



Universidade do Porto
Faculdade de Engenharia

FEUP



Tiago José Lázaro Mendes

Sistema de Apoio à Execução, Auditoria e Decisão do Projecto Porto Digital

na ParadigmaXis, S.A.

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



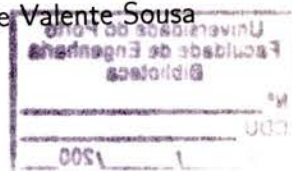
**Sistema de Apoio à Execução, Auditoria e Decisão do Projecto
Porto Digital
ParadigmaXis, SA**

Relatório do Estágio Curricular da LEIC 2004/05

Tiago José Lázaro Mendes

Orientador na FEUP: Professor João Correia Lopes

Orientador na ParadigmaXis: Engenheiro Alexandre Valente Sousa



Setembro de 2005

004(047.3)LEIC/ESC 5202 2005/MENL

Universidade do Porto
Faculdade de Engenharia
Biblioteca
Nº 81461
CDU 004.415(047.3)
Data 17/03/2006

ParadigmaXis — Arquitectura e Engenharia de Software, S.A.

Projecto: Sistema de Apoio à Execução, Auditoria e Decisão do Projecto Porto Digital

Tema: Processamento de Pedidos de Pagamentos

MD5 digital signature: 5A538C73661E87FB5BC04C71FE61E2CA
Last change: 13:28 4/Oct/2005
Last processing: 13:34 04/Oct/2005 (processed by dotNoweb 0.25.d)
Internet: <http://www.fe.up.pt/~ei00081/estagio>
Autor: Tiago Lázaro Mendes (ei00081@fe.up.pt)

Orientador na empresa:

Alexandre Valente Sousa
ParadigmaXis
Av. Boavista 1043
4100-129 Porto
Tlf: +351 226054872
Fax: +351 226054874
Email: alexandre.sousa@paradigmaxis.pt

Orientador na FEUP:

João Correia Lopes
DEEC
Faculdade de Engenharia
Universidade do Porto
Tlf: +351 225081759
Fax: +351 225081443
Email: jlopes@fe.up.pt

Copyright © 2005 ParadigmaXis, SA
Reservados todos os direitos
Reprodução autorizada, mediante indicação da fonte

Abstract

This report registers the training made at the ParadigmaXis company for the *Estágio Curricular* of the course *Licenciatura em Engenharia Informática e Computação*. The project subject is “*Sistema de Apoio à Execução, Auditoria e Decisão do Projecto Porto Digital*” and as the subject indicates, this training is included in the *Porto Digital* project.

The *Porto Digital* project is the result of Porto’s city application to the governmental project *Portugal Digital* and its purpose is to promote the development and use of Informatics and Communications at Porto. As a *Portugal Digital* project, the *Porto Digital* project will have a co-financing of 75% of all expenditures presented by a project partner. 50% of the expenditures will be paid by ERDF¹ and 25% paid by Portuguese Government, both through POS_C². However, all the process of presentation, confirmation and payment of the request of the co-financing has been really slow in other *Portugal Digital* projects. It has been verified that this delay creates financial problems to project partners. If this facts happens in *Porto Digital* project, it will compromise the achievement of the ambitious goals defined.

Until now, the refunding request process was done by exchanging *Excel* sheets through e-mail accounts. These *Excel* sheets had a lot of mistakes. To improve the performance it has been proposed a system that could decrease the inaccuracies rate and the numbers of iterations until the approval of the refunding request process.

C++ was chosen as the development language for the technological base. XML was chosen for the support data and the library Qt to develop the graphical use interface design of the application. For the development of the application it was used the *QtDesigner* tool. This tool was developed by Trolltech. Due to the utilization of XML and to its readable character, it was used the *Blowfish* algorithm for the encryption of the data made persistent in the file system. Otherwise any user with suitable operation system permission could change the information of the files.

Due to the delay of the *Porto Digital* process starting, not all goals of the project were achieved at this moment. Nevertheless, it is now a guarantee that technologies and design decision were correct and fit with the problem presented. Now, it is necessary to improve a few more features and finish the security mechanisms.

In spite of all the natural difficulties with the first contact the with labour market, the contact with professional work and the knowledge acquired made this experience an important complement to the training received at Faculdade de Engenharia da Universidade do Porto.

¹European Region Development Fund

²*Programa Operacional da Sociedade do Conhecimento*

Resumo

Este relatório documenta o estágio realizado na empresa ParadigmaXis, para a disciplina de Estágio Curricular da Licenciatura em Engenharia Informática e Computação. O projecto tem como tema “*Sistema de Apoio à Execução, Auditoria e Decisão do Projecto Porto Digital*” e como o próprio tema indica, enquadra-se no âmbito do Projecto Porto Digital.

O Projecto Porto Digital é o resultado da candidatura da cidade do Porto ao projecto governamental Portugal Digital e tem como objectivo fomentar o desenvolvimento e utilização de Tecnologias de Informação e Comunicação na cidade do Porto. Com a aprovação do Projecto Porto Digital no âmbito do projecto Portugal Digital ficou garantida a participação de 75% das despesas apresentadas pelos participantes no projecto. 50% da despesa será reembolsada pelo FEDER³ e 25% pelo OE⁴, ambas através do POS_C⁵. Contudo, todo o processo de apresentação, comprovação e pagamento dos pedidos de reembolso tem sido bastante lento noutros projectos no âmbito do projecto Portugal Digital. Constatase que este atraso tem criado problemas financeiros graves e que, caso se verifiquem no Projecto Porto Digital, este facto colocará em causa o cumprimento dos ambiciosos objectivos propostos.

Até a data, o processamento dos pedidos de reembolso era processado através do envio de folhas de *Excel* tendo-se verificado altas taxas de erros na introdução dos dados. Para melhorar a eficiência foi proposto um sistema que possibilite a diminuição de erros na introdução dos dados e permita diminuir o número de iterações até à aprovação do pedido de reembolso.

Para base tecnológica foi escolhido o *C++* como linguagem de desenvolvimento, o *XML* como repositório de dados e a biblioteca *Qt* para desenho da interface gráfica da aplicação. Para o desenvolvimento da aplicação foi utilizada a ferramenta *QtDesigner* disponibilizada pela Trolltech. Devido à utilização de *XML* e ao facto de ser legível por humanos usouse o algoritmo *Blowfish* para encriptação dos dados, tornados persistentes no sistema de ficheiros. Caso contrário, qualquer utilizador da máquina com privilégios adequados no sistema operativo poderia alterar a informação dos ficheiros.

Devido ao atraso no arranque do Projecto Porto Digital, a implementação encontra-se com algum atraso. Contudo, foi já possível verificar que as propostas tecnológicas e arquitectónicas são adequadas ao problema apresentado. Neste momento apenas é necessário o desenvolvimento de mais algumas funcionalidades e terminar os mecanismos de segurança de persistência e comunicação de dados.

Apesar das dificuldades naturais do primeiro contacto com o mundo de trabalho, o contacto com profissionais experientes e os conhecimentos adquiridos fazem desta experiência um importante complemento à formação recebida na Faculdade de Engenharia da Universidade do Porto.

³ *Fundo Europeu para o Desenvolvimento Regional*

⁴ *Orçamento de Estado*

⁵ *Programa Operacional da Sociedade do Conhecimento*

Agradecimentos

Agradeço:

À ParadigmaXis, que me proporcionou este estágio e disponibilizou todos os recursos necessários para a realização do mesmo;

À Faculdade de Engenharia da Universidade do Porto, que é um marco incontornável na minha formação tanto profissional como pessoal e a que me orgulho de pertencer;

Ao Engenheiro Alexandre Valente de Sousa, meu orientador na ParadigmaXis;

Ao Professor Correia Lopes, não só por ter aceite ser meu orientador, mas também pelo seu apoio e disponibilidade ao longo do estágio;

Ao Professor J. Magalhães Cruz, pela disponibilidade prestada sempre que se colocaram questões de segurança informática;

Aos meus colegas na ParadigmaXis, que me receberam de forma amistosa e sempre me trataram como um deles;

Ao Eng. Filipe Correia, pelo apoio arquitectónico e tecnológico.

À minha família e amigos, que sempre acreditaram em mim e me apoiaram incondicionalmente;

Aos meus colegas da LEIC, com quem nos últimos cinco anos partilhei óptimos momentos, não só em actividades académicas, como também nas intermináveis noites de trabalho na faculdade;

A todos o meu muito obrigado.

Conteúdo

Abstract	ii
Resumo	iii
Agradecimentos	iv
1 Introdução	2
1.1 A Empresa — ParadigmaXis, SA	2
1.2 O Projecto Porto Digital	2
1.3 Gestão de Pedidos de Reembolso	4
1.4 Objectivos de Estágio	5
1.5 Planeamento	5
1.6 Modelo de Desenvolvimento	6
1.7 Organização do Documento	7
2 Revisão Tecnológica	9
2.1 Introdução	9
2.2 PHP — Pré-Processador de Hipertexto	9
2.3 MySQL	10
2.4 Javascript	11
2.5 C/C++	11
2.6 Qt	13
2.6.1 QtDesigner	13
2.7 XML	14
2.8 SAX e DOM	17
2.9 Resumo	17
3 Requisitos da Aplicação a Desenvolver	20
3.1 Introdução	20
3.2 Requisitos Funcionais	22
3.2.1 Actores	23
3.2.1.1 Parceiro	23
3.2.1.2 APD	24
3.2.1.3 POS_C	24
3.2.1.4 Utilizador Geral	24
3.3 Requisitos Não Funcionais	26
3.3.1 Fiabilidade	26
3.3.2 Portabilidade	26
3.3.3 Desempenho	26

3.3.4	Usabilidade	27
3.3.5	Segurança	27
3.3.6	Escalabilidade	27
3.4	Resumo	27
4	Arquitectura do Aplicação a Desenvolver	28
4.1	Introdução	28
4.2	Arquitectura Lógica	29
4.3	Arquitectura Física	29
4.4	Resumo	33
5	Implementação	34
5.1	Introdução	34
5.2	Repositórios de Dados	34
5.2.1	Estrutura DOM	35
5.2.2	Verificação da Estrutura do DOM	39
5.2.3	Protecção dos Dados	39
5.3	Mecanismos Importantes	42
5.3.1	Autenticação	42
5.3.1.1	Algoritmo MDA	44
5.3.2	Verificação de Dados	45
5.3.3	Consistência e Persistência de Dados	47
5.3.3.1	Gravação	47
5.3.3.2	Estrutura do DOM de Alterações	48
5.4	Funcionalidades	48
5.4.1	Pesquisas e Filtros	48
5.4.1.1	Pesquisas	48
5.4.1.2	Filtros	51
5.4.2	Sincronização	52
5.4.3	Relatórios	54
5.5	Interface	55
5.6	Resumo	57
6	Melhoramentos	59
6.1	Introdução	59
6.2	Gestão de Memória	59
6.3	Mecanismos de Segurança	60
6.4	Mecanismos para Depuração de Erros	61
6.5	Testes Unitários	63
6.6	Relatórios	63
6.7	Resumo	65
7	Conclusões e Trabalho Futuro	67
	Bibliografia	70
	Glossário	73

A	Listagem dos Ficheiros XSD Schema	80
A.1	XSD Schema: rubricas.xsd	80
A.2	XSD Schema: documentos.xsd	82
A.3	XSD Schema: fornecedor.xsd	83
A.4	XSD Schema: pedido.xsd	85
A.5	XSD Schema: alteracoesFor.xsd	89
A.6	XSD Schema: alteracoesPP.xsd	92
A.7	XSD Schema: utilizador.xsd	95
Índices		96
I.8	Índice de Fragmentos de Código	96
I.9	Índice de Citações	97
I.10	Índice de Ficheiros	98
I.11	Índice Remissivo	98

Lista de Figuras

1.1	Planeamento do Estágio	6
1.2	Modelo de Desenvolvimento em Cascata	7
3.1	Modelo de Objectos do Domínio	21
3.2	Diagrama de Casos de Utilização	22
3.3	Diagrama de Casos de Utilização Disponíveis no Módulo Pedidos de Reembolso	23
3.4	Diagrama de Casos de Utilização Disponíveis no Módulo Gerir Fornecedores	24
3.5	Diagrama de Casos de Utilização Disponíveis no Módulo Gerir Fornecedores Porto Digital	25
3.6	Diagrama de Casos de Utilização Disponíveis no Módulo Gerir Pedidos de Pagamento	25
3.7	Diagrama de Casos de Utilização Disponíveis Através do Módulo Consultas	26
4.1	Modelo da Arquitectura	29
4.2	Modelo de Camadas Lógicas	30
4.3	Diagrama de Componentes	31
4.4	Diagrama de Distribuição	32
5.1	Diagrama da Estrutura de Documentos <i>XML</i> Válidos de RÚBRICAS	36
5.2	Diagrama da Estrutura de Documentos <i>XML</i> Válidos de PEDIDOS DE RE- EMBOLSO	37
5.3	Formulário para Mudar <i>Palavra-chave</i>	44
5.4	Diagrama de Estados de uma Pesquisa	49
5.5	Diagrama de Estados de um Filtro	52
5.6	Navegador da Aplicação	56
5.7	Barra de Ferramentas	57
5.8	Janela de FORNECEDORES	58
A.1	Diagrama da Estrutura de Documentos <i>XML</i> Válidos de RÚBRICAS	81
A.2	Diagrama da Estrutura de Documentos <i>XML</i> Válidos de DOCUMENTOS . .	82
A.3	Diagrama da Estrutura de Documentos <i>XML</i> Válidos de FORNECEDORES .	84
A.4	Diagrama da Estrutura de Documentos <i>XML</i> Válidos de PEDIDOS DE RE- EMBOLSO	88
A.5	Diagrama da Estrutura de Documentos <i>XML</i> Válidos de Alterações sobre FORNECEDORES	91
A.6	Diagrama da Estrutura de Documentos <i>XML</i> Válidos de Alterações sobre PEDIDOS DE REEMBOLSO	94
A.7	Diagrama da Estrutura de Documentos <i>XML</i> Válidos de UTILIZADORES .	95

Lista de Tabelas

2.1	Diferenças Fundamentais entre as duas <i>APIs</i> , <i>SAX</i> e <i>DOM</i>	18
5.1	Descrição das Restrições “ <i>Duras</i> ” Implementadas	46
5.2	Tabela das Teclas de Atalho	57
6.1	Exemplo de um Relatório para um Sub-projecto	64
6.2	Exemplo de um Relatório da Conta Corrente do <i>Parceiro 1</i>	66

Capítulo 1

Introdução

Este primeiro capítulo introduz o presente documento, assim como o contexto e os objectivos do estágio. Para tal é descrita a empresa, o contexto em que se inseriu o estágio, o problema proposto, o planeamento e a metodologia utilizada no desenvolvimento do projecto. Finalmente, é descrita a organização deste relatório, com uma breve indicação do conteúdo de cada capítulo.

