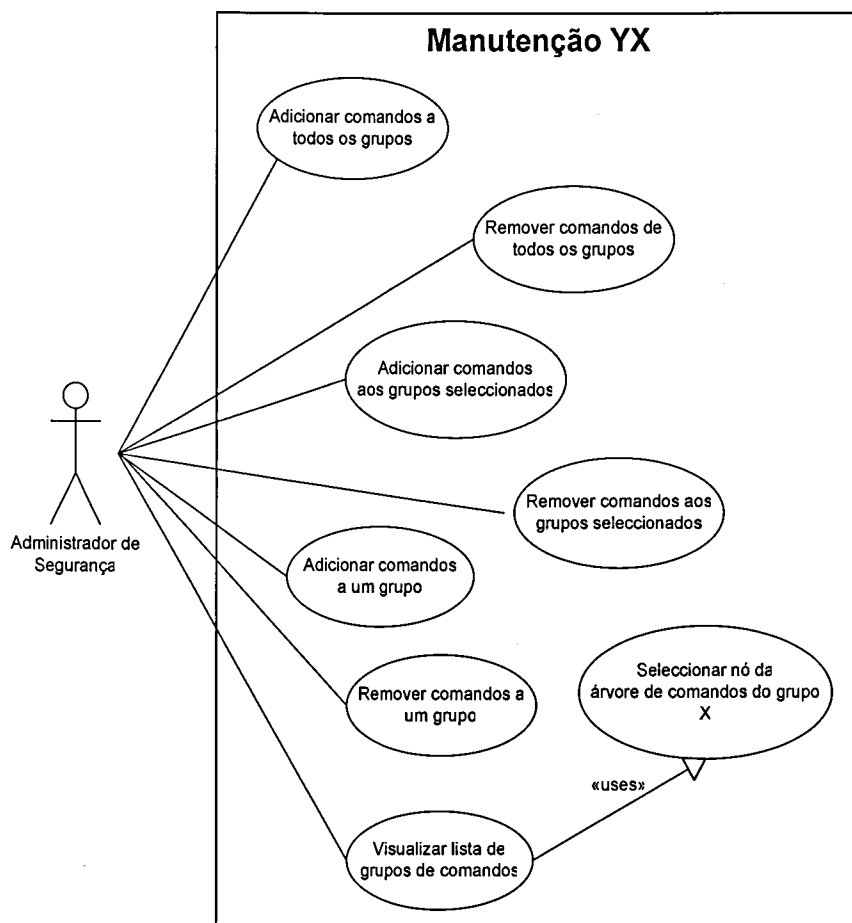


Figura 31 – Pacote de casos de utilização “Manutenção YX”.

Interface Gráfica

As operações disponibilizadas são idênticas às da manutenção XY. A diferença é que agora o grupo X é a *origem* dos comandos e os grupos de comandos que visualizamos na lista são o *destino*. A “restrição do nó seleccionado” mantém-se, ou seja, quando executamos uma das operações, a aplicação considera apenas os comandos do *nó seleccionado*, os quais *pertencem* ao grupo. Podemos ver então, na Figura 32, as 6 funções permitidas:

- opção 1 – adicionar comandos a todos os grupos;
- opção 2 – remover comandos de todos os grupos;
- opção 3 – adicionar comandos aos grupos seleccionados;
- opção 4 – remover comandos aos grupos seleccionados;
- opção 5 (*context-menu*) – adicionar comandos a um grupo;
- opção 6 (*context-menu*) – remover comandos a um grupo.

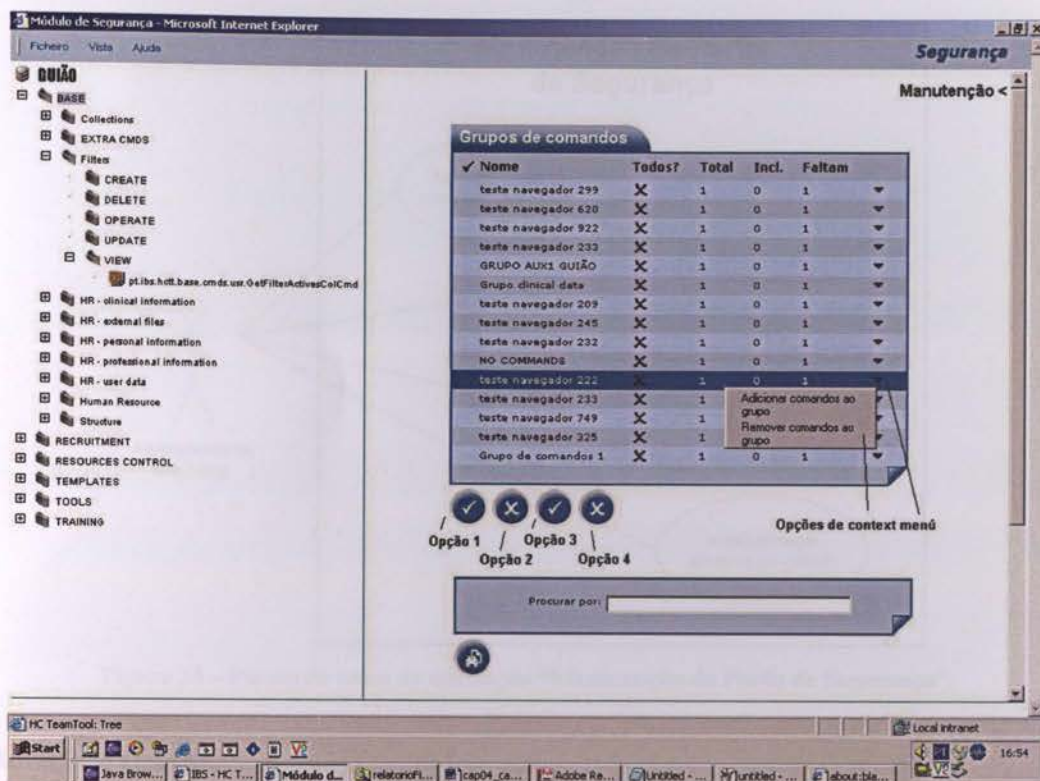


Figura 32 – Página web de manutenção YX de um grupo de comandos.

4.5 Manutenção de Perfis de Segurança

Avançou-se já com uma definição de perfil de segurança na secção 3.1.3 Gestão/Controlo de Autorizações. Pretende-se, como requisito, que um perfil de segurança tenha associadas três outras entidades: um grupo de comandos *autorizado*; um grupo de comandos *não autorizado*; um conjunto de autorizações aos módulos da aplicação.

Os comandos que não pertencem ao grupo de comandos *autorizado* estão, por defeito, não autorizados. Não é necessariamente verdade, no entanto, visto que um utilizador pode pertencer a mais do que um grupo de utilizadores. Os comandos que não estão a ser autorizados explicitamente pelo perfil de segurança A²⁰ podem estar a ser autorizados pelo perfil B. É conveniente portanto a existência de um grupo de comandos *não autorizado*, para que se possa expressamente negar o acesso a um ou mais comandos. A negação tem prioridade sobre a autorização. Além de evitar a proliferação de grupos de comandos muito parecidos, o grupo de comandos *não autorizado* confere robustez à gestão de autorizações pois permite de uma forma simples e explícita negar o acesso a operações. A Figura 33 apresenta este pacote de casos de utilização.

²⁰ Ou seja, que não pertencem ao grupo de comandos autorizado.

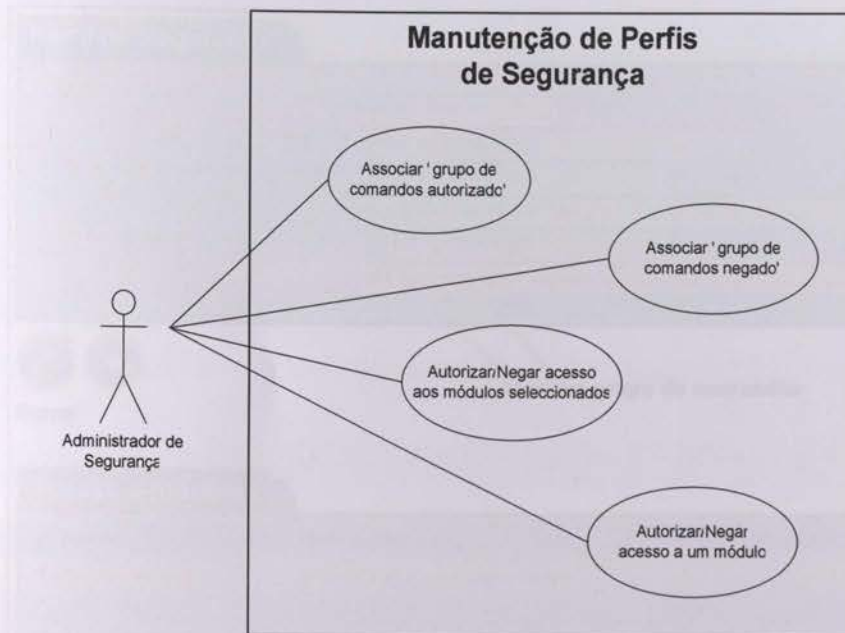


Figura 33 – Pacote de casos de utilização “Manutenção de Perfis de Segurança”.