1.1 A Empresa — ParadigmaXis, SA

A ParadigmaXis SA é uma empresa de capitais portugueses fundada em 2000. O seu *core business* centra-se na integração de sistemas de *software* dedicados ao processamento de informação, cujo objectivo é criar soluções à medida, de elevada qualidade, recorrendo a práticas de engenharia de *software* e reutilizando componentes normalizados. Para tal, recorre à prática conjunta de *eXtreme Programming* [Beck 2000], reutilização de *software*, paradigma de orientação a objectos, padrões de *software* [Gamma et al. 1995] e programação literária [Ramsey 1994].

Inserida num mercado altamente competitivo e especializado, a ParadigmaXis considera fundamental, para um crescimento sustentado da sua actividade, a actuação em rede. No seguimento desta política, tem privilegiado o estabelecimento de parcerias com um vasto conjunto de empresas do ramo das Tecnologias de Informação e Comunicação (TICs). Neste sentido, em Maio de 2001, a ParadigmaXis, em parceria com a Procme — Gestão Global de Projectos DOMA, realizou um *spin-off* originando a Ubicuos — Sistema de Georeferenciação SA. Como parceiros encontram-se ainda a Sidereus SA, especializada em sistemas legados e criptografia, a Eurotux SA, especializada no planeamento, integração e concretização de sistema informáticos e soluções de conectividade e a InfoPortugal, vocacionada para a mobilidade, disponibilizando conteúdos de interesse turísticos ou informativos. Foram igualmente estabelecidos protocolos com a Oracle (*Oracle Partner Program*), Autodesk (*Autodesk Developer Network*) e OnTrack (produtos e serviços de recuperação de dados). Recentemente, foi criada no seio da ParadigmaXis SA, uma nova área de actuação dedicada ao desenvolvimento de *software* crítico para fins militares.

1.2 O Projecto Porto Digital

O projecto Porto Digital é fruto da candidatura da cidade do Porto ao programa Portugal Digital. Como principal objectivo, o projecto Porto Digital pretende contribuir de

forma decisiva para a evolução, não só do município do Porto, como de toda a sua Área Metropolitana, para uma Sociedade da Informação e do Conhecimento [Associação Porto Digital 2004]. Para este fim, agruparam-se no seio do projecto Porto Digital um conjunto de iniciativas interligadas, a serem implementadas num espaço temporal de 36 meses. As várias iniciativas do projecto Porto Digital estão agregadas em 13 sub-projectos, que estão agrupados em seis categorias:

- Sub-projectos Infra-estruturas — projectos que pretendem permitir o acesso ao mundo digital;
- Sub-projectos Dinamização — projectos que pretendem dinamizar a comunidade para o mundo digital e a utilização das novas tecnologias;
- Sub-projectos Acessibilidades — inclui apenas um projecto que tem por objectivo distribuir pela cidade pontos de acesso à Internet e aos serviços disponibilizados pelo Porto Digital;
- Sub-projectos de *e-Government* — projectos que pretendem reformular os processos administrativos, a criação do *e-cidadão* e a melhoria da qualidade de vida;
- Sub-projectos Sectoriais — projectos que têm uma intervenção ao nível de uma área sectorial, neste caso ao nível do emprego, da economia e da cultura;

Os diferentes sub-projectos do Porto Digital serão alojados em quatro pólos ou áreas de intervenção: Administração Local, Educação, Empresarial e Cidadão. Cada pólo tem uma entidade hospedeira responsável por articular os diferentes sub-projectos e por organizar actividades dinamizadoras que garantam a visibilidade e divulgação dos sub-projectos. Em suma, estas entidades funcionam como entidade coordenadora da área de intervenção respectiva.

De salientar que na base deste projecto estiveram quatro preocupações fundamentais:

1. o Indivíduo, pois é para ele que se dirigem as iniciativas do Porto Digital, sendo crítico que os cidadãos do Porto sintam que foram considerados os seus valores, as suas necessidades pessoais e as suas motivações individuais. É também fundamental, seguindo a escola escandinava, garantir que a introdução crescente de novas tecnologia no quotidiano das pessoas contribua para melhorar a sua qualidade de vida e não tenda a desumanizar a nossa sociedade;
2. a Sociedade, dada a importância de avaliar correctamente as evoluções que se pretendem introduzir nas instituições, bem como no modelo de sociedade que aspiramos. Pretendendo obter a universalidade no acesso às tecnologias de informação, e a multiplicação de conteúdos em português, o projecto tem presente o risco da info-exclusão e pretende contribuir para a criação de um quadro legal que salvaguarde a liberdade, a privacidade, a segurança e a dignidade dos membros da comunidade;
3. o Mercado, dada a necessidade de tornar a cidade do Porto competitiva de modo a atrair investimentos tecnológicos. Pela capacidade dinamizadora da economia e por permitir desenvolver áreas de competências e competitividade a nível internacional, é essencial atrair este tipo de investimento multiplicador de conhecimento e riqueza.
4. a Tecnologia, reflecte a necessidade de ver a tecnologia como um meio e não como um fim em si mesmo, englobando as infra-estruturas, a vertente de formação dos utilizadores e da utilização das aplicações.

Sempre regido pelas quatro preocupações agora apresentadas, e com o objectivo de contribuir para a alteração estrutural no modo de funcionamento e no modo de estar da cidade do Porto face aos desafios do mundo moderno, o Porto Digital pretende obter resultados em quatro áreas: Educação, Emprego, Burocracia, Qualidade de Vida. Do vasto conjunto de benefícios, quer para os cidadãos do Porto, quer para a própria cidade, destacam-se:

- a melhoria significativa da infra-estruturas de suporte à utilização das TICs na educação e nos laboratórios de Investigação e Desenvolvimento (I&D);
- o forte incremento na competitividade do tecido empresarial da cidade do Porto, concentrando toda a informação sobre a orientação vocacional e emprego num único ponto de interface;
- largo conjunto de sítios *Web* transaccionais e informativos;
- a melhoria da eficiência da Administração Pública e a sua aproximação aos cidadãos que serve;
- e, por fim, a melhoria da qualidade de vida urbana dos habitantes, trabalhadores e visitantes da cidade do Porto.

1.3 Gestão de Pedidos de Reembolso

Da experiência resultante de outras cidade digitais, nomeadamente dos projectos Maia Digital e Gaia Digital, constata-se que a execução das cidades digitais tem graves problemas financeiros do ponto de vista de tesouraria. Este problema surge, em parte, da dificuldade da entidade financiadora, o Programa Operacional da Sociedade do Conhecimento (POS_C), de efectivar os pagamentos atempadamente. A comparticipação de 75% das despesas por parte POS_C é garantida pelo Orçamento de Estado (OE) e pelo Fundo Europeu para o Desenvolvimento Regional (FEDER). O OE reembolsa 25% e o FEDER 50% das despesas efectuadas. Noutros projectos Cidades/Regiões Digitais, tem-se verificado uma maior dificuldade no reembolso da componente referente ao OE. Quer o atraso do pagamento, quer a maior facilidade em reembolsar a componente do FEDER são espectáveis durante a execução do projecto Porto Digital.

O problema torna-se crítico quando, às dificuldades de reembolso da entidade financiadora, é necessário considerar o período de processamento de um PEDIDO DE REEMBOLSO até este ser entregue ao POS_C. Antes de qualquer PEDIDO DE REEMBOLSO ser entregue ao POS_C é necessário que seja avaliada quer a elegibilidade da despesa quer a correcção dos dados e documentos constantes na FOLHA DE PEDIDO DE PAGAMENTO referentes a esse pedido. É comum, que os dados entregues por cada um dos Parceiros, para cada um dos projectos em que estão envolvidos, contenham várias incorrecções. Estes erros obrigam a que equipa de gestão da Cidade/Região Digital disponibilize os seus recursos para eliminá-los, ou então, rejeite o PEDIDO DE REEMBOLSO e o reenvie ao respectivo Parceiro para correcção. Em qualquer dos casos, esta situação provoca atrasos e agrava o problema de tesouraria encontrado em muitos projectos no âmbito do Portugal Digital.

O problema do atraso no reembolso das despesas efectuadas no âmbito do projecto Porto Digital é que este coloca problemas na exigência ao Parceiro do cumprimento dos compromissos de trabalho assumidos. De salientar que este problema também poderá conduzir à tomada de decisões tecnicamente menos correctas para se evitar recorrer a um Parceiro

sem grande capacidade financeira, mas com custos mais baixos e mais empenhado, em prol de um Parceiro com uma maior envergadura financeira, mas com uma relação de benefício pior. Em muitos projectos do Portugal Digital, os Parceiros, numa atitude defensiva, têm executado uma parte dos projectos em que estão envolvidos e exigido o recebimento do reembolso para prosseguirem o projecto. Esta situação tem condicionado fortemente o desempenho de várias Cidades/Regiões Digitais e, garantidamente, os objectivos ambiciosos do Porto Digital não poderão ser atingidos se esta for a política de execução adoptada [Associação Porto Digital 2004].

1.4 Objectivos de Estágio

Este estágio tem por objectivo o estudo, desenho e implementação de uma aplicação que possa gerir o processamento de PEDIDOS DE REEMBOLSO dos vários Parceiros do projecto Porto Digital à entidade financiadora, o POS.C.

Com esta aplicação pretende-se diminuir o tempo entre a entrega de um PEDIDO DE REEMBOLSO por parte de um Parceiro e o momento de reembolso por parte do POS.C. A melhoria de eficiência no processamento de pagamento será baseada em dois vectores. O primeiro, a diminuição da taxa de erros no preenchimento das folhas de cálculo por parte dos Parceiros e da equipa de gestão da Associação Porto Digital (APD). O segundo, a diminuição do tempo de tratamento dos dados enviados pelos Parceiros e processados na Associação Porto Digital para posterior envio para a entidade financiadora. Desta forma, libertar-se-ão recursos para as tarefas de avaliação da elegibilidade das despesas e controle e actualização de orçamentos, ou seja, a actividades analíticas, necessariamente realizadas pelo Homem.

Para o desenvolvimento deste sistema serão utilizadas tecnologias de representação de informação, comunicação de dados, encriptação simétrica e assimétrica e programação orientada a objectos.

1.5 Planeamento

O sucesso em qualquer projecto, sobretudo de engenharia de *software*, é baseado num bom planeamento e cuidado acompanhamento do seu cumprimento. Com o objectivo de minimizar os riscos deste estágio, antecipando possíveis problemas e preparando soluções para os mesmos, foi definido pelos seus intervenientes um plano de acção. De referir que o planeamento de qualquer projecto não é, nem deve ser, rígido. Pelo contrário, deve evoluir ao longo do projecto à medida que mais e melhor informação sobre as tarefas, recursos e objectivos são obtidas pelos intervenientes [Sommerville 1995]. O plano de actividades cumpridas durante este estágio é apresentado na Figura 1.1.

O estágio iniciou-se dia 7 de Março e as primeiras seis semanas foram dedicadas à integração na empresa e ao projecto Porto Digital. Ao estudo do projecto Porto Digital foram dedicadas as duas primeiras semanas, sendo que a segunda semana, o estudo do projecto Porto Digital decorreu em paralelo com o estudo da solução implementada no sítio *Web CityHelp* e das tecnologias utilizadas: *PHP*, *MySQL* e *Javascript*. A actualização deste sítio *Web* deveu-se ao atraso inicial no arranque do projecto Porto Digital. A terceira semana, dedicada ao estudo das tecnologias utilizadas na implementação do sítio *Web CityHelp* e às primeiras alterações solicitadas pela empresa, decorreu entre 21 e 25 de Março. A concretização de todas as alterações prolongou-se até ao final da quarta semana,

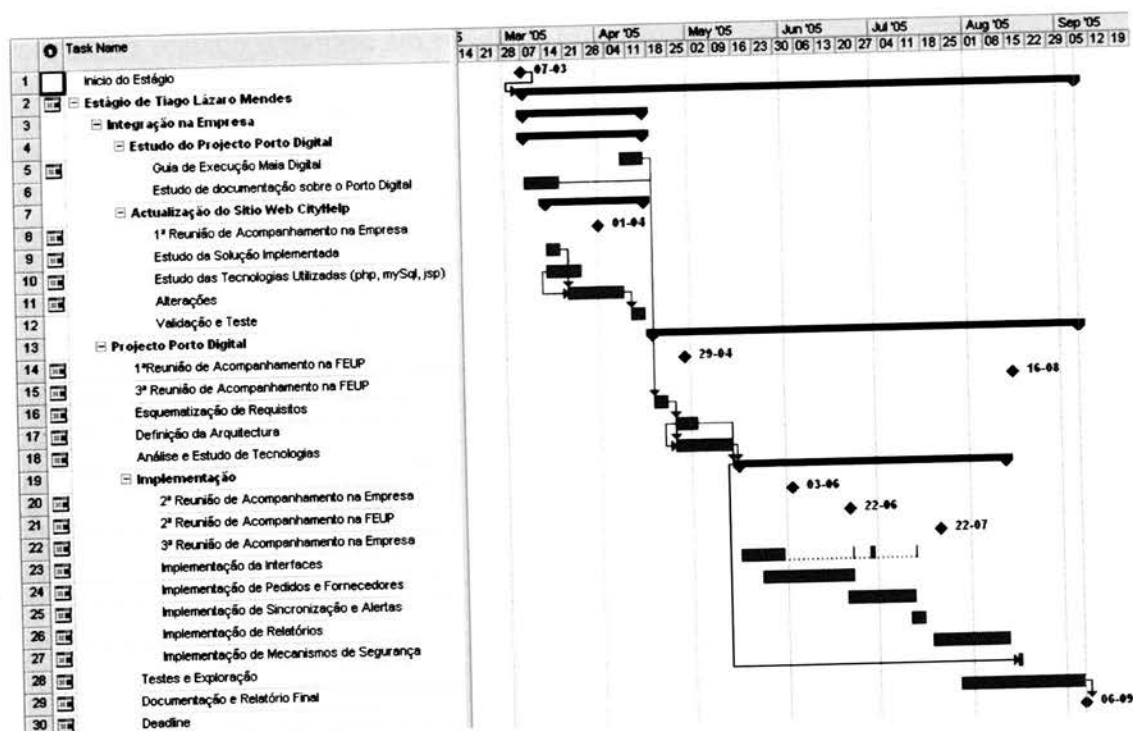


Figura 1.1: Planeamento do Estágio

8 de Abril. A última semana desta primeira fase do estágio foi reservada ao estudo do guia de execução do projecto Maia Digital e à validação e teste das alterações realizadas ao sítio *Web* do *CityHelp*. Estes testes foram realizados nas mesmas condições da página em produção, estando a página e a base de dados de suporte a serem executadas no servidor.

Tendo resultado do estudo da documentação do Porto Digital e do guia de execução do Maia Digital a compreensão dos requisitos da aplicação a desenvolver, as seguintes 17 semanas foram dedicadas à identificação dos requisitos, à definição da arquitectura e estudo de tecnologias, à implementação do protótipo da aplicação e a testes de exploração. Devido às derrapagens nos prazos de finalização das actividades precedentes, a actividade de Testes e Exploração resumiu-se à pesquisa e documentação das soluções de testes unitários disponíveis para a tecnologia utilizada. Deste período de dezassete semanas, as últimas três foram também dedicadas à estruturação da informação necessária à escrita da documentação relativa ao projecto. As três últimas semanas do estágios foram dedicadas exclusivamente à escrita deste documento e de documentação para a empresa.

1.6 Modelo de Desenvolvimento

Um modelo de desenvolvimento de *software* é um conjunto de actividades estruturadas para levar a cabo a implementação de uma aplicação ou solução tecnológica [Sommerville 1995]. Existe um vasto conjunto de modelos de *software* que se aplicam consoante as características do projecto e da aplicação a desenvolver. No desenvolvimento do sistema inerente à realização deste estágio foi seguido o modelo em cascata. A escolha deste modelo deveu-se ao facto de no modelo em cascata as fases de desenvolvimento — análise, desenho, implementação e teste — estarem perfeitamente definidas. À partida também se

poderia ter seguido o modelo em espiral. O modelo em espiral assume que o processo de desenvolvimento ocorre em ciclos, cada um contendo fases de avaliação e planeamento. O modelo em espiral organiza o desenvolvimento de *software* num processo iterativo em que vários conjuntos de quatro fases se sucedem até se obter o sistema final. Um ciclo inicia-se com a identificação dos objectivos, soluções e restrições dessa fase do projecto. Na segunda fase do ciclo é analisada e escolhida a melhor solução para alcançar o objectivo desse ciclo. Nesta fase é também realizada uma análise de risco. Na terceira fase do ciclo ocorre o desenvolvimento do produto e na quarta fase é avaliado o resultado alcançado e prepara-se o início do ciclo seguinte. Ainda que esta solução podesse diminuir o risco do projecto, a impossibilidade de acompanhamento contínuo por parte do coordenador da empresa, poderia causar paragens no desenvolvimento nas fases de avaliação e definição de ciclos ou, por outro lado, a definição de ciclos muito longos, o que levaria a um desenvolvimento em cascata.

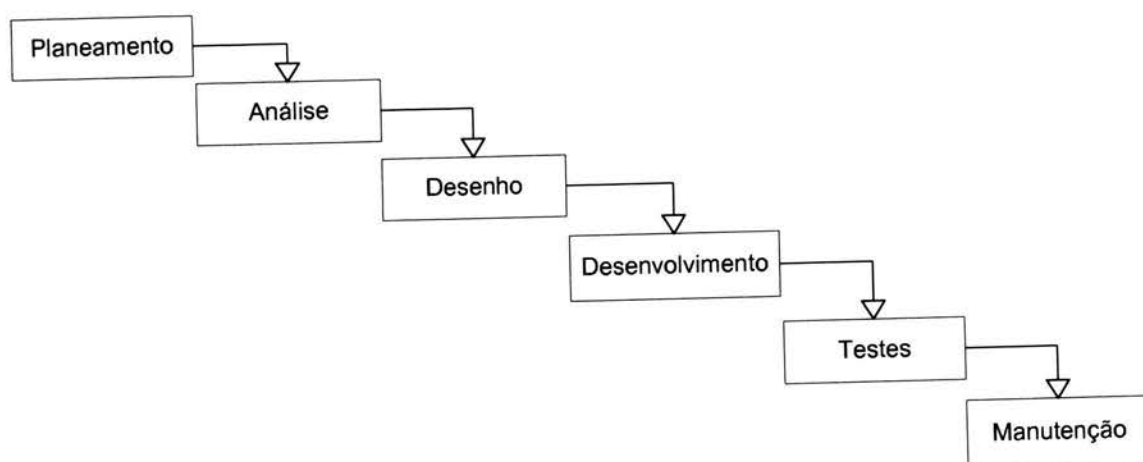


Figura 1.2: Modelo de Desenvolvimento em Cascata

De salientar que o planeamento do estágio apresentado na secção anterior foi definido em coerência com este modelo de desenvolvimento.

1.7 Organização do Documento

O presente documento está estruturado em sete capítulos e um anexo.

Este capítulo é dedicado à introdução do contexto em que foi realizado o estágio, bem como os seus objectivos e a metodologia seguida para os alcançar.

No segundo capítulo — Revisão Tecnológica — são descritas as tecnologias investigadas durante a realização deste estágio. Algumas não se aplicam directamente à aplicação desenvolvida porque foram estudadas na fase de integração na empresa e aplicadas na actualização do sítio *Web CityHelp*. Para todas as tecnologias são apresentadas as suas principais características e a relevância da sua utilização. Também é indicado se a sua utilização foi no âmbito específico do desenvolvimento do protótipo criado neste estágio, ou na fase de integração na empresa.

No capítulo três — Requisitos da Aplicação a Desenvolver — é apresentado o contexto que levou ao aparecimento do problema. Simultaneamente são descritos os principais casos

de utilização, assim como os requisitos funcionais e não funcionais da aplicação.

De seguida, no capítulo quatro — Arquitectura da Aplicação a Desenvolver — apresenta-se uma solução para uma aplicação que respeita os requisitos do problema apresentados no capítulo anterior. A arquitectura é descrita através da arquitectura lógica e física usando a linguagem *UML*.

No quinto capítulo — Implementação — são descritos os detalhes da implementação da aplicação definida, bem como o modo como as tecnologias foram utilizadas em cada parte da aplicação.

Nos dois últimos capítulos deste documento são analisados os resultados obtidos com o estágio e com a aplicação que foi desenvolvida.

No capítulo seis — Melhoramentos — é feita uma análise do protótipo desenvolvido e são propostos futuros melhoramentos.

O último capítulo — Conclusões e Trabalho Futuro — apresenta as conclusões obtidas no final do estágio, destacando os objectivos cumpridos e conhecimentos adquiridos, bem como as principais dificuldades encontradas e o trabalho futuro.

Em anexo encontra-se a listagem de ficheiros *XSD Schema* que validam os ficheiros *XML* com que a aplicação está preparada para trabalhar. São também apresentados os diagramas da estrutura desses ficheiros *XML*.

Capítulo 2

Revisão Tecnológica

Neste capítulo são apresentadas as tecnologias investigadas durante a realização deste estágio, salientando-se a sua relevância para o projecto e para as áreas em que foram utilizadas.

2.1 Introdução

Dado o atraso inicial verificado no arranque operacional do projecto Porto Digital, típico num projecto desta dimensão, foi-me atribuída no âmbito da empresa ParadigmaXis a responsabilidade de actualizar o sítio *Web* do sistema *CityHelp*, assim como iniciar a implementação do módulo de administrador. Durante este período estive em contacto com três tecnológicas tipicamente utilizadas em aplicações *Web*: *PHP*, *MySQL* e *Javascript*. Já no âmbito da actividade no seio da Associação Porto Digital, o leque de tecnologias estudadas durante a realização deste estágio foi aberto ao *C/C++*, *Qt/QtDesigner* e *XML*.

2.2 PHP — Pré-Processador de Hipertexto

O *PHP* (Pré-Processador de Hipertexto) é uma linguagem de programação para geração de documentos *HTML* [Pereira e Poupá 2004]. Criada por Rasmus Lerdorf em 1994, inicialmente o *PHP* foi utilizado para obter informações sobre o número de visitas à sua página pessoal. A primeira versão disponibilizada para uso geral só surgiu em 1995 ainda sobre o nome de *Personal Home Page Tools*. Nesse mesmo ano, o interpretador foi reescrito recebendo o nome *PHP*.

O *PHP* é uma linguagem de *scripting*, interpretada e executada do lado do servidor, sendo embebida no código *HTML* da página *Web*. Quando um utilizador abre a página em *PHP*, o servidor processa os comandos *PHP* e envia o resultado para o navegador do visitante. Ao ser executado no servidor, o *PHP* possibilita, por exemplo, o acesso a bases de dados. O *PHP* suporta uma ampla variedade de tecnologias de gestão de base de dados, tais como *dBase*, *Interbase*, *PostgreSQL*, *Sybase*, *ODBC*, *Oracle*, *mSQL*, *MySQL*, entre outras.

Tal como permite uma grande liberdade de escolha quanto à tecnologia de gestão de base de dados, o *PHP* também permite várias escolhas quer ao nível do sistema operativo, quer ao nível do servidor *Web*. Quanto ao sistema operativo, o *PHP* pode ser executado em qualquer das plataformas: *Unix*, *Linux*, *Windows*, *Solaris* e *HP-UX* [Pereira e Poupá 2004]. No lado do servidor, o *PHP* pode ser executado por diversos servidores, entre

outros, o *Apache*, o *IIS* (*Microsoft Internet Information Server*), o *Personal Web Server* e os servidores *iPlanet*.

Além da liberdade de escolha do sistema operativo, servidor e base de dados, o *PHP* permite a possibilidade de utilizar uma programação estruturada ou orientada a objectos.

Outra característica a realçar é a possibilidade de gerar imagens, ficheiros *PDF*¹ e animações *Flash* de forma dinâmica. O *PHP* também permite, através da utilização de extensões em código aberto (*software* livre), aumentar o potencial de utilização para fins mais específicos, tais como o pagamento *on-line*, protocolos de comunicação, instanciação de objectos *Java*, interpretador de documentos *XML*, conversão de datas, funções de pesquisa e compressão.

A versão 4.0, lançada em 2000, ao suportar sessões de forma nativa acabou por colmatar a falha mais apontada pelos seus críticos [Serrão e Marques 2001].

Apesar do *PHP* ser focado como uma linguagem interpretada (de *script*) do lado do servidor, o *PHP* pode ser utilizado também como [Daza 2005]:

- *script* de Linha de Comandos — pode-se fazer um *script PHP* funcionar sem se possuir um servidor *Web* ou navegador. Apenas é necessário o interpretador. Este tipo de uso é ideal para *scripts* executados usando o *cron* de *Linux* ou a agenda de tarefas (*Scheduled Tasks*) dos sistemas operativos *Windows*. Esses *scripts* podem ser utilizados também para rotinas de processamento de texto.
- ferramenta de desenvolvimento de aplicações com interfaces gráficas — o *PHP* não é, seguramente, a melhor linguagem para desenvolvimento de aplicações com interface gráfica. No entanto o *PHP-GTK* permite o desenvolvimento deste tipo de aplicações.

Pela sua portabilidade, facilidade de aprendizagem e massiva utilização na *Web*, o *PHP* foi a linguagem de programação aplicada para a implementação do sítio *Web* do *CityHelp*.

2.3 MySQL

MySQL é um servidor de base de dados poderoso, estável e extremamente eficiente, trabalhando com uma das linguagens de manipulação de dados mais utilizadas no mundo — a *SQL*². Com milhares de instalações no mundo inteiro é uma óptima solução para empresas e portais de pequeno e médio porte [Yarger et al. 1999]. Por este motivo foi usado na implementação do sítio *Web* do *CityHelp* para manutenção de dados referentes a vendas, clientes e suporte de informações sobre a aplicação e actualizações.

O *MySQL* foi criada por Michael Widenius na companhia sueca TcX. Em 1994 a TcX desenvolvia aplicações *Web* utilizando para gestão da base de dados uma ferramenta desenvolvida também por Michael Widenius, o *UNIREG*. Contudo, esta ferramenta criava demasiado “*overhead*” para ser bem sucedida na geração de páginas *Web* dinâmicas.

Partindo do legado da *mSQL*, a TcX decidiu criar uma linguagem que superasse a performance da *mSQL* que, na época, definia os padrões de performance das bases de dados [Yarger et al. 1999]. Para alcançar este objectivo o *MySQL* suporta multiprocessamento e multi-utilizador no acesso à base de dados. Também com o objectivo de garantir um rápido acesso aos dados armazenados, inicialmente o *MySQL* apenas disponibilizava 20% das funções *SQL*. Todavia, estes vinte por cento eram suficientes para implementar

¹ *Portable Document Format*

² *Structured Query Language*

80% das aplicações com acesso a bases de dados. Por exemplo, o *MySQL* não suportava tabelas temporárias, junções e transacções [MySQL 2004a] e [Russel e Stephens 2004] até à versão 4.1 [MySQL 2001] lançada no final de 2001, e *stored procedures* até à versão 5.1 [MySQL 2004b] lançada no início de 2005. Também suporta aplicações *stand-alone* e a maioria das principais linguagens de programação — *C*, *Java*, *Perl*, *Python*, *PHP* — têm suporte de acesso a dados armazenados em servidores *MySQL*.

2.4 Javascript

O *Javascript* é uma linguagem de programação criada especificamente para a Internet pela Netscape em 1995. Com o objectivo de possibilitar uma interactividade superior à que se consegue em aplicações implementadas apenas com *HTML*, a linguagem *Javascript* é embebida no *HTML* mas, ao contrário do *PHP*, é interpretada pelo navegador do cliente.

Orientada a eventos e acedendo ao *DOM*³ do navegador do cliente [Pereira e Poupá 2004], o *Javascript* pode servir como plataforma de integração com *Applets Java*, *ActiveX* e outros *plug-ins* de navegadores. Pequenas diferenças no modo de aceder ao *DOM* do navegador originaram diferentes implementações de *Javascript*. As duas principais implementações são a da Netscape (*JavaScript*) e da Microsoft (*JScript*). Em 1997 a *ECMA*⁴ apresenta o *ECMA-262* que define a norma para a linguagem *javascript*.

Esta tecnologia foi utilizada nas verificações durante a introdução de dados nos diversos formulários disponibilizados na página *Web* do *CityHelp*.

2.5 C/C++

Uma linguagem de programação tem dois objectivos: disponibilizar um meio para o programador especificar um conjunto de acções a serem executadas numa máquina; e definir um conjunto de conceitos que o programador utiliza para conceptualizar o problema e uma possível solução. O primeiro objectivo requer uma linguagem próxima da máquina para que as interacções com esta sejam simples e eficientes. A linguagem *C* foi desenhada com este objectivo. O segundo objectivo necessita de uma linguagem que esteja pensada para a descrição do problema, para que os conceitos da solução possam ser expressos de forma directa e concisa. As funcionalidades adicionadas ao *C* para a criação do *C++* foram criadas com este propósito [Stroustrup 1997].