Interface Gráfica

Depois de seleccionarmos a opção 4, na página inicial do módulo de segurança (Figura 11), surge-nos uma lista com todos os grupos de utilizadores. Para cada grupo, é possível manter o seu perfil de segurança.

✓ Nome	Descrição	
grupoaguiar	TESTES	▼
grupoaguiar2	testereferenced	▼
Grupo p gmata	perfil vai entrar em conflito com admin	▼
Grupo utilizadores 1	para o lcastro	▼
GUIÃO	guião de testes segurança	▼
TESTE NAVEGADOR	grupo com muitos utilizadores testar o navegador	▼
USERS ADMIN	todos os privilégios	▼

Manter perfil de segurança

Figura 34 – Página web de selecção do perfil de segurança.

Após seleccionarmos o perfil de segurança a manter, podemos associar o grupo de comandos *autorizado*, o grupo de comandos *não autorizado* e autorizar/negar o acesso a módulos, como podemos ver na página de manutenção de perfis de segurança (Figura 35). A função de autorizar/negar acesso “funciona como um interruptor”, ou seja, concede autorização se o módulo não estiver autorizado e vice-versa.


Perfil de Segurança

Nome:

Descrição:

Grupo de comandos autorizado:

Grupo de comandos não autorizado:

  **Gravar**

Associar grupo de comandos

Acesso a módulos

✓ Nome	Descrição	Autorização
✓ Colaboradores	Módulo Base - Colaboradores	✓ ▼
✓ Estrutura	Módulo Base - Estrutura	✓ ▼
✓ Recrutamento	Módulo de Recrutamento	✓ ▼
✓ Formação	Módulo de Formação	✓ ▼
✓ Gestão de Recursos	Módulo de Gestão de Recursos	✓ ▼
✓ Base de conhecimento	Manutenção da Base de Conhecimento	✓ ▼

 **Autorizar/Negar Acesso**

Figura 35 – Página web de manutenção de perfis de segurança.

5 Implementação

Este capítulo pretende apresentar de uma forma estruturada e objectiva, os detalhes técnicos da solução implementada.

A primeira parte é dirigida ao leitor que pretende uma visão geral, sem grande detalhe. Tipicamente, será consultada por analistas e gestores de projecto que desejam compreender o enquadramento técnico do módulo de segurança no HCTT. Com este objectivo, apresenta-se inicialmente uma visão geral da arquitectura lógica e física de toda a aplicação.

Esta visão de alto nível introdutória é complementada de seguida com secções onde se aprofunda o nível de detalhe técnico. Apresentam-se pormenores da implementação de uma página *web* muito específica – a manutenção de perfis de segurança – exemplificando todo o processo de desenvolvimento praticado e particularidades técnicas comuns à concretização de todos os casos de utilização. A implementação de todo o módulo tem uma dimensão considerável e a sua documentação exaustiva e detalhada foge um pouco ao âmbito do relatório de estágio, aproximando-se mais do conceito de documentação técnica. Ao apresentar um exemplo, o leitor poderá extrapolar facilmente a dimensão do trabalho envolvido, não perdendo a noção das especificidades técnicas implementadas.

5.1 Arquitectura Lógica do HCTT

Na Figura 36 podemos visualizar a arquitectura lógica do sistema. Esta arquitectura é compatível com a norma *Java 2 Platform, Enterprise Edition (J2EE)*²¹ [11] e segue um estilo de arquitectura em 3 camadas:

- **Camada de interface** – Esta camada é constituída por todas as páginas de *internet* com que o utilizador interage e por todas as classes JAVA (*adapters*) que suportam as páginas de acordo com a regras impostas pela *framework UniIFace*. Estas classes (os *adapters*) permitem a ligação à camada de lógica. Nesta camada existe uma subdivisão em dois pacotes lógicos: o UNIIFace e o GUI (Graphical User Interface). O UNIIFace é uma aplicação reutilizável, proprietária da IBS AB (secção 2.1.1, p.14). Esta *framework* desempenha um papel essencial na apresentação da informação gráfica e no controlo do fluxo de dados. Os outros componentes desta camada estão agrupados no pacote lógico *Graphical User Interface*. A razão para esta separação prende-se com o facto de estes últimos terem sido produzidos especificamente para a aplicação HCTT (ficheiros JSPs, XMLs, ficheiros de tradução) e os outros fazerem parte integrante da *framework* UNIIFace.

²¹ A norma J2EE define conceitos e *standards* para o desenho de aplicações. Para mais informações consultar a referência

- **Camada de lógica de negócio** – Desta camada fazem parte os *comandos*, classes com as quais comunicam os *adapters* e cujo objectivo é fazer a ligação ao núcleo da lógica da aplicação – os *controllers*. Aqui existe também uma subdivisão em dois pacotes lógicos. Os *comandos* encapsulam uma operação, mais ou menos complexa, que corresponde directamente a um “desejo” do utilizador. O conceito de comando, que já apresentámos (secção 3.1.2, p.21), forma uma camada vertical na arquitectura e constitui um mecanismo que permitirá implementar uma solução de segurança elegante e central. Os *controllers* são componentes que possuem operações directamente relacionadas com a camada de persistência e fazem a interface entre esta camada e os *comandos*. Toda a lógica é centralizada nos *controllers* o que implica directamente que não exista código de lógica de negócio nos *comandos*.
- **Camada de persistência** – Implementada na grande maioria com classes JAVA geradas automaticamente pelo *Persistence Builder*. Esta *framework* permite gerar semi-automaticamente um conjunto de classes organizadas em *modelos* e *serviços*. A função destas classes é, por um lado, estabelecer a ligação com a base de dados (*services*) e, por outro, providenciar um modelo utilizável pela camada de lógica das entidades relacionais (*models*).

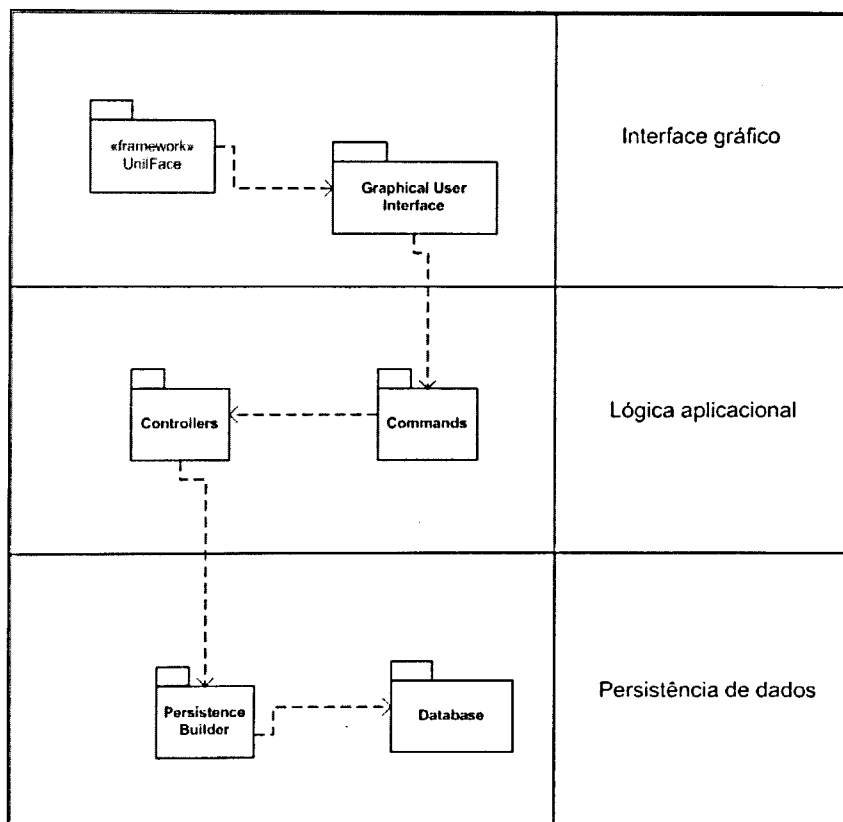


Figura 36 – Arquitectura lógica do sistema – camadas.

A Figura 37 apresenta-nos uma visão ligeiramente mais detalhada, restrita ao âmbito da aplicação HCTT propriamente dita, e centra-se também na divisão entre tipos de

ficheiros. Vejamos que a *framework* UniIFace não aparece agora parte integrante do diagrama mas simplesmente uma indicação de quais são os componentes com os quais interage. Na parte direita da figura podemos ver a divisão dos componentes por tipos de ficheiros. No primeiro nível temos os ficheiros de recurso (*resource files*).

- **XML** – definem a estrutura e conteúdos das páginas web e o mapeamento entre os botões e o código JAVA (adapters).
- **JSP** – páginas web dinâmicas geradas a partir dos ficheiros XML.
- **Bundles** – ficheiros que contêm traduções para várias línguas do texto que é visualizado nas páginas.
- **Other files** – outros ficheiros, por exemplo, ficheiros de configuração.

No segundo nível da arquitectura, temos os ficheiros com o código JAVA da aplicação (Java files). De todos estes ficheiros, existe uma parte gerada automaticamente, que corresponde às classes geradas pelo *Persistence Builder*.