A linguagem de programação *C* foi desenvolvida no início dos anos 70 nos laboratórios da Bell. Inicialmente desenhada como uma linguagem para o sistema *Unix*, rapidamente o seu número de utilizadores aumentou, expandindo-se para outros sistemas. A utilização da linguagem *C* em diversas plataformas conduziu à proliferação de várias variantes de *C*. As pequenas diferenças entre as várias variantes tornava impraticável o desenvolvimento em *C* de aplicações portáteis. Sendo premente a necessidade de uma norma para a linguagem *C*, o *ANSI*⁵ formou um comité, que em cooperação com a *ISO*⁶, publicou em 1990 a norma mundial de *C* — norma *ANSI/ISO 9899:1990* [Deitel e Deitel 2001].

Actualmente, a linguagem *C* está disponível na maioria dos computadores, é independente do *hardware* e, com alguma preocupação no desenho da aplicação, é possível escrever

³ *Document Object Model* - especifica o modo como os objectos de uma página *Web* são representados.

⁴ *European Computers Manufacturing Organization*

⁵ *American National Software Institute*

⁶ *International Standards Organization*

programas em *C* que são portáteis para a maioria dos computadores. Da mesma forma que a linguagem *BCPL* precedeu o *C*, o *C* foi a génese do *C++*. O *C*, com algumas excepções, pode ser visto como um subconjunto do *C++* tendo sido escolhido como a base do *C++* pelas seguintes razões [Stroustrup 1997]:

1. é versátil, sucinto e de baixo-nível;
2. é adequado para a maioria das tarefas dos sistemas de programação;
3. é executável em qualquer máquina, sobre qualquer sistema operativo;
4. adequa-se ao ambiente de desenvolvimento *Unix*.

Apesar de as linguagens *C* e *C++* partilharem a sintaxe, há bastantes diferenças entre elas. Enquanto que *C* é uma linguagem procedimental, o *C++* é uma linguagem orientada a objectos (*OOP*). Ambas as linguagens são expressivas e permissivas, mas o *C++*, ao ser mais expressivo, obriga ao mesmo tempo a um maior cuidado na definição dos objectos. O enfoque na estrutura do *C++* reflecte o forte crescimento da dimensão das aplicações desenvolvidas em *C*. Assim, o *C++* não foi construído tendo por base um vasto tipo de dados ou complexas operações primitivas [Stroustrup 1997]. Pelo contrário, a definição de operações e padrões é uma actividade de programação fundamental em *C++*.

Tal como *C*, o *C++* não foi desenhado especificamente para computação de dados numéricos. Ainda assim, dado o seu nível de eficiência, é utilizada por muitos matemáticos e engenheiros em aplicações científicas. O trabalho tradicional destes cientistas é muitas vezes combinado com gráficos ou componentes visuais que não são suportados nas linguagens tradicionais de computação numérica, por exemplo a linguagem *Fortran* [Barton e Nackman 1994]. Gráficos e interface de utilizadores são áreas onde o *C++* é usado massivamente.

No entanto, no ano de 1995 surgiu nesta área, uma linguagem capaz de rivalizar com o *C++*: o *Java*. Esta linguagem, desenvolvida na Sun Microsystems, teve um grande impacto no desenvolvimento de aplicações. Pelas suas potencialidades e cada vez mais larga comunidade de utilizadores, a utilização da linguagem *Java* deve ser analisada em qualquer projecto. Tal facto não se verificou neste projecto já que a utilização do *Qt* era um requisito tecnológico. O detrimento, logo à partida do *Java* em relação ao *Qt* deveu-se a receios no desempenho da aplicação em máquinas com poucos recursos. Apesar da diferença de performance entre aplicações desenvolvidas em *Java* e *C++* ser cada vez mais pequena, a arquitectura do *Java* é um entrave à sua eficiência. Apesar de esta arquitectura ser também o sucesso da portabilidade do *Java*, causa perda de eficiência já que o código fonte é compilado para *Java bytecode*, e é este código que é posteriormente executado pela *Java Virtual Machine*.

Como exemplo das capacidades do *C++* como linguagem de desenvolvimento de interfaces gráficas, pode-se referir que as primeiras interfaces, quer do sistema *Windows*, quer do sistema *Apple Macintosh* foram desenvolvidas em *C++* [Solin 2000]. Em *Unix* as principais classes que suportam o sistema *X* são desenvolvidas em *C++*. Por exemplo, o projecto *KDE*⁷, que tem por objectivo disponibilizar ambientes de trabalho em *Linux* amigáveis, estáveis, atractivos e consistentes, tem por base o *C++*. A base em *C++* é chamada através de uma poderosa biblioteca gráfica desenvolvida pela Trolltech, o *Qt* [KDE 2005].

⁷ *K Desktop Environment*

2.6 Qt

A forma como as pessoas utilizam os computadores mudou radicalmente nos últimos anos. Poucos são os utilizadores que ainda consideram recorrer a soluções em que a interacção com a aplicação se processe através de comandos executados numa consola. A criação de aplicações com interfaces gráficas tornou-se crítica para o desenvolvimentos de projectos bem sucedidos.

O *Qt* é uma linguagem de programação que possibilita o desenvolvimento de aplicações com interface gráfica. As bibliotecas *Qt* têm por base a linguagem de programação *C++*, sendo por este motivo orientada a objectos, facilmente extensível e permite programação por componentes. Criada por Havard Nord e Eirik Chambe-Eng, o *Qt* surge em Maio de 1995. Lançado comercialmente pela Trolltech no início de 1996, tem desde então sido disponibilizado através de duas licenças: uma licença comercial, *Qt Enterprise Edition*, para desenvolvimento de aplicações comerciais e outra livre, *Qt Open Source Edition*, para desenvolvimento de aplicações *open-source*.

Uma das principais vantagens do *Qt* em relação a outras ferramentas para o desenvolvimento de interfaces gráficas e linguagens de programação é a sua portabilidade. Com a utilização da biblioteca *Qt* é possível aos programadores ter apenas uma única fonte de código que depois é compilada para a maioria dos principais sistemas operativos disponíveis: do *Windows 95* ao *XP*, *Mac OS X*, *Linux*, *Solaris*, *HP-UX* e *Unix* com *X11*. Alias, o *Qt* não só permite a migração entre plataformas, ao disponibilizar e encapsular as estruturas das respectivas plataformas, como também as “emula” [Dalheimer 1999]. Esta característica do *Qt* permite desenvolver aplicações em *Linux* e executá-las com a aparência de uma aplicação *Windows* em qualquer sistema operativo da Microsoft. Assim é possível diminuir os riscos e custos de migração entre plataformas já que os utilizadores se sentirão seguros graças ao aspecto familiar da aplicação. De salientar, que o *Qt*, ao contrário de outras tecnologias de desenvolvimento de interfaces gráficas como o *Gtk+*, não só permite o desenvolvimento da interface como permite o desenvolvimento de toda a aplicação. Para isso o *Qt* disponibiliza soluções de comunicação, leitura e escrita de ficheiro, *APIs* para tratamento de documentos *XML* e todas as estruturas de dados típicas em qualquer linguagem de programação.

Outro factor para o sucesso que o *Qt* tem apresentado, deve-se à sua documentação de referência. Neste capítulo, a Trolltech disponibiliza uma documentação vasta e profunda sobre as classes, métodos e características da biblioteca *Qt*. Esta documentação pode ser consultada localmente em formato *HTML* ou no sítio *Web* da Trolltech⁸. A sua consulta é simples e rápida e, ao também disponibilizar algumas aplicações de exemplo, permite uma iniciação bastante simples à ferramenta [Solin 2000]. À medida que a utilização do *Qt* se tornou mais popular, surgiu a necessidade de ambientes de desenvolvimento que possibilitassem a implementação de interfaces gráficas através de funcionalidades gráficas. Existem, nesta área, várias soluções no mercado, como por exemplo, o *QtEz*, o *QtArchitect*, o *Ebuilder* e o *QtDesigner*.

2.6.1 QtDesigner

O *QtDesigner* é uma poderosa ferramenta de desenvolvimento de interfaces gráficas desenvolvida pela Trolltech. Suportado pelas bibliotecas gráficas *Qt*, permite um desenvolvimento rápido de interfaces gráficas “emulando” a aparência e o comportamento de uma

⁸<http://www.trolltech.com>

aplicação dos sistemas operativos que suporta.

O *QtDesigner* também permite desenvolvimento *stand-alone* ou integrado com outros *IDE's*⁹ como o *Microsoft Visual Studio .NET* em *Windows* ou o *KDevelop* em *Linux*. Contudo a integração com outros *IDE's* só está disponível para aplicações desenvolvidas na última versão da linguagem *Qt*, a versão 4.0 [TrollTech 2005].

As interfaces geradas pelo *QtDesigner* são definidas através de ficheiros *XML* que depois são compilados com os restantes ficheiros do projecto.

2.7 XML

Hoje em dia, é banal a associação das bases de dados relacionais a qualquer necessidade de armazenamento de dados. Todavia, há outras soluções a ter em conta neste contexto. O *XML eXtensible Markup Language* é uma delas.

As raízes do *XML* encontram-se em duas linguagens de marcação, o *SGML*¹⁰ e o *HTML*¹¹. A principal base do *XML* é o *SGML*, no entanto o *XML* herda algumas características do *HTML*. Tanto o *XML* como o *SGML* e o *HTML* são linguagens de marcação, no entanto, apenas o *XML* e o *SGML* se podem considerar meta-linguagens por permitirem a definição de outras linguagens. Por outro lado o *HTML* é uma linguagem concreta, não sendo possível estendê-la [Ramalho e Henriques 2002].

O *XML* é ideal para muitos tipos de aplicações. Contudo, as duas utilizações mais comuns são como formato de troca de dados e formato para publicação de dados [Bradley 2002]. Ou seja, contrariamente a outras tecnologias também destinadas ao armazenamento de dados, o formato *XML* possibilita a difusão e publicação dos dados.

A estruturação da informação consiste na representação de dados num formato padronizado, auto-descritivo, que tanto pode ser compreendido pelo ser humano como pela máquina. Esta representação auto-descritiva é alcançada através da marcação dos dados com etiquetas que caracterizam esse bloco de informação. O conjunto da abertura e fecho de uma etiqueta e a informação contida entre elas é um elemento. À meta-informação associada directamente a uma etiqueta podem estar também associados parâmetros adicionais chamados atributos.

Mas, para que um determinado bloco de informação esteja estruturado segundo *XML*, é necessário que siga normas bem definidas. Para que um documento *XML* seja considerado “bem formado” deve cumprir as seguintes regras:

1. a primeira declaração tem de indicar que se trata de um documento *XML* através da instrução `<?xml version="1.1" ?>`. A esta declaração podem ser adicionados atributos adicionais que caracterizem o documento, como por exemplo o tipo de codificação de caracteres;
2. as etiquetas têm de estar equilibradas, ou seja, a última etiqueta aberta tem de ser a primeira a ser fechada, por exemplo `<a><b> ... </b></a>`;
3. um elemento pode ser vazio sendo definido por uma única etiqueta que deve terminar com os caracteres `</>`, ou seja, do tipo `<nome atributo="valor"/>`. Contudo, um elemento que não seja vazio tem que ter sempre as duas etiquetas;

⁹ *Integrated Development Environment*

¹⁰ *Standard Generalized Markup Language*

¹¹ *Hypertext Markup Language*

4. os valores dos atributos têm de estar sempre entre aspas ou pelicas. Por exemplo,
`<nome atributo="valor"/>`;
5. tem um só elemento raiz.

De salientar que um documento XML é *case-sensitive*, sendo `<a>` e `<A>` duas etiquetas distintas. De seguida apresenta-se um documento XML “bem formado”.

15 *(Exemplo: Documento XML bem Formado 15)* ≡

```
<?xml version="1.0" encoding="iso-8859-1"?>
<projectos>
  <projecto codigo="INF01">
    <nome>Infraestrutura Física</nome>
    <categoria>Infraestrutura</categoria>
    <duracao>30</duracao>
    <entidade executora>UP</entidade executora>
    <parceiros>
      <parceiro>ADDP</parceiro>
      <parceiro>aeiou</parceiro>
      <parceiro>CM do Porto</parceiro>
      <parceiro>Eurotux</parceiro>
      <parceiro>IPP</parceiro>
      <parceiro>Metro</parceiro>
      <parceiro>UFP</parceiro>
      <parceiro>UPortucalense</parceiro>
    </parceiros>
  </projecto>
  <projecto codigo="GOV06">
    <nome>e-Cidadania</nome>
    <categoria></categoria>
    <duracao>30</duracao>
    <entidade executora>CM Porto </entidade executora>
    <parceiros>
      <parceiro>Bull</parceiro>
      <parceiro>Indra</parceiro>
      <parceiro>CPC</parceiro>
      <parceiro>UO</parceiro>
    </parceiros>
  </projecto>
</projectos>
```

Além de documentos “bem formados”, também existe o conceito de documentos “válidos”. Um documento diz-se “válido” quando estiver de acordo com um *DTD*¹² ou *XML Schema* associado. O *DTD* e o *XML Schema* definem as regras a que cada instância, documento *XML*, tem de obedecer para pertencer ao tipo de documento definido. Estas regras, dizem respeito não só à estrutura do documento, mas também ao domínio dos valores por omissão.

Quanto à descrição de um documento *XML*, esta pode ser orientada ao formato, à estrutura ou ao conteúdo [Ramalho e Henriques 2002]. A primeira abordagem implica a

¹² *Document Type Declaration*

marcação de um *XML* segundo o aspecto visual de um elemento, mas não o identifica nem especifica qual a sua função, assim não serão possíveis outras utilizações do documento. A descrição orientada à estrutura baseia-se na hierarquia genérica. Neste caso, considera-se o documento composto por secções e cada secção pode conter, ou não, novas secções. O uso abusivo deste tipo de anotação pode levar a situações em que elementos diferentes em termos de conteúdo sejam anotados com a mesma marca. Por este motivo, não será possível mais tarde ter um processamento orientado ao conteúdo. Já a orientação ao conteúdo é o coração da anotação genérica. A identificação dos elementos pelo seu conteúdo e propósito, em lugar da sua forma ou posição hierárquica, faz com que se atinja o expoente máximo da flexibilidade no contexto aplicacional [Ramalho e Henriques 2002].

Como suporte de armazenamento de dados, a tecnologia *XML* disponibiliza várias soluções [Ramalho e Henriques 2002] e [Vakali e Maddalena 2005]:

- a primeira solução, os documentos são armazenados no sistema de ficheiros do próprio sistema operativo do computador;
- a segunda solução passa por armazenar os documentos numa base de dados relacional. Ainda que esta solução possa parecer óbvia, a conversão dos documentos *XML* numa base de dados relacional não é trivial. Já tiveram lugar algumas tentativas nesta área, sendo a mais divulgada e utilizada, a proposta *MDA*¹³ da Omnimark¹⁴. Atentos a este problema, os principais construtores de base de dados relacionais já incorporaram algumas funcionalidades nos seus produtos que permitem lidar com documentos *XML*;
- outra possibilidade baseia-se na criação de uma base de dados orientada a objectos onde o modelo hierárquico modela directamente a árvore documental. Apesar da sua eficiência e havendo algumas dezenas de produtos no mercado, esta abordagem não é a mais vulgar, provavelmente pelo seu custo.
- a última solução de armazenamento de documentos *XML* baseia-se na criação de uma base de dados *XML* nativa. Este tipo de base de dados utiliza como unidade lógica de armazenamento um documento *XML*. Em termos de pesquisa sobre bases de dados *XML* nativas existem duas soluções dominantes:
 - *XML Query* (*XQuery*)¹⁵;
 - *XPath*¹⁶.

A maioria das bases de dados *XML* suportam indexação o que permite melhorar a eficiência das pesquisas. Ao permitir indexação, este tipo de base de dados tornam-se mais escaláveis e internamente podem funcionar com modelos proprietários, base de dados relacionais ou base de dados orientado a objectos.

De salientar que esta linguagem não é pertença de nenhuma empresa tendo sido desenvolvida por grupos de pessoas independentes sob a égide do *W3C*¹⁷ [W3C 2005]. A primeira recomendação para a definição desta linguagem surge em Fevereiro de 1998. A última versão, a versão 1.1 [Bray et al. 2004], foi concluída em Fevereiro de 2004.

¹³ *Micro-Document Architecture*

¹⁴ <http://www.omnimark.com>

¹⁵ <http://www.w3.org/xml/query>

¹⁶ <http://www.w3.org/tr/xpath>

¹⁷ *World Wide Web Consortium*

2.8 SAX e DOM

Apesar do vasto conjunto de vantagens e aplicações do formato *XML*, este torna-se frequentemente verboso e, deste modo, a sua utilização pouco optimizada quer em termos de memória, quer em termos de velocidade de leitura dos dados. Comparativamente com outros tipos de armazenamento o *XML* não é, à partida, o mais eficiente.

Mas, se inicialmente era necessário que as aplicações fizessem a análise gramatical dos ficheiros *XML*, o que impunha ao programador um trabalho suplementar, actualmente, existem duas *APIs*¹⁸ fundamentais que definem o funcionamento dos analisadores gramaticais de *XML* facilitando a sua utilização. Uma baseada numa estrutura em árvore — *DOM*¹⁹ — e outra, baseada em sequências de eventos — *SAX*²⁰.

O *DOM* [Wood et al. 1998] dá ao programador acesso à informação armazenada em *XML* na forma de um modelo hierárquico de objectos. Estes objectos correspondem a uma estrutura de dados em árvore. Preservando a ordem pela qual os elementos aparecem no *XML*, o *DOM* cria, para cada elemento do *XML*, um nó e uma lista de nós onde armazena os filhos desse nó. É com esta estrutura em árvore que se interage para aceder à informação contida no documento *XML*.

Já o *SAX* [Brownell 2002] possibilita o acesso aos dados por meio de lançamento e tratamentos de eventos, não sendo criado nenhum objecto que represente a informação. Estes eventos, gerados pelo analisador gramatical (*parser*), ocorrem durante a análise do documento *XML* e marcam acontecimentos como o início e fim do documento, abertura e fecho de etiquetas, secções de texto, comentários ou instruções de processamento. Uma funcionalidade do *SAX* de grande utilidade é a possibilidade de utilização de filtros. Deste modo, fazendo os dados passar por uma cadeia de filtros, é possível efectuar várias modificações aos dados numa única passagem [Connolly 1997]. Por este motivo o tempo de acesso em *SAX* é inferior ao necessário em *DOM* em que é necessário percorrer os dados por cada modificação que se pretenda concretizar.

O *DOM* é, sem dúvida, mais exigente com o sistema e necessita de mais memória disponível do que o *SAX*. No entanto, o *SAX* exige um maior esforço por parte do programador já que é necessário definir o modelo de objectos e tratar os eventos gerados pelo analisador. Estando ambas as *APIs* implementadas para várias linguagens e de utilização livre, cada uma pode ser particularmente útil mediante os requisitos da aplicação a desenvolver [Marchal 2001].

Na tabela 2.1 são apresentadas as principais diferenças entre as duas *APIs*, *SAX* e *DOM*.

2.9 Resumo

Numa fase inicial do estágio foram utilizadas as tecnologias *PHP*, *MySQL* e *Javascript*. O contacto com estas tecnologias não se verificou no âmbito do projecto de estágio, mas na actualização do sítio *Web* de uma solução disponibilizada pela empresa. Como já referido, esta responsabilidade foi-me atribuída devido a atrasos no arranque do projecto Porto Digital.

O *PHP* é uma linguagem de *scripting*, interpretada e executada do lado do servidor, sendo embebida no código da página *Web* para geração de documentos *HTML*. Ao ser

¹⁸ *Application Program Interface*

¹⁹ *Document Object Model*

²⁰ *Simple API for XML*

	<i>Simple API for XML</i>	<i>Document Object Model</i>
Fonte	Desenvolvido pela lista de distribuição <i>XML-Dev</i> .	É uma recomendação do <i>W3C</i> (<i>World Wide Web Consortium</i>).
Requer que o analisador gramatical	Analise o documento e gere os eventos necessários.	Analise o documento e organize os dados segundo uma estrutura em árvore de onde são acedidos.
Requer que o programador	Implemente funções para tratar os eventos e crie o modelo de objectos.	Interaja com a árvore para obter a informação.
Modelo de objectos	Definido pelo programador.	Árvore gerada pelo analisador gramatical.
Prós e contras.	Utilização inicial pouco intuitiva. Mais eficiente. Existe a possibilidade de guardar apenas determinados dados. Deste modo poupa-se o tempo e recursos gastos no processamento de dados que nunca serão necessários. O programador obtém o controlo sobre os dados durante o analisador gramatical o que se torna ideal para processamento sequencial.	Utilização inicial mais intuitiva. Menos eficiente. Mesmo os dados não necessários são guardados numa estrutura em memória. O maior uso de memória torna-o desaconselhável para ficheiros de grandes dimensões. O programador obtém o controlo sobre os dados após o analisador gramatical o que se torna ideal para o processamento não sequencial.

Tabela 2.1: Diferenças Fundamentais entre as duas APIs, *SAX* e *DOM*

executado no servidor, o *PHP* possibilita o acesso a várias base de dados. O *MySQL* é um servidor de base de dados poderoso, estável e extremamente eficiente. As principais linguagens de programação têm suporte de acesso a dados armazenados em servidores *MySQL*. O *Javascript* foi utilizado nas verificações necessárias durante a introdução dos dados nos formulários.

Já no âmbito do projecto Porto Digital foi estudada a linguagem de programação *C/C++* e *Qt*, a ferramenta de desenvolvimento de interfaces gráficas *QtDesigner*, e a linguagem de marcação *XML*.

A linguagem de *C* começou a ser utilizada no desenvolvimento de plataformas *Unix*. A rápida expansão desta linguagem sobre várias máquinas levou ao aparecimento de várias versões de *C* com incompatibilidades entre elas. Por este motivo era difícil o desenvolvimento de aplicações portáteis entre diferentes máquinas e para resolver este problema, em 1990, o *ANSI* definiu um conjunto de regras que todos os compiladores de *C* tinham de verificar. O *C* é uma linguagem de baixo nível, versátil, muito flexível e executável em qualquer máquina sobre qualquer sistema operativo. Por estes motivos foi utilizada como base para o desenvolvimento do *C++*.

Ao disponibilizar a possibilidade de desenvolvimento orientado a objectos, o *C++* permitiu a modelização de código e, desta forma, o aumento da produtividade comparativamente com as técnicas de programação até então. As potencialidades do *C++* foram aproveitadas por Haavard Nord e Eirick Chambe-Eng para o desenvolvimento do *Qt*.

O *Qt* é uma ferramenta para o desenvolvimento de aplicações com interfaces gráficas baseada em *C++*, orientada a objectos e que permite programação por componentes. A

sua principal vantagem é permitir ao programador ter apenas uma única fonte de código que depois é compilada para a maioria dos principais sistemas operativos: *Windows 95* ao *XP*, *Mac OS X*, *Linux Solaris*, *HP-UX* e *Unix* com *X11*. À medida que a utilização do *Qt* se tornou popular, surgiu a necessidade de uma ferramenta que possibilitasse a criação das interfaces gráficas através de funcionalidades gráficas. Com o objectivo de satisfazer esta necessidade a Trolltech criou o *QtDesigner*.

Suportado pelas bibliotecas gráficas do *Qt*, o *QtDesigner* não só permite o desenvolvimento *stand-alone* como também integrado com outros *IDE's* como o *Microsoft Visual Studio .NET* em *Windows* ou o *KDevelop* em *Linux*.

Apesar da associação das bases de dados a qualquer necessidade de armazenamento de dados, esta não é a única solução. Outra possibilidade é a utilização da linguagem de marcação *XML*. O *XML* é uma tecnologia desenvolvida pelo *W3C* como sendo um formato de intercambio e publicação de documentos. Ao basear-se na utilização de marcadores para caracterização da informação também pode funcionar como repositório de dados. Para leitura de ficheiro *XML* existem duas *APIs*: a *SAX* orientada a eventos e a *DOM* orientada ao objecto.

Capítulo 3

Requisitos da Aplicação a Desenvolver

Este capítulo apresenta não só o contexto em que surge o problema, como também os requisitos funcionais e não funcionais da aplicação a desenvolver, os seus actores e casos de utilização. Na descrição dos requisitos funcionais são utilizados diversos diagramas *UML*¹ [Rumbaugh et al. 1999] para formalizar a sua captura.

3.1 Introdução

A aplicação a desenvolver tem por objectivo simplificar um processo que em outras Cidades/Regiões Digitais tem causado alguns problemas na execução dos projectos: o processamento de PEDIDOS DE REEMBOLSO à entidade financiadora.

Pretende-se diminuir o tempo entre a entrega de um PEDIDO DE REEMBOLSO por parte de um Parceiro e o momento de pagamento por parte do Programa Operacional da Sociedade do Conhecimento (POS_C). Para tal, vai-se diminuir o número de iterações e intervenientes no processo. Ao contrário de outras cidades digitais, em que um Parceiro apresenta um PEDIDO DE REEMBOLSO ao Parceiro responsável pelo sub-projecto, no projecto Porto Digital, enviará o pedido directamente para a Associação Porto Digital (APD). Desta forma, o Parceiro processa todos os seus PEDIDOS DE REEMBOLSO simultaneamente, independentemente do sub-projecto a que se refere.

Apesar desta simplificação, é de esperar que após a entrega de um PEDIDO DE REEMBOLSO por parte de um Parceiro, este só seja efectivamente reembolsado, em média, 120 depois. Este período de tempo é justificado pela necessidade da Associação Porto Digital carimbar e fotocopiar os documentos de despesa apresentados por cada um dos 43 Parceiros, bem como a preencher a folha de cálculo que será posteriormente enviada para o Programa Operacional da Sociedade do Conhecimento.

O processo de recolha dos documentos e preenchimento e verificação da folha de cálculo deverá levar um mês. Terminado este processo é enviado para o POS_C a FOLHA DE PEDIDOS DE PAGAMENTO. Num processamento de uma FOLHA DE PEDIDOS DE PAGAMENTO por parte da APD ao POS_C é de esperar que o POS_C demore entre 45 a 90 dias a verificar a FOLHA DE PEDIDOS DE PAGAMENTOS e a confirmar que os valores estão correctos e são elegíveis. Assim se alcança um valor de 120 dias para o reembolso de uma despesa apresentada por um Parceiro.

¹ *Unified Modelling Language*

Neste caso não foram previstos erros na introdução de dados por parte do Parceiro ou por parte da APD, nem problemas quanto à elegibilidade das despesas. Na ocorrência deste tipo de erros todo o processamento de reembolso será atrasado, tendo-se verificado em outras cidades digitais, intervalos entre a realização da despesa e o seu pagamento, de um ano.

A melhoria que se ambiciona alcançar com esta aplicação terá por base dois factores. O primeiro, a diminuição da taxa de erros no preenchimento das folhas de cálculo por parte do Parceiro e Associação Porto Digital e, conseqüentemente, a diminuição do tempo de tratamento dos dados enviados pelos Parceiros e processados na Associação Porto Digital para posterior envio para a entidade financiadora. A aprovação pela entidade financiadora também será simplificada e informatizada com este sistema. O segundo, o tempo de envio da documentação que passa a ser efectuada via Internet.

Usando *UML* [Silva e Videira 2001], é apresentado e descrito na Figura 3.1 o diagrama de objectos do domínio da aplicação, sendo representadas as entidades envolvidas no processamento de PEDIDOS DE REEMBOLSO.

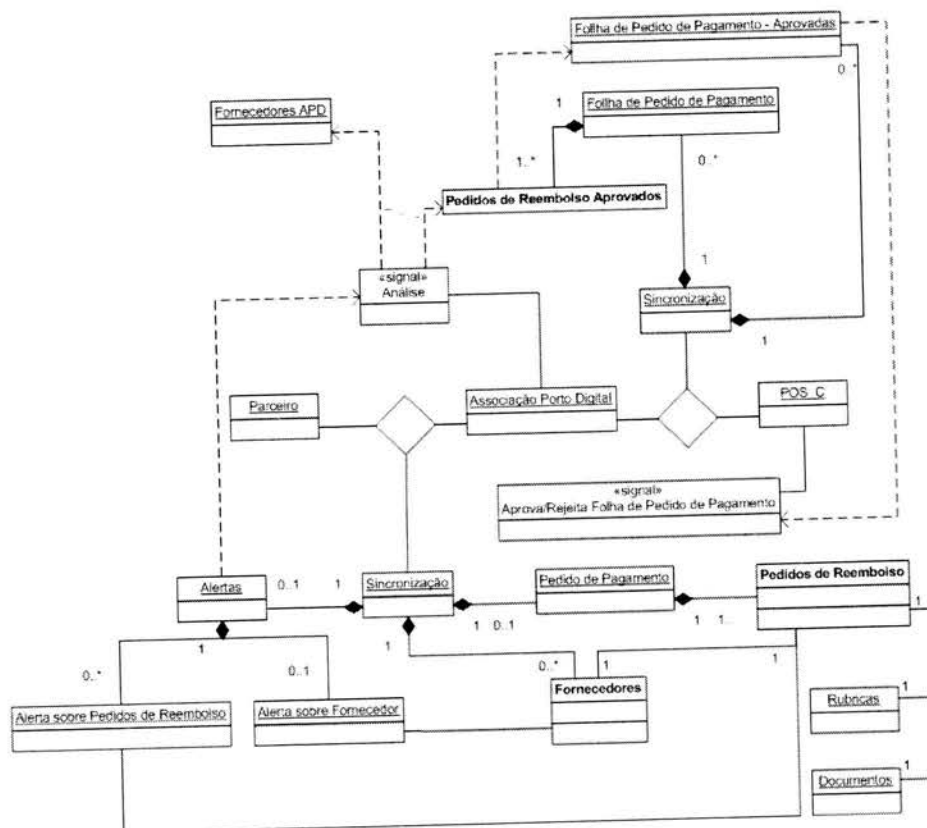


Figura 3.1: Modelo de Objectos do Domínio

Dado ser um projecto com um orçamento avultado e com grande volume de documentação, esta aplicação também fomentará uma gestão mais transparente e, ao criar um rasto documental dos PEDIDOS DE REEMBOLSO e FORNECEDORES envolvidos, simplificará o trabalho das auditorias que venham a ocorrer.