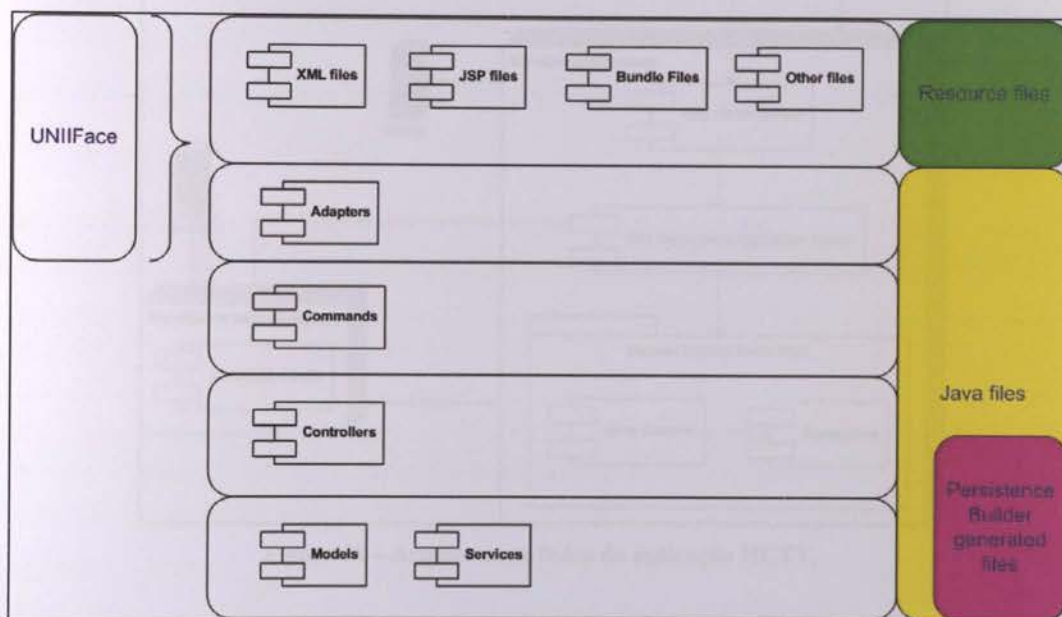


Figura 37 – Arquitectura lógica da aplicação HCTT.

5.3 Base de Dados

5.2 Arquitectura Física do HCTT

A arquitectura da aplicação, a nível físico, é apresentada na Figura 38. É uma arquitectura física denominada *thin client*. Sinteticamente, esta designação indica que a máquina cliente é responsável por pouco mais que permitir o acesso à aplicação ficando todo o processamento centralizado no(s) servidor(es). No caso do HCTT existem dois servidores, o aplicacional e o de base de dados e embora estejam fisicamente separados poderiam estar na mesma máquina.

A grande vantagem desta solução é centralizar toda a configuração e *software* relevantes nos servidores. Alterações ao sistema reduzem-se a intervenções nestas máquinas e o esforço associado à instalação/manutenção de clientes é muito reduzido.

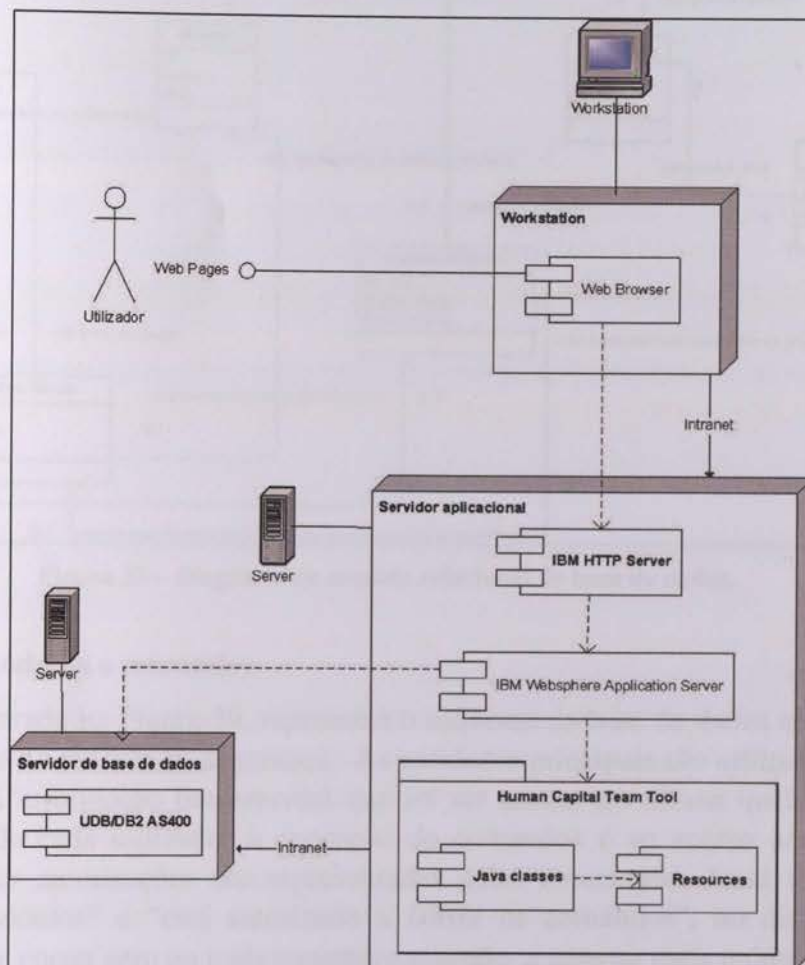


Figura 38 – Arquitectura física da aplicação HCTT.

5.3 Base de Dados

O ambiente de desenvolvimento do HCTT é suportado pelo DB2/400 da IBM e acessado através da tecnologia JDBC. A Figura 39 apresenta o diagrama do modelo relacional.

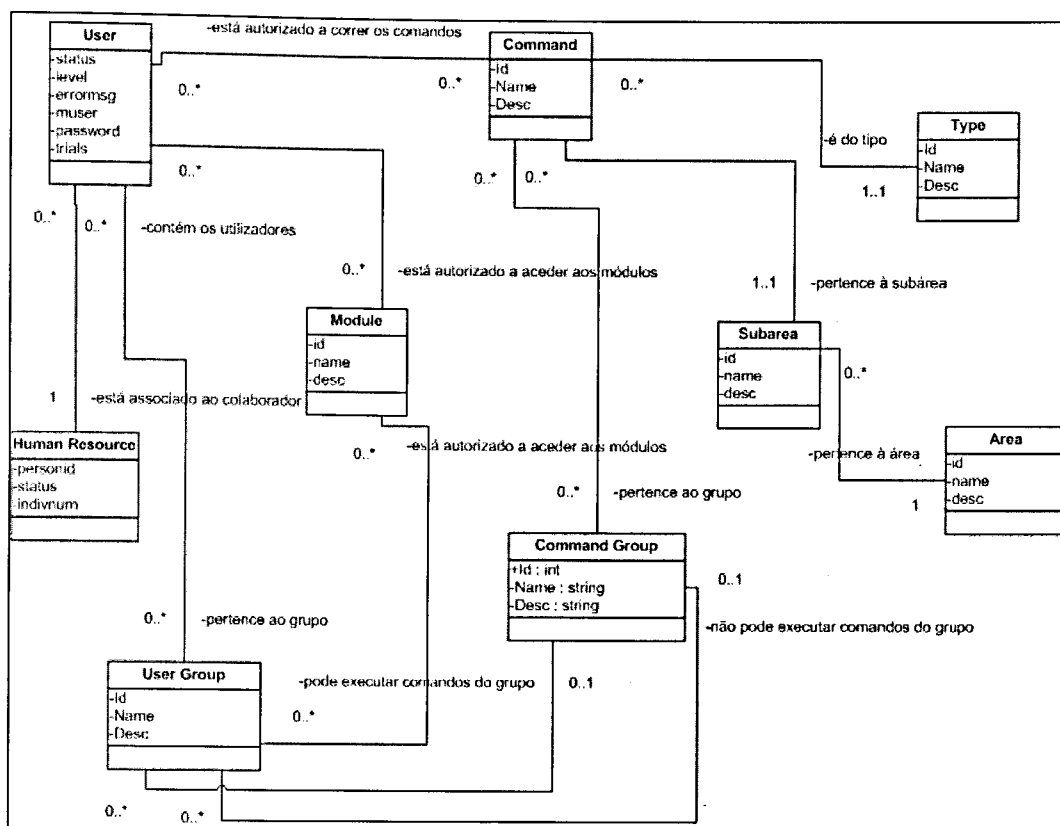


Figura 39 – Diagrama do modelo relacional de base de dados.

Utilizador, módulos e comandos

O modelo ilustrado na Figura 39, representa o esquema da base de dados que suporta a funcionalidade do módulo de segurança. As entidades principais são *utilizador*, *módulo* e *comando*. A informação fundamental que irá ser usada, em última instância, são as autorizações de cada utilizador à execução de comandos e ao acesso aos diferentes módulos. Estas autorizações são representadas pelas associações “está autorizado a aceder aos módulos” e “está autorizado a correr os comandos”, no diagrama. Um utilizador pode correr zero ou mais comandos e aceder a zero ou mais módulos.