3.2 Requisitos Funcionais

Os casos de utilização são desenhados para definir o modo como os utilizadores interagem com o sistema. Por conseguinte, o diagrama de casos de utilização esquematiza todos os requisitos funcionais [Jacobson et al. 1999].

Assim, na Figura 3.2 é apresentado o diagrama geral de casos de utilização. De seguida serão analisados os actores do sistema e são descritos os casos de utilização que representam as funcionalidades que a aplicação deverá disponibilizar, ou seja, os requisitos funcionais.

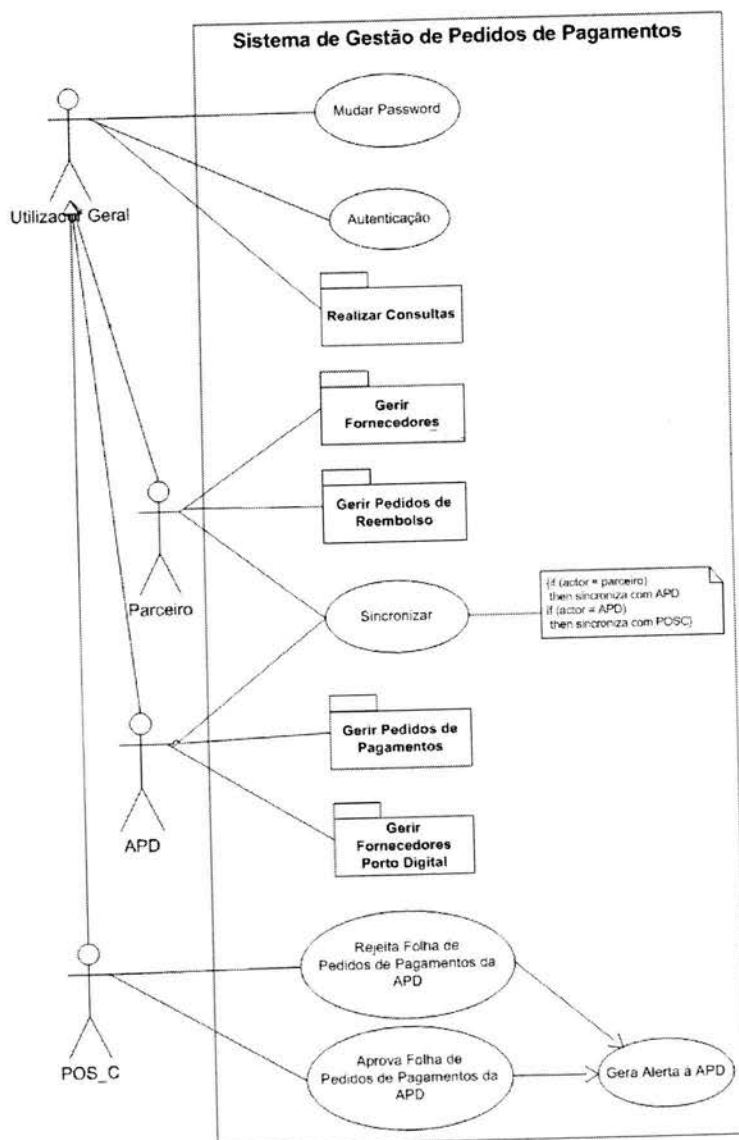


Figura 3.2: Diagrama de Casos de Utilização

3.2.1 Actores

No diagrama de casos de utilização estão representados quatro actores:

- Utilizador Geral;
- Parceiro;
- APD;
- POS_C.

3.2.1.1 Parceiro

O actor Parceiro representa cada um dos parceiros do projecto Porto Digital e tem acesso a todas as funcionalidades do módulo de **Gestão de Pedidos de Reembolso** e do módulo de **Gestão de Fornecedores**. O Parceiro pode também aceder a outro caso de uso, *Sincronizar*, permitindo-lhe sincronizar os dados sobre os PEDIDOS DE REEMBOLSO e FORNECEDORES com a Associação Porto Digital.

Através do módulo de **Gestão de Pedidos de Reembolso**, o Parceiro acede às funcionalidade de adição ou eliminação de linha, introdução de dados e sua verificação. Existem dois tipos de verificação. Uma verificação de restrições “suaves” que indicam ao utilizador potenciais erros nos dados introduzidos. Esta verificação só é realizada quando o utilizador a solicita. A outra, uma verificação de restrições “duras”, é executada sempre que o utilizador altera o valor de algum campo de um PEDIDO DE REEMBOLSO. Nesta verificação, a aplicação avisa o utilizador do facto e obriga-o a corrigir o erro.

Na Figura 3.3, é apresentado o diagrama que representa os casos de uso acedidos através deste módulo.

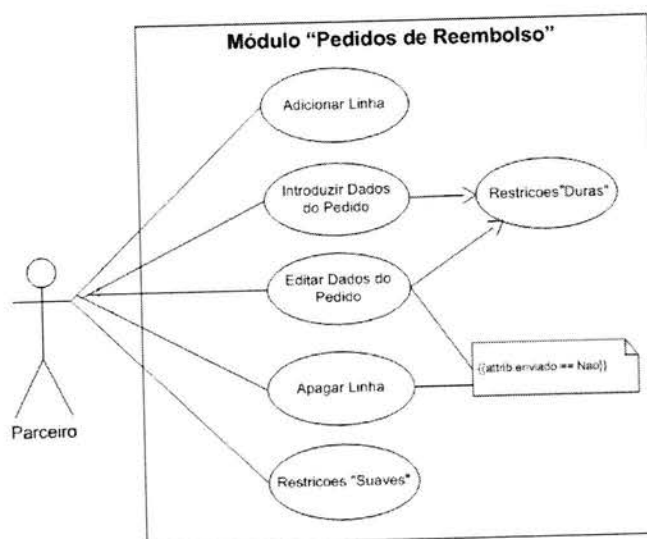


Figura 3.3: Diagrama de Casos de Utilização Disponíveis no Módulo Pedidos de Reembolso

Já através do módulo de **Gestão de Fornecedores** o parceiro acede às funcionalidades de adição, edição e eliminação de um FORNECEDOR. Nos casos de utilização de edição e eliminação de um FORNECEDOR, é necessário que os dados deste ainda não tenham sido

sincronizados com a Associação Porto Digital. Nesta situação os dois casos de utilização serão cancelados e o utilizador informado no caso de tentar aceder-lhes. Sempre que o utilizador adicionar um FORNECEDOR, os dados do FORNECEDOR serão validados e, caso ocorra algum erro ou incoerência de valores dos dados, o utilizador é informado através de mensagem de alerta.

Na Figura 3.4, é apresentado o diagrama que representa os casos de uso agora descritos.

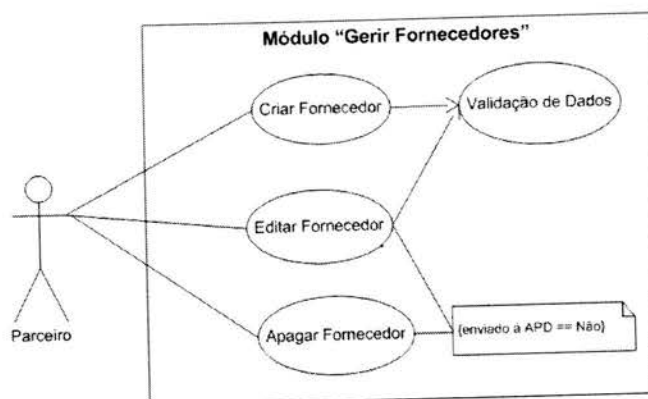


Figura 3.4: Diagrama de Casos de Utilização Disponíveis no Módulo Gerir Fornecedores

3.2.1.2 APD

O actor APD representa a equipa de gestão da Associação Porto Digital e, além de herdar os casos de utilização do actor Utilizador Geral, também pode aprovar ou rejeitar os de PEDIDOS DE REEMBOLSO e aprovar, associar ou rejeitar FORNECEDORES apresentados pelos Parceiros. Estes casos de utilização são acedidos através dos módulos Gerir Pedidos de Pagamentos e Gerir Fornecedores Porto Digital.

Independentemente da decisão da APD é automaticamente gerado um ALERTA ao Parceiro que é enviado quando este iniciar a aplicação, ou tentar sincronizar com a Associação Porto Digital. Este caso de utilização é transparente para a Associação Porto Digital.

Nas figuras Figura 3.5 e Figura 3.6 apresentam-se, respectivamente, os diagramas dos casos de utilização disponíveis no módulos Gerir Fornecedores Porto Digital e Gerir Pedidos de Pagamento.

3.2.1.3 POS_C

O actor POS_C representa a entidade financiadora do projecto Porto Digital, o Programa Operacional Sociedade do Conhecimento, tendo como casos de utilização próprios a aprovação ou rejeição de uma FOLHA DE PEDIDOS DE PAGAMENTO.

Em ambos os casos será gerado, de forma transparente para o POS_C, um ALERTA para a APD para que esta possa proceder às diligências necessárias.

3.2.1.4 Utilizador Geral

O actor Utilizador Geral representa o conjunto de funcionalidades que estão disponíveis para todos os actores descritos até ao momento: Parceiro, APD e POS_C. Os casos de

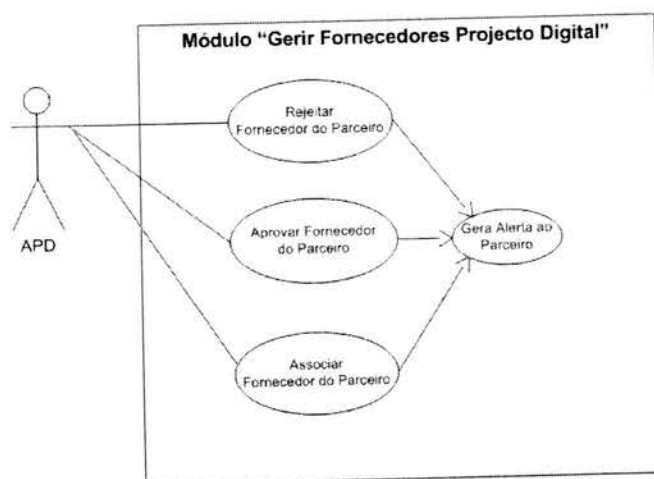


Figura 3.5: Diagrama de Casos de Utilização Disponíveis no Módulo Gerir Fornecedores Porto Digital

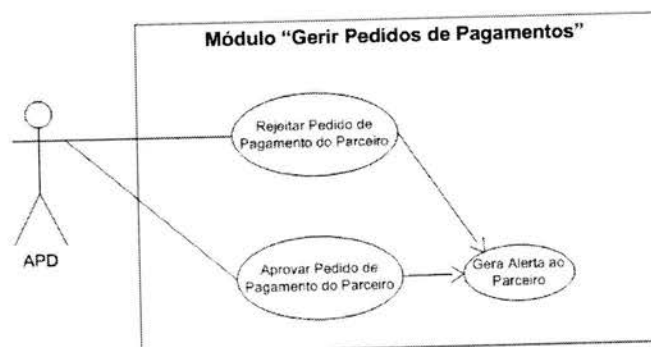


Figura 3.6: Diagrama de Casos de Utilização Disponíveis no Módulo Gerir Pedidos de Pagamento

utilização deste actor englobam a *Autenticação*, a *Mudança de Palavra-chave* e as funcionalidades agrupadas no módulo de *Consultas*.

Ao aceder a este módulo, é possível ao actor realizar consultas sobre as *RÚBRICAS*, *DOCUMENTOS*, *RELATÓRIOS* e *FORNECEDORES* dos sub-projectos. O tipo de *RELATÓRIOS* a que o Utilizador Geral pode aceder depende do tipo de utilizador que é: *Parceiro*, *APD* ou *POS_C*. O resultado da pesquisa sobre *FORNECEDORES* também dependerá do tipo de utilizador. Enquanto o *Parceiro* só visualizará os seus *FORNECEDORES*, já a *APD* e o *POS_C* irão ter acesso a todos os *FORNECEDORES* envolvidos no projecto. Apenas as consultas de *RÚBRICAS* e *DOCUMENTOS* não dependerão do actor que a efectua.

Na Figura 3.7 é apresentado o diagrama que captura os casos de utilização agrupados no módulo *Consultas*.

De salientar, que estas funcionalidades são herdadas pelos restantes três actores.

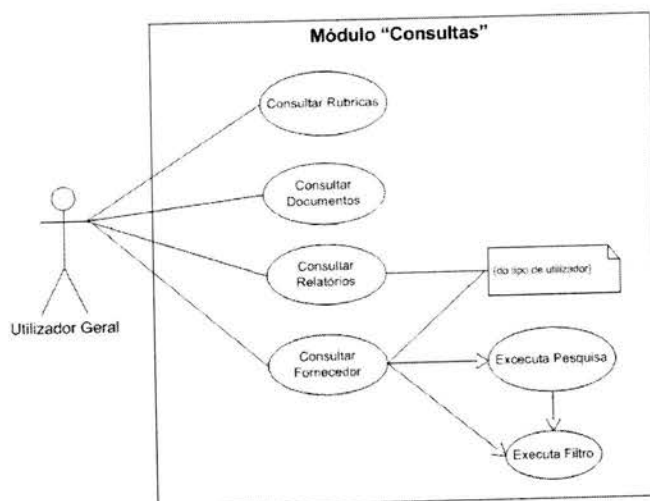


Figura 3.7: Diagrama de Casos de Utilização Disponíveis Através do Módulo Consultas

3.3 Requisitos Não Funcionais

Em muitas situações os requisitos não-funcionais podem ser mais críticos do que os requisitos funcionais, já que se os primeiros não são satisfeitos, os segundos ficam sem utilidade.

Nesta secção são descritos os requisitos não funcionais de fiabilidade, portabilidade, desempenho, usabilidade, segurança e escalabilidade que a aplicação deverá ter.

3.3.1 Fiabilidade

Dado a natureza crítica da informação com que a aplicação trabalhará, é crucial que os dados sejam consistentes. Por este motivo a aplicação deverá suportar o tratamento de excepções e suportar a recuperação de dados no caso de falhar.

3.3.2 Portabilidade

Como a aplicação será disponibilizado a todos os Parceiros do Porto Digital, é necessário garantir que a aplicação é independente do sistema operativo do cliente. Neste momento estão identificados dois tipos de sistemas operativos sobre os quais a aplicação terá de ser executada: *Windows* e *Linux*.

3.3.3 Desempenho

O desempenho do sistema deverá ser uma preocupação ao longo do desenvolvimento do mesmo. Os utilizadores não poderão estar períodos de tempo longos à espera que determinada acção execute. A aplicação deverá ter um tempo de acesso aos dados e execução de operações relativamente baixos, evitando assim que o utilizador perca demasiado tempo com determinada acção.

Outro ponto importante no desempenho, é relativo à sincronização de dados entre a aplicação que é executada em cada um dos Parceiro e a que será instalada na APD, e depois entre esta e o POSC. Apesar de não ser crítico, é necessário garantir que o desempenho nas comunicações seja razoável e que o utilizador esteja informado durante todo o processo.

É necessário ter em consideração que é expectável que a aplicação venha a ser executada em equipamentos bastante antigos e com baixa capacidade de processamento, máquinas típicas nos departamentos financeiros da maioria das instituições.

3.3.4 Usabilidade

É fundamental que todas as interfaces e a navegação entre elas sejam intuitivas para o utilizador. Deste modo poderemos melhorar a curva de aprendizagem de utilização do sistema, ou seja, com pouco esforço o utilizador deverá obter o máximo da aplicação.

É também importante que as operações, sobretudo de introdução de dados, sejam relativamente uniformes e claras de forma a evitar dificuldades ou erros sistemáticos. Se assim não for, estaremos a comprometer uma utilização natural e intuitiva por parte do utilizador.

3.3.5 Segurança

Como referido anteriormente, o sistema trabalhará com informação crítica. Por este motivo é vital garantir que estes dados não serão alterados indevidamente por um utilizador do sistema operativo onde é executada a aplicação, ou por uma terceira pessoa/entidade durante o processo de sincronização de dados.

3.3.6 Escalabilidade

Pretende-se que este sistema de processamento de PEDIDOS DE REEMBOLSO seja a base de um vasto sistema de suporte à gestão e decisão da Associação Porto Digital. A solução apresentada também poderá ser disponibilizada a outros projectos no âmbito do Portugal Digital, motivo pelo qual se torna necessário que esta seja escalável e de fácil manutenção.

3.4 Resumo

Este projecto tem por objectivo o desenvolvimento de uma aplicação que possa simplificar, agilizar e monitorizar todo o processo de PEDIDOS DE REEMBOLSO de cada Parceiro até à entidade financiadora do projecto, bem como o processo de aprovação desta e consequente comunicação à Associação Porto Digital e ao Parceiro. Para melhor compreensão do problema foi apresentado o Modelo de Objectos de Dominio que representa os principais objectos e interações entre eles.

De seguida foi definido o Diagrama de Casos de Utilização e, deste modo, as funcionalidades que a aplicação terá de disponibilizar a cada actor. Os casos de utilização foram apresentados através de diagramas *UML* e definem o modo como os utilizadores irão interagir com o sistema.

Com a definição dos requisitos não funcionais foram caracterizadas as condições de usabilidade, desempenho, segurança e fiabilidade que a aplicação deverá respeitar durante a execução. A aplicação também deverá de ser portátil e escalável.

Capítulo 4

Arquitectura do Aplicação a Desenvolver

Tendo por base o contexto do problema e os requisitos apresentados no capítulo precedente, é definida neste capítulo a arquitectura de um sistema capaz de responder de forma eficiente às necessidades definidas. Após uma visão geral, a arquitectura é descrita através da definição da arquitectura lógica e física.

4.1 Introdução

A não obrigatoriedade de conexão à Internet foi uma das características que se impôs. Deste modo quis-se evitar que os Parceiros, a Associação Porto Digital ou o POS_C tivessem de estar conectados à *Web* sempre que quisessem trabalhar com a aplicação.

Por este facto, cada um dos Parceiros, a Associação Porto Digital e a entidade financiadora terão um repositório de dados local. Deste modo poderão trabalhar com a aplicação sem necessidade de comunicações.

As comunicações entre as várias entidades intervenientes efectuem-se através de servidores, existindo dois servidores: um para gerir as comunicações entre os Parceiros e a Associação Porto Digital e outro para gerir as comunicações entre a Associação Porto Digital e o Programa Operacional Sociedade do Conhecimento.

Para a sincronização dos dados, cada Parceiro, a APD e o POS_C deverão ter um dispositivo de acesso à rede *Web*. Este dispositivo pode ser um modem *dial-up*, uma rede Intranet com acesso central à Internet ou qualquer outro mecanismo.

Cada PEDIDO DE REEMBOLSO tem um ou mais documentos que comprovam a realização da despesa. Para criar as imagens destes documentos, cada Parceiro e a Associação Porto Digital, terão de possuir um digitalizador. Neste momento não ficou definido a necessidade da entidade financiadora também ter um dispositivo de digitalização porque esta apenas aprova ou rejeita uma FOLHA DE PEDIDOS DE PAGAMENTO da APD. Caso se venha a definir que cada aprovação ou rejeição do POS_C deve ser acompanhada de um relatório explicativo, então também o POS_C deverá possuir um dispositivo de digitalização de documentos.

Na Figura 4.1 é ilustrada uma visão geral da arquitectura.

De salientar que na figura apresentada se admite a possibilidade de que a utilização desta aplicação possa ser alargada a outras cidades/regiões digitais.

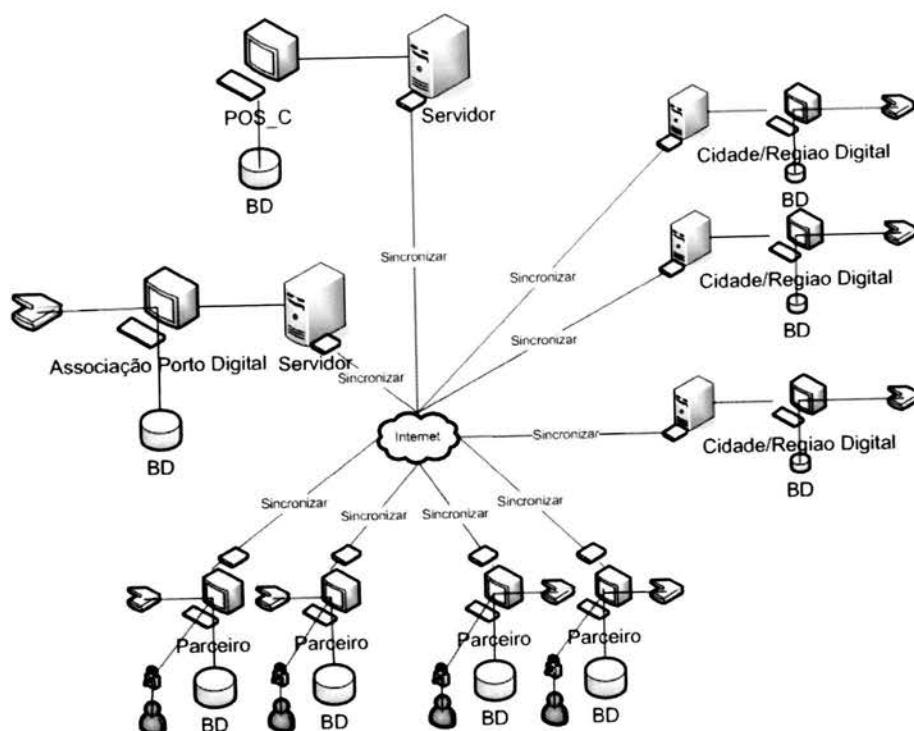


Figura 4.1: Modelo da Arquitectura

4.2 Arquitectura Lógica

Com a apresentação da arquitectura lógica do sistema, pretende-se explicar o modo como foram distribuídos e se relacionam os vários componentes do sistema.

A arquitectura adoptada é uma arquitectura de três camadas, em que cada uma delas apenas comunica com as camadas adjacentes. Este facto favorece a manutenção e escalabilidade do sistema. A Figura 4.2 ilustra o modelo lógico a implementar.

A camada **Repositório de Dados** é responsável pela manutenção física de todos os dados referentes ao sistema. A apresentação dos dados é da responsabilidade da camada de **Interface** que obtém estes dados através da camada **Lógica de Negócios**. Esta camada tem por função processar os dados que recebe da camada de **Repositório de Dados**.

4.3 Arquitectura Física

O problema em questão não obrigava, à partida, à utilização de um repositório de dados em formato *XML*. A implementação de uma base de dados suportada por um motor *SQL* satisfazia plenamente o objectivo de tornar persistentes os dados da aplicação. Contudo, a utilização de base de dados em cada Parceiro provocaria maiores problemas de operacionalização e manutenção da aplicação. Com esta solução, seria também necessário a implementação de um interface para que os dados armazenados na base de dados fossem convertidos para, e do formato de transmissão escolhido. Assim, o *XML* permite eliminar a necessidade de interfaces de transformação de dados já que pode ser utilizado quer para tornar persistentes os dados da aplicação quer para transmissão dos mesmos.

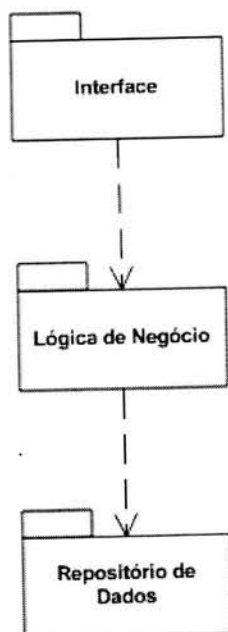


Figura 4.2: Modelo de Camadas Lógicas

No que respeita às soluções de armazenamento possíveis recorrendo à tecnologia *XML*, optou-se pelo armazenamento dos documentos *XML* no sistema de ficheiros do sistema operativo. Esta opção, em detrimento do armazenamento dos documentos numa base de dados relacional, deveu-se, em grande medida, à complexidade na conversão dos dados em *XML* para uma base de dados relacional. Já os custos financeiros de uma base de dados orientada a objectos colocou de parte esta possibilidade.

A necessidade de aceder, durante a execução da aplicação, a todos os dados dos vários documentos, fez com que a utilização do *DOM* fosse a opção natural em detrimento do *SAX*. Além disso, o *DOM* revela-se ideal para armazenar informação quando o modelo de dados pretendido é já uma árvore. Deste modo, também foram evitados certos riscos inerentes à maior complexidade de implementação do *SAX*. Dado o vasto conjunto de tecnologias contactadas pela primeira vez neste estágio, a facilidade e simplicidade do *DOM* foram observados na tomada de decisão.

Serão assim criados quatro ficheiros *XML* que irão conter a informação relativa a FORNECEDORES, PEDIDOS DE REEMBOLSO, RÚBRICAS e DOCUMENTOS. Para a gestão dos ALERTAS recebidos da APD serão utilizados dois documentos *XML*. De suporte ao mecanismo de persistência de dados em caso de falha global da aplicação, serão criados ainda mais dois ficheiros: um para manter as alterações não gravadas sobre os FORNECEDORES e outro para manter as alterações não gravadas sobre os PEDIDOS DE REEMBOLSO. Quanto às RÚBRICAS e aos DOCUMENTOS não será possível ao utilizador alterá-los, pelo que não são necessários ficheiros para manter alterações. A informação contida nos ficheiros que guardam os dados das RÚBRICAS e DOCUMENTOS podia ter sido introduzida directamente na aplicação, já que deverá manter-se sem alterações durante todo o projecto. No entanto, com a informação sobre RÚBRICAS e DOCUMENTOS contida no código da aplicação, não seria possível à equipa de gestão alterar a lista de RÚBRICAS e DOCUMENTOS. Assim, se vier a ser necessário alterar a lista de RÚBRICAS e DOCUMENTOS apenas será necessário trocar os ficheiros *XML*.

Na camada **Repositório de Dados** ainda serão encontrados mais dois ficheiros. Um para armazenamento de dados de configuração e outro para armazenamento de dados necessários à autenticação dos utilizadores. A gestão de consultas e acessos a estes ficheiros será mantida através de uma biblioteca da qual dependerão os componentes da camada **Lógica de Negócio**. De referir, que dado o carácter legível de um documento *XML*, será necessário implementar um mecanismo que impeça o acesso directo à informação contida nestes documentos. Este mecanismo será incorporado no componente **Persistência de Dados**.

Na camada **Lógica de Negócio** estarão implementadas quer as funções de processamento dos dados, desde pesquisas e filtros às operações de introdução, alteração ou eliminação de dados, através da biblioteca **Funções**, quer os mecanismos necessários à comunicação e sincronização de dados entre os vários actores, através da biblioteca **Comunicação**. Neste componente **Funções** também estará implementada toda a lógica necessária à verificação e validação, quer dos dados referentes aos **FORNECEDORES**, quer os dados referentes aos **PEDIDOS DE REEMBOLSO**. Ambos os componentes são chamadas através do cerne da aplicação — **run** — que reporta os resultados à interface gráfica, o componente **Interface Gráfica**.

No diagrama de componentes, apresentado na Figura 4.3 pode verificar-se, de forma intuitiva, as três camadas lógicas enumeradas na secção anterior, bem como os componentes agora descritos.

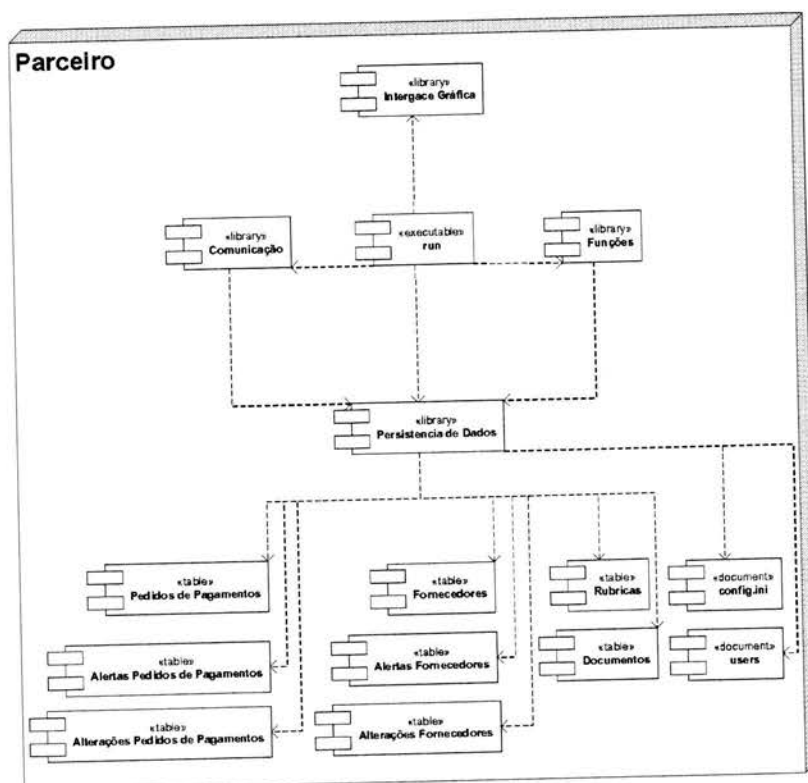


Figura 4.3: Diagrama de Componentes

Já em relação à distribuição dos diversos componentes, a Figura 4.4 ilustra a arquitectura física do sistema ao nível da topologia de *hardware* sobre a qual são executados os diversos componentes do sistema.