No entanto, como se verificou já no capítulo 4, a gestão destas autorizações só se tornará possível, de um modo razoável, se utilizarmos, por um lado, os conceitos de grupos de utilizadores e grupos de comandos e, por outro, uma divisão e classificação equilibrada do grande número de comandos existentes. De seguida apresentam-se as entidades restantes e as suas relações.

Classificação de comandos

A classificação de comandos adoptada associa um *tipo* e uma *sub-área funcional* a cada comando registado na base de dados. Um comando pode ter um e um só tipo associado, e uma e uma só sub-área. Um tipo pode ter muitos comandos associados e o mesmo se passa para a sub-área. Devido ao elevado número de comandos existentes, concluiu-se que uma só divisão funcional não seria suficiente. Decidiu-se portanto fazer uma

divisão ao nível das sub-áreas, associando-lhes uma e uma só *área*. Uma área pode ter mais do que uma sub-área, ou estar vazia.

Grupos de comandos

Além da classificação equilibrada de comandos, foi necessário também criar uma entidade que agrupasse os comandos em conjuntos manipuláveis. Um grupo de comandos pode ter zero ou mais comandos e um comando pode pertencer a zero ou mais grupos de comandos.

Grupos de utilizadores

Para tornar o conjunto de utilizadores manipulável, criou-se a entidade grupo de utilizadores, que pode conter zero ou mais utilizadores. Criaram-se mais três relações de autorizações: duas com os grupos de comandos e uma com a entidade módulo. Estas relações são a materialização do conceito de perfil de segurança (secção 3.1.3, p.22). A um grupo de utilizadores estão associados zero ou mais módulos. As relações com os grupos de comandos têm, no máximo, um grupo associado mas podem não ter nenhum.

Autorizações

Podemos ver então que existem várias relações que guardam informação das autorizações dos utilizadores. No entanto, todas as autorizações são calculadas no momento em que são atribuídas na manutenção de perfis de segurança e registadas nas relações entre a entidade utilizador e as entidades *módulo* e *comando*, de modo a que fiquem *centralizadas*. Isto pretende aumentar a eficiência de consulta das autorizações e simplificar todo o processo. Sendo assim, as relações de autorizações que vimos para os grupos de utilizadores servem para *determinar* estas autorizações. Na lógica de negócio, quando queremos saber se um utilizador pode ou não executar um comando, ou pode ou não aceder a um módulo, são apenas estas relações que vão ser consultadas (ver relações entre a tabela *user* e as tabelas *comando* e *módulo*).

5.4 Mecanismos Importantes

Existem alguns mecanismos de implementação importantes que saem fora dos exemplos apresentados na secção anterior. O seu carácter essencial, como suporte do módulo de segurança, fez com que não pudéssemos deixar de apresentá-los.

5.4.1 Processo de Autenticação

O processo de autenticação já anteriormente implementado, foi modificado para carregar as autorizações dos utilizadores no momento do *login*. A Figura 40 apresenta um diagrama com as principais classes participantes no processo.

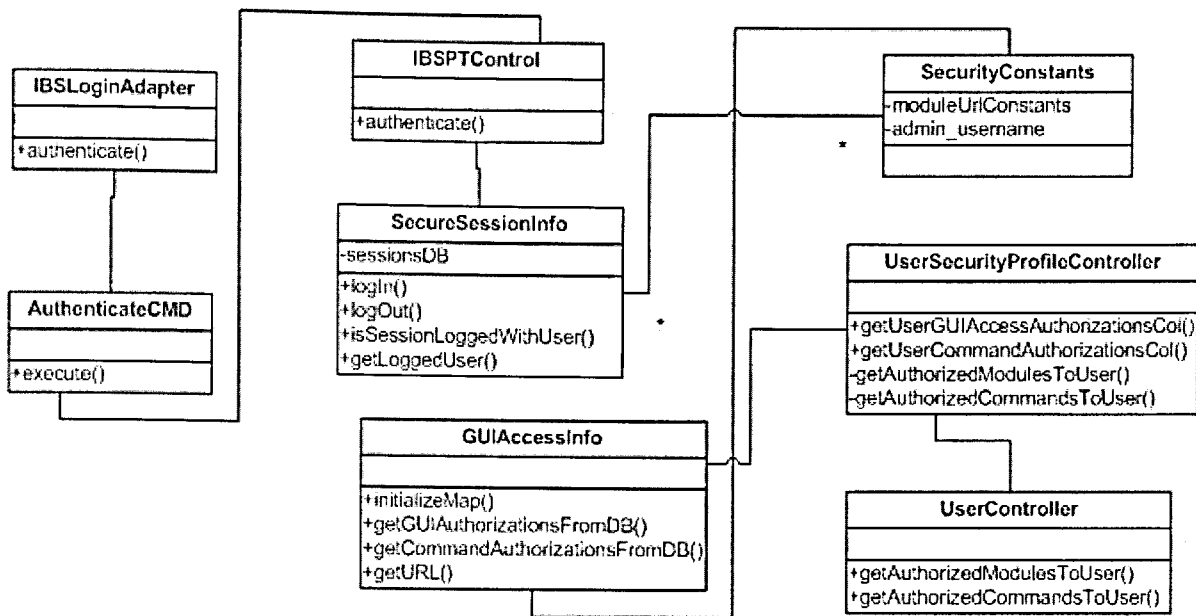


Figura 40 – Processo de autenticação: classes participantes.

O *adapter* associado à página de *login* da aplicação é o *IBSLoginAdapter* que invoca o comando *AuthSCUserCmd* que valida o utilizador e a palavra-passe na base de dados. Este mecanismo básico estava já implementado.

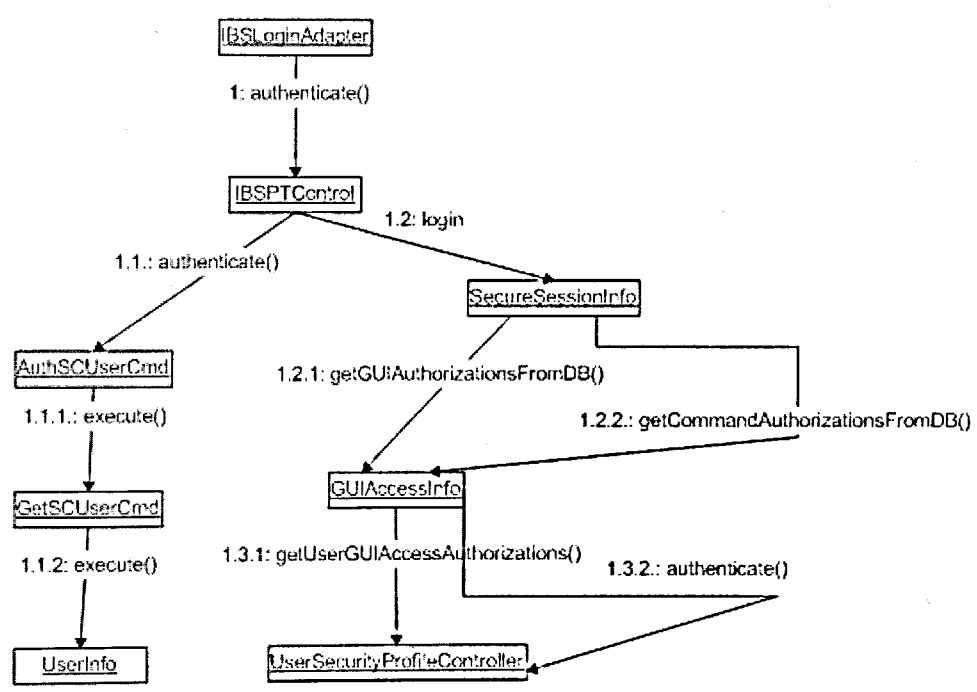


Figura 41 – Processo de autenticação: diagrama de colaboração.

A alteração introduzida foi a adição de uma estrutura de dados (*SecureSessionInfo*) que guarda as autorizações do utilizador aos comandos e as autorizações aos módulos. Esta estrutura de dados está alojada no servidor e só o servidor tem acesso a ela.

Imediatamente após ter efectuado o *login* na aplicação, esta classe consulta a base de dados e carrega as autorizações para memória. Podemos ver o diagrama de colaboração deste processo na Figura 41.

5.4.2 Segurança de Execução

Todos os comandos da aplicação estão organizados numa estrutura hierárquica que permitiu centralizar a segurança de execução. Como podemos ver na Figura 42, todos os comandos *estendem ABUsrCmd*. Existe mais um nível, que pretende “agrupar” os comandos por módulos funcionais.

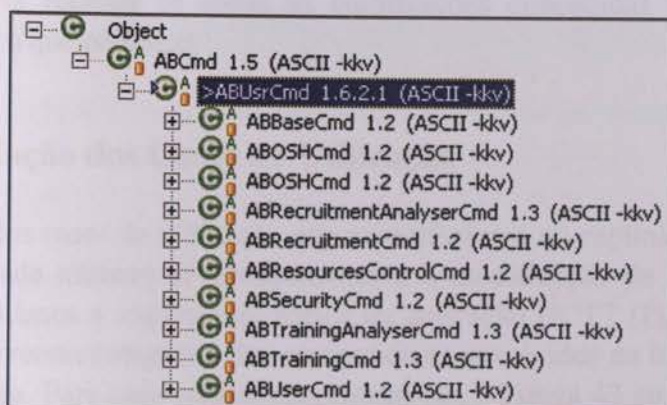


Figura 42 – Hierarquia de comandos.

No *construtor* da classe *ABUsrCmd* é feita a verificação se determinado comando está autorizado a ser executado pelo utilizador que o invocou. Todos os comandos têm como parâmetro o utilizador que o está a utilizar. Se for o utilizador ADMIN, a autorização não é consultada na base de dados, pois este utilizador pode executar qualquer comando.

5.4.3 Actualização/Integridade das Autorizações

Sempre que são efectuadas alterações a grupos de utilizadores, grupos de comandos ou a associações dos grupos de utilizadores, a aplicação **recalcula as autorizações de todos os utilizadores**. A autorização à execução de comandos é instantânea, e efectiva a partir desse momento. Os utilizadores necessitam de fechar a sessão, e iniciar uma nova, para serem afectados pelas alterações.

Regras para a definição de autorizações

A negação tem prioridade sobre a autorização. Esta regra será importante na resolução de conflitos de autorizações porque um utilizador pode pertencer a mais do que um grupo de utilizadores. Este requisito tem uma implicação prática importante. Conceptualmente, todos os utilizadores de um grupo *são afectados* pelo mesmo conjunto de autorizações a comandos e a módulos. No entanto, podem não *usufruir* das

mesmas autorizações pois as negações podem entrar em conflito com autorizações de outros perfis. As regras para resolução de conflitos são as seguintes:

- Para definir o conjunto C – este conjunto contém todos os comandos que um utilizador específico pode executar – é necessário definir primeiro dois conjuntos A e B. Chamemos ao conjunto final, conjunto C. O conjunto A é definido pela reunião de todos os *grupos de comandos autorizados* dos grupos de utilizadores a que pertence. O conjunto B é definido pela reunião de todos os *grupos de comandos não autorizados* dos grupos de utilizadores a que pertence. O conjunto C é igual a A - B (A menos B).
- Não existem conflitos no que diz respeito à autorização de acesso a módulos porque não é possível negar o acesso a um módulo explicitamente. Isto quer dizer que o conjunto de módulos a que um utilizador específico pode aceder é definido pela reunião de todas as autorizações concedidas pelos grupos de utilizadores a que pertence.

5.5 Implementação dos Casos de Utilização

A implementação dos casos de utilização que especificámos no capítulo 4 deu origem à criação de um grande número de componentes e à modificação de muitos outros já existentes. Apresentámos a arquitectura lógica da aplicação HCTT (Figura 37, p.50) e aí verificámos as diversas categorias dos componentes envolvidos na implementação do módulo de segurança. Para cada uma dessas categorias, a Figura 43 apresenta o número de componentes envolvidos na implementação desenvolvida ao longo do estágio.

Tipo de componente	Quantidade
JSPs	29
Bundles	2
Adapters	5
Comandos	60
Controllers	14
Models & Services	195 + 106
Outros ficheiros/classes	21

Figura 43 – Componentes do módulo de segurança.

É um número muito grande de componentes e não seria de todo razoável apresentá-los a todos detalhadamente. Além disso, a implementação dos casos de utilização partilham de um padrão de desenvolvimento comum que garantidamente se repetiria, se optássemos por uma descrição exaustiva de cada implementação específica, sem acrescentar grande valor a este relatório.

Optou-se portanto por uma solução que se afasta do conceito de *documentação técnica detalhada* e se aproxima mais de uma *amostra exemplificativa*, tanto do processo de desenvolvimento, bem como de alguns detalhes técnicos importantes, que não poderiam deixar de estar presentes. De entre todos os casos de utilização especificados, considerámos que o que cumpriria melhor o papel de uma *boa amostra* seria o de “Manutenção de Perfis de Segurança” (secção 4.5, p.45).

Dividiu-se a secção em duas partes: *Interface Gráfica e Lógica*. A primeira mostra como se constroem as páginas *web* do HCTT e explica de que forma o *UniFace* faz a ligação entre a interface gráfica e a lógica de negócio implementada, a qual é apresentada na segunda parte.

5.5.1 Interface Gráfica

Em primeiro lugar é necessário construir a página *web* que foi especificada. Foi já referido que a criação de páginas é feita em *build time* através de uma ferramenta de geração automática - o *JSP Generator* - a partir da definição das páginas contidas em ficheiros XML (a Figura 44 evidencia também este facto). A *framework UniFace* possibilita a criação de vários tipos de páginas e a possibilidade de agregar várias páginas básicas numa só (página *multi-part*).

Relembremos o aspecto da página, consultando a Figura 35 (p.47). Nesta situação temos uma página *multi-part* composta por duas páginas que foram definidas no ficheiro *Profile.XML*. A primeira é do tipo *view_detail_edit*²² e a segunda do tipo *view_list_maintenance*²³.

Carregamento da página

Vimos já na Figura 4 (p.15), que o *UniFace* se ocupa da interacção com o *web browser*, gerindo as ligações entre as páginas e as operações que são efectuadas pelo utilizador²⁴. Os ficheiros XML guardam, para além de outras informações, as ligações entre as páginas e a ligação à lógica de negócio.

Quando o utilizador seleccionou “Manter Perfil” (Figura 34), o *UniFace* iniciou o carregamento do JSP *SECPROFILEMAINTENANCE_MV*, que está definido no ficheiro *Profile.XML*. Como contém duas páginas distintas, o carregamento é feito primeiro para uma e depois para outra. Para páginas do tipo *view_detail_edit*, o *UniFace* invoca o método *loadObject()* de um *adapter*, e para as páginas do tipo *view_list_maintenance* invoca o método *loadList()*. Na Figura 44 podemos ver que *adapter* está associado a esta página (*ProfileAdapter.java*) e de um modo esquemático,

²² Neste caso, queremos visualizar a informação de uma entidade particular: o grupo de utilizadores cujo perfil de segurança estamos a manter. Esta página possibilita ainda a edição dessa informação, daí as caixas de texto à frente de cada campo.

²³ Aqui queremos visualizar apenas a informação de várias entidades e disponibilizar operações genéricas que podem ser aplicadas a cada uma das linhas da lista.

²⁴ Existem situações em que esta gestão é auxiliada por *Java Script* e por *servlets* construídos “à medida” de forma a estender a funcionalidade oferecida pelo *UniFace*. De qualquer dos modos, a própria *framework* possibilita a integração destas tecnologias.

perceber este mecanismo de carregamento da página. Estes métodos (*loadObject()* e *loadList()*) invocam *comandos* cuja função é disponibilizar a informação que está na base de dados ao *UniFace* de modo a que esta apareça na página *web*.

Funcionalidade

Além dos métodos *standard* dos *adapters* (*loadObject()* e *loadList()*) que são utilizados para carregar as páginas *web*, existem também outros métodos que são definidos à medida da funcionalidade da página. Neste caso temos duas operações distintas: *gravar* as associações a grupos de comandos e *autorizar/negar acesso* a um determinado módulo da aplicação. Podemos verificar na Figura 44 que estas operações estão directamente associadas aos métodos *updateProfile()* e *toggleAccess()*, respectivamente. Novamente, estas associações estão definidas, de um modo estático, no ficheiro *Profile.XML*, e os métodos invocam *comandos* que encapsulam a operação.

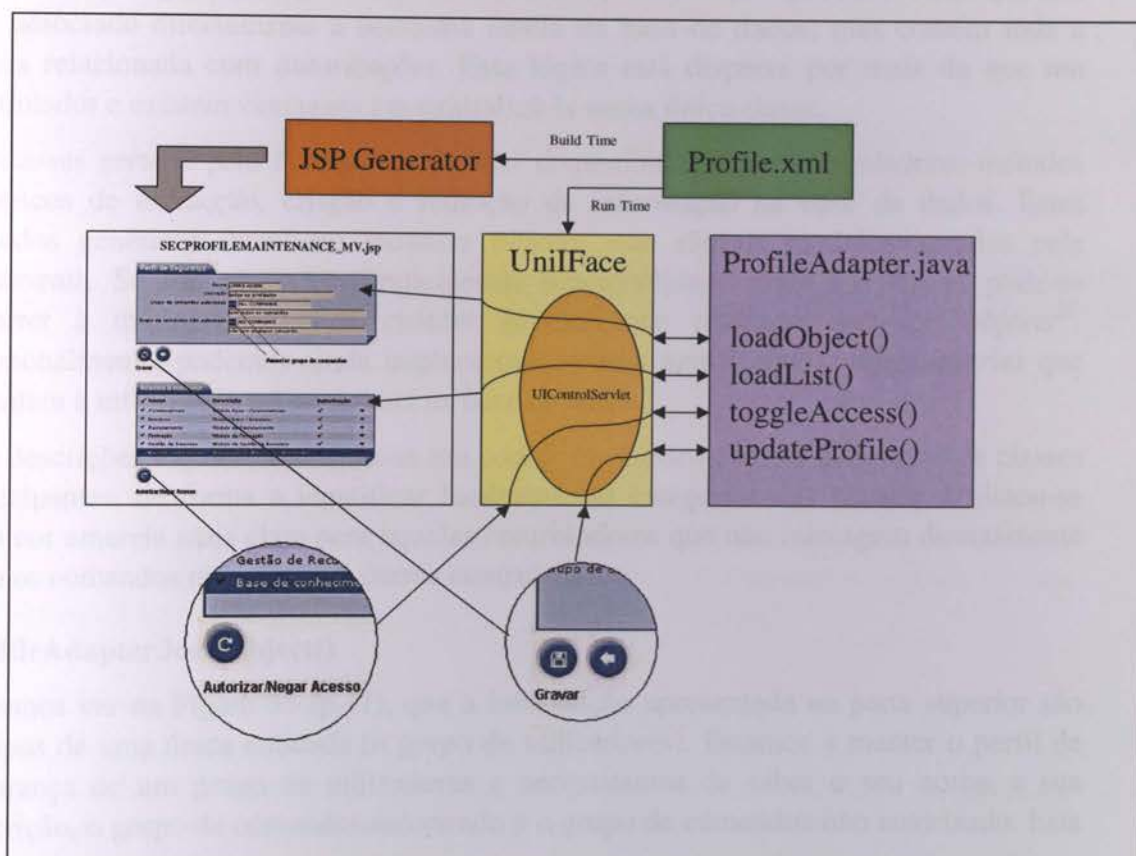


Figura 44 – Ligação da interface gráfica à lógica de negócio.

5.5.2 Lógica

Antes de avançarmos para a explicação do funcionamento de cada método da classe *ProfileAdapter* (Figura 44), importa considerar algumas generalidades da arquitectura que foram respeitadas em toda a implementação do módulo de segurança.

Quando analisámos a arquitectura lógica da aplicação HCTT, verificámos que a camada dos *adapters* comunica com uma outra camada, que contém os *comandos*. Assim sendo, os métodos dos *adapters* são implementados de forma a receber os parâmetros que vêm da página *web* e invocar a execução de um ou mais *comandos*. Os *adapters* utilizam classes denominadas por *attribute ids*, que contém constantes de strings de *tags* utilizadas nos ficheiros XML. Deste modo, é centralizada toda a informação necessária para a comunicação entre estes ficheiros e as classes Java.

Todos os comandos da aplicação estão organizados numa estrutura hierárquica que será mencionada na secção 5.4.2 (p.55). Por sua vez, os comandos invocam métodos dos *controllers* que contém toda a lógica de negócio.

Cada controlador está associado a uma única entidade ou relação da base de dados. Tipicamente, teremos o mesmo número de controladores do que *homes*²⁵ existentes. Existem alguns casos excepcionais, onde se utilizam controladores que não estão associados a nenhuma *home*, que encapsulam lógica de negócio relacionada. Esta característica de arquitectura torna o código mais legível e de mais fácil manutenção. Nas próximas figuras iremos ver um exemplo, o *UserSecurityProfileController*, que não está associado directamente a nenhuma tabela da base de dados, mas contém toda a lógica relacionada com *autorizações*. Esta lógica está dispersa por mais do que um controlador e existem vantagens em centralizá-la numa única classe.

As classes geradas pelo *Persistence Builder* disponibilizam aos controladores métodos genéricos de extracção, criação e remoção de informação na base de dados. Estes métodos genéricos devolvem *business objects*, que são os modelos gerados pela ferramenta. Se fôr necessário implementar funcionalidade extra nas *homes* pode-se recorrer à implementação de *custom queries*, que retornam *business objects*²⁶. Adicionalmente, podemos ainda implementar *custom non-business object queries* que retornam a informação tal como está na base de dados.

Nas descrições seguintes, utilizou-se um *código cromático* para os diagramas de classes participantes, de forma a identificar facilmente as categorias das classes. Utilizou-se uma cor amarela mais clara para aqueles controladores que não interagem directamente com os comandos mas sim com outros controladores.

ProfileAdapter.loadObject()

Podemos ver na Figura 35 (p.47), que a informação apresentada na parte superior são campos de uma única entidade (o grupo de utilizadores). Estamos a manter o perfil de segurança de um grupo de utilizadores e necessitamos de saber o seu nome, a sua descrição, o grupo de comandos autorizado e o grupo de comandos não autorizado. Esta

²⁵ As classes *home* são os modelos criados pela ferramenta Persistence Builder que estão associados a uma tabela específica da base de dados. Cada controlador interage com uma só tabela. Se necessitar de interagir com outras, deverá utilizar métodos do respectivo controlador.

²⁶ Um método genérico das *homes* é o método *find()* que retorna um *business object* a partir da sua chave primária. Há situações em que queremos fazer uma pesquisa a partir de outro campo. Nestes casos é necessário implementar uma *custom (Business Object) query*.

informação vai ser colocada numa classe do tipo *Map*²⁷ e depois redireccionada para o *UniFace*.

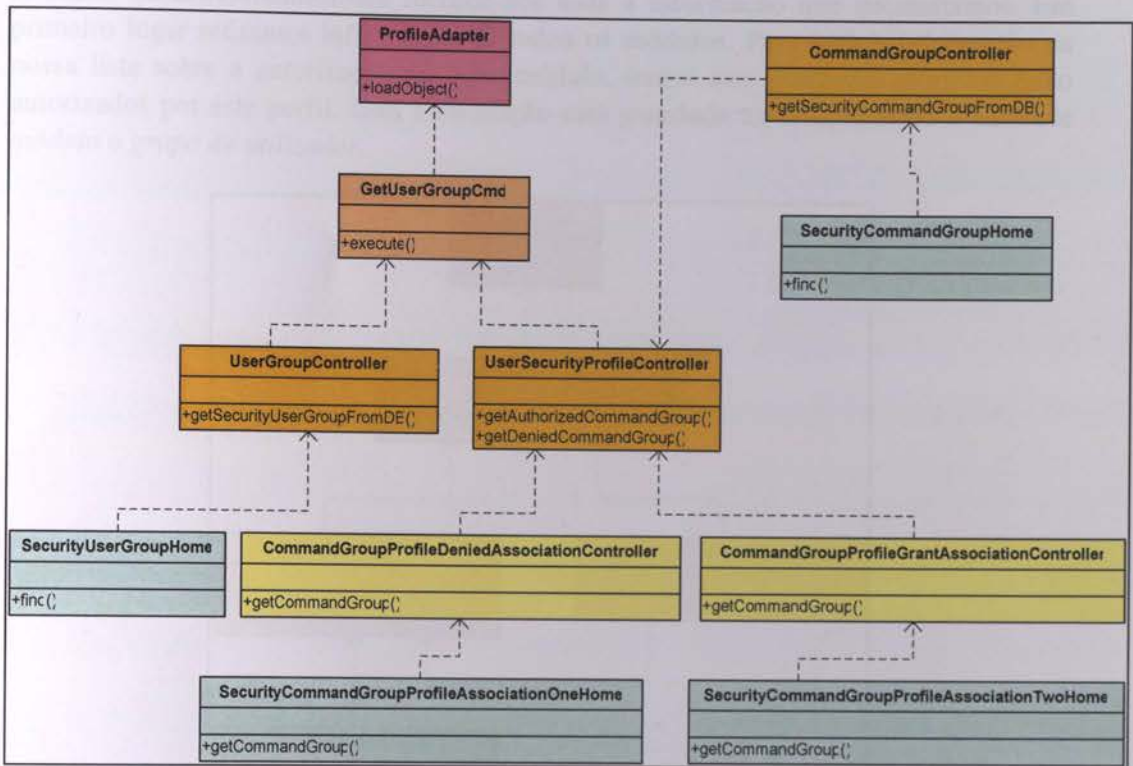


Figura 45 – Lógica do método *ProfileAdapter.loadObject()*: classes participantes.

Para carregar esta informação é necessário consultar 4 tabelas, como podemos ver na Figura 45: existem 4 classes participantes do tipo *home* (côr azul). É necessário consultar a tabela *SecurityUserGroup* (informação do nome e descrição), a tabela *SecurityCommandGroup* (informação dos grupos de comandos autorizado e não autorizado) e as tabelas de associação entre essas duas entidades.

ProfileAdapter.loadList()

Voltemos a considerar a Figura 35 (p.47), mas agora a parte inferior que contém uma lista de todos os módulos da aplicação e informação sobre se este perfil de segurança que estamos a manter *autoriza* ou *não autoriza* o acesso.

Os métodos *loadList()* retornam objectos do tipo *List*²⁸ que contém *Maps*, com a informação de cada linha que vemos na página. Neste caso, verificámos que é

²⁷ Um *Map* é uma estrutura de dados típica que guarda um conjunto de pares chave-valor. É possível depois aceder eficientemente a esse valores indicando as suas chaves.

²⁸ Uma *List* é também uma estrutura de dados típica que contém vários elementos encadeados e aos quais é possível aceder através de índices, que são os números de ordem pelos quais os objectos estão posicionados no conjunto.

apresentado o nome do módulo, a sua descrição e um símbolo gráfico que indica se o perfil de segurança autoriza ou não o acesso ao módulo.

A classe *SecurityModuleHome* fornece-nos toda a informação que necessitamos. Em primeiro lugar retiramos informação de todos os módulos. Para incluir informação na nossa lista sobre a autorização de cada módulo, temos que saber que módulos estão autorizados por este perfil. Esta informação está guardada na relação entre a entidade *módulo* e *grupo de utilizador*.

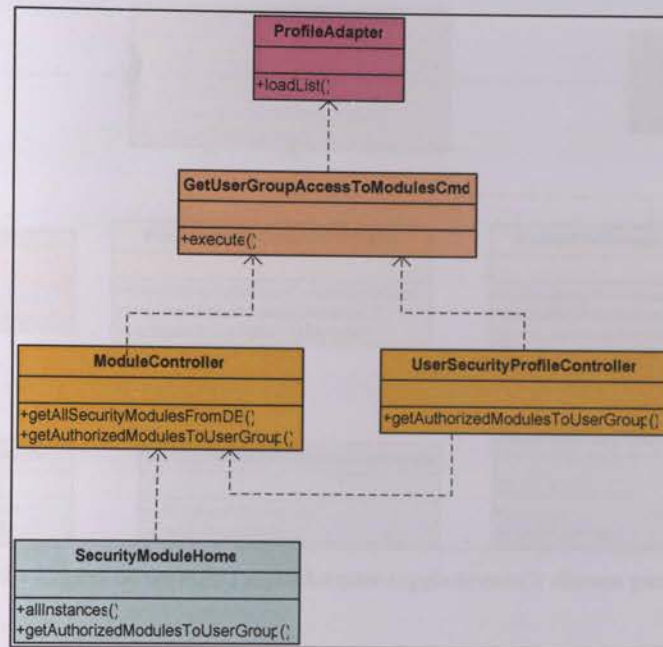


Figura 46 – Lógica do método *ProfileAdapter.loadList()*: classes participantes.

ProfileAdapter.toggleAccess()

Existem 3 situações em que é necessário recalcular todas as autorizações da aplicação, tal como se menciona na secção 5.4.3 (p.55). Ao alterarmos as autorizações a módulos de um perfil (*ToggleUserGroupAccessToModuleCmd*), necessitamos de invocar o comando *ProcessAuthorizationsCmd*.

O primeiro comando verifica se o módulo que seleccionámos está ou não autorizado (*UserSecurityProfileController.isUserGroupAuthorizedToAccessModule()*) e depois *nega* ou *autoriza* o módulo, respectivamente.

O algoritmo de (re-)cálculo de todas as autorizações remove primeiro todas as autorizações existentes (*CmdSecurityController.removeAuthorizations()*) e depois utiliza uma única pergunta *SQL* (um pouco complexa) para determinar as novas autorizações. As autorizações a módulos estão numa tabela separada (*SecurityModuleUserAssociation*) e a sua recalculação segue o mesmo raciocínio. Finalmente, a estrutura de dados *SecureSessionInfo* é actualizada (secção 5.4.1).

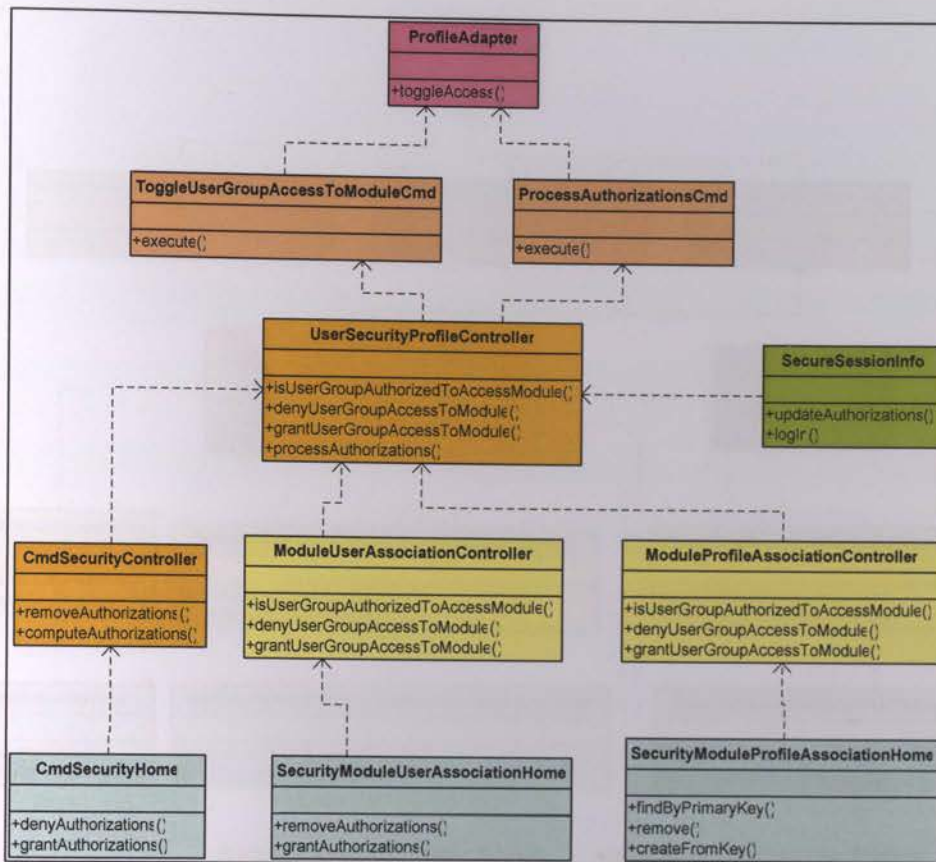


Figura 47 – Lógica do método *ProfileAdapter.toggleAccess()*: classes participantes.

ProfileAdapter.updateProfile()

Tal como o método *toggleAccess()* anteriormente descrito, esta operação invoca também o comando *ProcessAuthorizationsCmd* que re-calcula todas as autorizações da aplicação.

Na página que estamos a considerar é possível alterar os grupos de comandos autorizado e não autorizado com recurso a uma outra página que lista todos os grupos de comandos existentes. Para isso será necessário utilizar os botões respectivos para associar outros grupos (Figura 35, p.47). Quando voltamos à página de manutenção do perfil, se quisermos guardar estas alterações, utilizamos o botão gravar. Esta operação elimina as associações entre o grupo de utilizadores e os grupos de comandos e grava a nova informação.

Além das classes utilizadas para a re-computação de autorizações, utilizam-se novamente as *homes* correspondentes às associações entre as entidades *grupo de utilizadores* e *grupo de comandos* (tal como no método *loadObject()*) mas agora outros métodos, responsáveis por apagar e inserir informação.

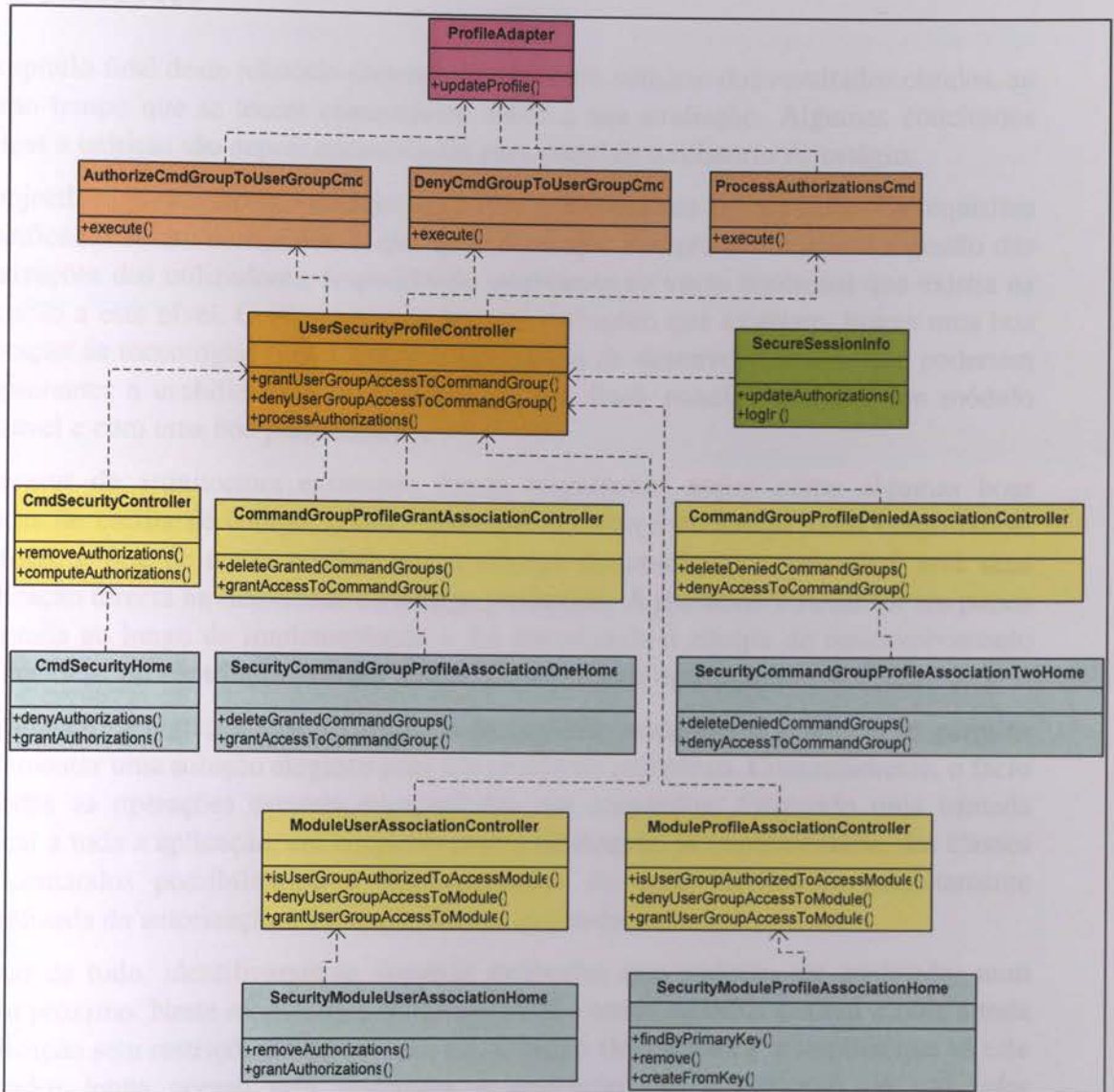


Figura 48 – Lógica do método *ProfileAdapter.updateProfile()*: classes participantes.

6 Conclusões

No capítulo final deste relatório pretende-se fazer um sumário dos resultados obtidos, ao mesmo tempo que se tecem comentários sobre a sua avaliação. Algumas conclusões técnicas e teóricas são depois apresentadas para concluir o relatório de estágio.

Os objectivos para o módulo de segurança foram plenamente conseguidos. Os requisitos especificados foram cumpridos, o que quer dizer que cumpre eficazmente a gestão das autorizações dos utilizadores, respondendo totalmente ao vazio funcional que existia na aplicação a este nível. Contornaram-se bem as restrições que existiam, houve uma boa adaptação às tecnologias novas usadas pela equipa de desenvolvimento, que poderiam comprometer a usabilidade e os resultados, e no final, conclui-se que é um módulo utilizável e com uma boa performance.

As regras de arquitectura existentes foram respeitadas, assim como algumas boas práticas de escrita de código, como é exemplo o esforço de redigir bons comentários. No final, a fase de testes revelou que o esforço desenvolvido a este nível teve uma implicação directa na fiabilidade do código produzido. A *framework* JUnit foi um pouco explorada ao longo da implementação e foi introduzida à equipa de desenvolvimento revelando-se um contributo valioso, antevendo-se futuras aplicações.

A arquitectura lógica do HCTT revelou-se bastante poderosa e flexível, ao permitir implementar uma solução elegante para um problema complexo. Concretamente, o facto de todas as operações estarem encapsuladas em comandos, formando uma camada vertical a toda a aplicação, em conjunto com a hierarquia, já implementada, das classes dos comandos possibilitaram a implementação de uma verificação perfeitamente centralizada da autorização à execução de um comando.

Apesar de tudo, identificaram-se algumas melhorias que poderão ser analisadas num futuro próximo. Neste momento, o utilizador com o nome ADMIN poderá aceder a toda a aplicação sem restrições. Esta solução não é muito flexível porque implica que só este utilizador tenha acesso sem restrições à aplicação. Se o conceito de *utilizador administrador* fosse estendido para os outros utilizadores, poderíamos permitir que um determinado utilizador, qualquer que ele fosse, pudesse aceder à aplicação sem restrições. O conceito de aceder sem restrições significa que no momento de verificação da autorização, a aplicação não consulta a base de dados e assume que a autorização é concedida.

Uma outra melhoria que está no momento a ser estudada é a implementação de grupos de comandos *standard* que farão parte da instalação da aplicação e que não poderão ser modificados pelo utilizador. Estes grupos pré-definidos apresentarão ao utilizador um ponto de partida útil, uma sub-divisão prática do grande número de comandos existente. A classificação de comandos que se implementou permite, de facto, a criação usável de grupos de comandos, mas a inexistência de qualquer grupo de comandos à partida

obriga o utilizador a defini-los. Se uma boa solução inicial estiver presente, prontamente utilizável, pode aumentar ainda mais a usabilidade do módulo, na maior parte das instalações. Estes grupos iniciais poderão mesmo ir sendo aperfeiçoados com base na informação recolhida nas sucessivas instalações da aplicação em variados clientes e situações.

O estágio revelou-se uma experiência muito valiosa a nível pessoal, porque permite consolidar os conhecimentos adquiridos no meio académico. Particularmente, verificou-se que no meio empresarial aparecem inúmeras variáveis que não permitem a aplicação fiel das teorias de Engenharia de Software, devido às grandes limitações de tempo e de recursos. É necessário um exercício constante de cedência relativamente às necessidades de resultados em tempo útil.

7 Referências

Publicações

- [1] Ian Sommerville, *Software Engineering*, Addison-Wesley, 2001
- [2] E. Gamma, R. Helm, R. Johnson, and J. Vlissades, *Design Patterns: Elements of Reusable Software Architecture*, Addison-Wesley, 1995

Sítios na Internet

- [3] <http://www.ibs.pt>
- [4] <http://www.ibs.com>
- [5] <http://www.quest.com/jprobe/index.asp>
- [6] <http://www.irongrid.com/catalog/default.php>
- [7] http://www.cbd-hq.com/wordabuse/wordabuse_frameworks.asp
- [8] <http://java.sun.com/products/jdbc/overview.html>
- [9] <http://jakarta.apache.org/tomcat/~>
- [10] <http://computing-dictionary.thefreedictionary.com/>
- [11] http://publib.boulder.ibm.com/infocenter/wasinfo/index.jsp?topic=/com.ibm.was.ee.doc/info/ee/ae/covr_j2ee.html
- [12] <http://www.eclipse.org>
- [13] <http://java.sun.com/products/jaas/>
- [14] <http://www.junit.org/index.htm>

8 Glossário

[15] Navegador genérico do HCTT

É uma normalização gráfica e funcional da aplicação; sempre que é necessário utilizar listas de elementos, apenas entre 15 a 20 elementos são visíveis de cada vez; conjuntamente com a lista, inclui-se uma funcionalidade que permite pesquisar e *navegar* em toda a lista. Pode ver-se um exemplo na Figura 49. As funcionalidades presentes são: a) pesquisar de elementos; neste caso, aparecem apenas os elementos da lista que preenchem os requisitos da pesquisa; b) Ir para uma página à escolha do utilizador; repare-se que o próprio navegador contém informação do total de páginas da lista que está a ser visualizada; c) ir para a página anterior; d) ir para a página seguinte.

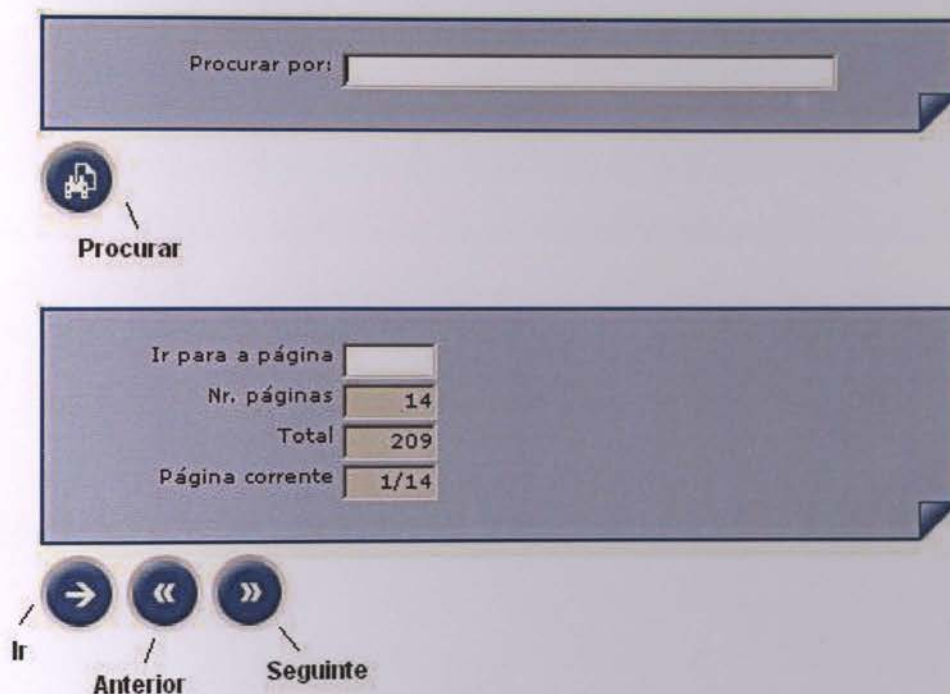


Figura 49 – Navegador genérico do HCTT

[16] JDBC

“JDBC significa Java Database Connectivity e é um standard industrial para a conectividade independente da base de dados entre a linguagem de programação JAVA e um grande número de bases de dados” [10]. Consultar [8] para mais informação.

[17] Open-Source

“É um método e uma filosofia para licenciamento e distribuição de software designado para encorajar o uso e o melhoramento do software escrito por voluntaries assegurando que qualquer pessoa pode copiar o código fonte e modificá-lo livremente (...)”[10].

[18] Depuração (*debugging*)

“Tentativa de determinar a causa de sintomas ou defeitos detectados através de testes ou pela utilização normal de uma aplicação”[10].

[19] Debugger

Aplicação, usualmente integrada num IDE mas não necessariamente, que auxilia o processo de depuração de código [18].





FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

BIBLIOTECA



0000081490

004(047.
EIC5202 20