Assim, em cada Parceiro será instalada a aplicação que será utilizada para formalizar

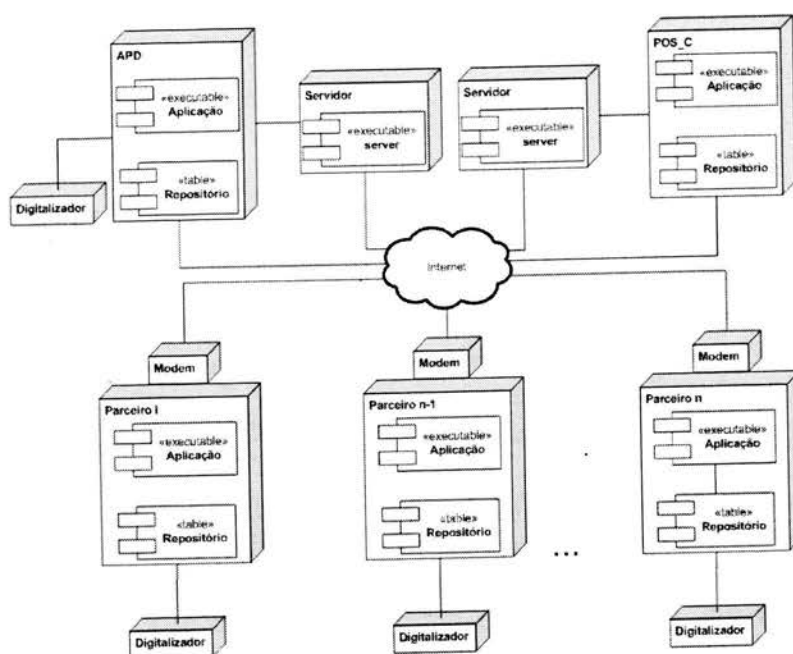


Figura 4.4: Diagrama de Distribuição

os PEDIDOS DE REEMBOLSO à Associação Porto Digital. É necessário que cada Parceiro possua um digitalizador para que possa obter uma imagem digital dos documentos que façam prova das despesas apresentadas. Através de uma ligação de acesso telefónico, da Intranet a que a máquina possa estar ligada ou através de outro sistema, é obrigatório que os Parceiros estejam ligados à Internet. Esta conexão não tem de ser contínua, sendo apenas necessária aquando da sincronização.

Na Associação Porto Digital também será instalada a aplicação fornecida aos Parceiros mas, neste caso, com pequenas alterações nas funcionalidades disponibilizadas. Estas diferenças de funcionalidades serão mais substanciais à medida que novas funcionalidades específicas deste actor venham a ser definidas. Na Associação Porto Digital será necessário um digitalizador para a criação das imagens dos documentos das próprias despesas e um meio de ligação ao servidor que recebe os PEDIDOS DE REEMBOLSO. Este acesso ao servidor permitirá à aplicação na Associação Porto Digital a consulta e tratamento de novos PEDIDOS DE REEMBOLSO, quer a criação de ALERTAS para os diversos Parceiros. É também através deste servidor que a Associação Porto Digital enviará as FOLHAS DE PEDIDOS DE PAGAMENTO ao POS_C e que este retornará a sua avaliação.

O POS_C, ao contrário dos Parceiros e da APD, não tem qualquer necessidade de digitalizador já que não se prevê que venha a necessitar de anexar documentos a ALERTAS que envie.

Tanto o servidor na APD como no POS_C, não terão qualquer lógica de validação de dados. A verificação da autenticidade do emissor e integridade dos dados será garantida pelas respectivas aplicações. O servidor na APD apenas receberá os PEDIDOS DE REEMBOLSO por parte dos Parceiros e enviará aos mesmos, os ALERTAS gerados pela APD que lhes digam respeito. De forma similar, o servidor na entidade financiadora receberá as FOLHAS DE PEDIDOS DE PAGAMENTO provenientes da APD e enviará os relatórios de avaliação dos pedidos contidos nas FOLHAS DE PEDIDOS DE PAGAMENTO. Ambos os servidores, como é natural, terão de estar disponíveis via Internet de forma contínua para

que a transmissão de dados possa ocorrer a qualquer momento.

4.4 Resumo

A arquitectura lógica da aplicação é apresentada segundo três camadas: **Repositório de Dados**, **Lógica de Negócio** e **Interface**. Na camada de **Repositório de Dados** optou-se pelo armazenamento de documentos *XML* no sistema de ficheiros do sistema operativo e a utilização da *API DOM* para a leitura dos dados. Esta camada possui a lógica necessária para verificação dos dados. Na camada **Lógica de Negócio** estarão implementadas as funções de processamento e comunicação dos dados. A camada de **Interface**, que será desenvolvida quase na sua totalidade através do *QtDesigner*, será o meio de interação do utilizador com a aplicação.

Esta aplicação estará instalada em cada um dos Parceiros e a máquina onde será executada terá de dispor de um dispositivo de digitalização de documentos e um meio de acesso à Internet. O dispositivo de digitalização é necessário para introdução no sistema de ficheiros do sistema operativo dos documentos comprovativos da despesa. O acesso à Internet é obrigatório para se realizar a sincronização de dados.

Esta aplicação, a correr nos Parceiros irá comunicar com um servidor que controla a comunicação entre os Parceiros e a Associação Porto Digital. Este servidor recebe os **PEDIDOS DE PAGAMENTOS** e os novos **FORNECEDORES** de cada Parceiro e envia ao Parceiro os **ALERTAS** que recebeu da Associação Porto Digital. A aplicação que será executada na Associação Porto Digital envia para este servidor os **ALERTAS** que pretenda que este envie para os Parceiros e recebe os dados que o servidor recebeu dos Parceiros. A aplicação da APD também envia para este servidor as **FOLHAS DE PEDIDOS DE PAGAMENTOS** a serem enviadas para a entidade financiadora e também recebe dele a respectiva decisão do **POS.C**. Para a comunicação com a Associação Porto Digital, o **POS.C** terá um servidor que receberá as **FOLHAS DE PEDIDOS DE PAGAMENTOS** e enviará o **ALERTA** de aprovação ou rejeição de uma determinada **FOLHA DE PEDIDOS DE PAGAMENTO**.

A aplicação a ser instalada na Associação Porto Digital é um super-conjunto da aplicação a ser instalada em cada Parceiro e a aplicação a ser instalada no Programa Operacional Sociedade do Conhecimento é um sub-conjunto da aplicação a ser instalada em cada Parceiro.

Capítulo 5

Implementação

Este capítulo descreve os principais detalhes da implementação do sistema definido anteriormente. É também dado a conhecer o modo como as tecnologias foram utilizadas em cada parte da aplicação.

5.1 Introdução

A implementação partiu da arquitectura definida tendo sido utilizadas como tecnologias fundamentais o *XML*, o *C/C++* e o *Qt*. No entanto, foi necessário tomar algumas decisões no decorrer deste processo.

Ao ser utilizado o *XML* como sistema de base de dados e formato de comunicação de dados, nesta fase foi necessário definir a estrutura e o modo de verificação da estrutura dos vários documentos *XML*.

A linguagem *C/C++* foi utilizada através da biblioteca *openssl/tls* para implementação do mecanismos de autenticação e para a encriptação local dos documentos *XML*. Quer para o mecanismo de autenticação quer para a encriptação local dos documentos *XML* foram analisadas várias possibilidades.

O *Qt* foi a linguagem base de todo o desenvolvimento; utilizada para a validação da estrutura dos documentos *XML*, verificação dos dados introduzidos pelo utilizador, pesquisas sobre as várias estruturas de dados até à sincronização e interface gráfica.

Nas seguintes secções serão analisadas de forma pormenorizada a implementação da camada de **Repositório de Dados**, dos principais mecanismos e funcionalidades e a **Interface Gráfica**.

5.2 Repositórios de Dados

Tendo-se elegido neste projecto o *XML* como formato de armazenamento e transmissão de dados, nesta fase foi necessário definir a estrutura e o modo de verificação da estrutura dos vários documentos *XML*.

Na secção 5.2.1 é apresentada a estrutura dos objectos *DOM* com que a aplicação está preparada para interagir. Para garantir que a aplicação apenas trabalha com documentos *XML* com a estrutura pretendia, depois da aplicação ler os documentos *XML* para os objectos *DOM*, realiza pesquisas sobre os *DOM* retornando mensagens de erro caso a estrutura do *DOM* não coincida com a que deve receber. A descrição desta verificação é feita na secção 5.2.2.

Dado o carácter legível do *XML* foi implementado um mecanismo de cifração/decifração através do algoritmo de encriptação *Blowfish*. A descrição deste mecanismo, assim como de outras soluções analisadas é apresentado na secção 5.2.3.

5.2.1 Estrutura DOM

A dúvida entre a informação que seria atributo ou elemento subsistiu durante algum tempo. Este problema não tem, como boa parte dos problemas de engenharia, uma solução única [Ramalho e Henriques 2002]. Tentando procurar a melhor solução de compromisso foi seguido como regra geral que os dados relativos a um FORNECEDOR, PEDIDO DE REEMBOLSO, DOCUMENTO ou RÚBRICA seriam guardados como atributos; atributos de um elemento que representa o tipo de dado a que se refere. Deste modo pretendeu-se diminuir o tempo de pesquisa sobre a estrutura *DOM* já que se obtêm os dados em níveis de pesquisa superiores. Ainda que o ganho de eficiência possa não ser considerável numa única pesquisa, como existe em muitas situações a necessidade de realizar pesquisas sobre os *DOM*, no global, é possível obter ganhos consideráveis porque o ganho em cada pesquisa é cumulativo com as precedentes. Assim um elemento *folha* é do tipo:

35a $\langle$ Exemplo: Elemento Folha Genérico dos Documentos XML 35a $\rangle \equiv$
`<"tipoDeDado" "listaDeAtributos" />`

Onde `tipoDeDado` é:

- `rubrica` no *XML* que guarda os dados sobre as RÚBRICAS permitidas no projecto;
- `documento` no *XML* que guarda os dados sobre os DOCUMENTOS permitidos no projecto;
- `fornecedor` no *XML* que guarda os dados sobre os FORNECEDORES;
- `pedido` no *XML* que guarda a informação sobre os PEDIDOS DE REEMBOLSO;
- `alteracoes` no *XML* que guarda a informação das operações sobre os FORNECEDORES não gravadas ou descartadas antes do encerramento da aplicação;
- `alteracoesPP` no *XML* que guarda a informação das operações sobre os PEDIDOS DE REEMBOLSO não gravadas ou descartadas antes do encerramento da aplicação;
- `alertas` no *XML* que guarda a informação sobre os alertas relativos aos FORNECEDORES;
- `alertaPedidosDePagametros` no *XML* que guarda a informação sobre os alertas relativos aos PEDIDOS DE REEMBOLSO;
- `utilizador` no *XML* que guarda a informação sobre os utilizadores autorizados a aceder à aplicação.

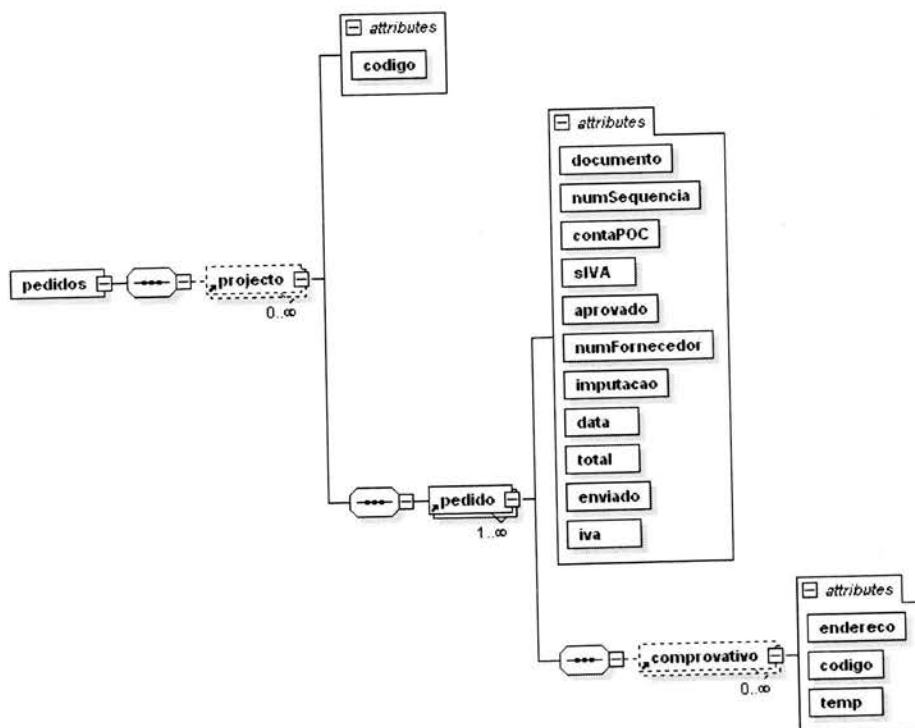
Por exemplo, o elemento representativo de uma RÚBRICA permitida no projecto é deste tipo:

35b $\langle$ Exemplo: Elemento Rúbrica 35b $\rangle \equiv$
`<rubrica despesas="De capital" categoria="Imobilizações Corpóreas"
descricao="Hardware" contaPOC="423"
tipo_rubrica="Equipamento Básico"/>`

ou mais documentos de quitação. Para guardar a informação relativa aos documentos de quitação de cada despesa, cada PEDIDO DE REEMBOLSO terá tantos filhos quanto os documentos necessários para provar a existência e pagamento da despesa. Este elemento tem `<comprovativo>` como etiqueta. Assim, o seguinte exemplo de XML pode ser um XML de PEDIDOS DE REEMBOLSO em que estão representados dois pedidos. Um referente ao projecto de código "INF01" e que tem dois documentos para suportar o pedido, e outro referente ao projecto "ACE05", com apenas um documento de apoio:

37 *Exemplo: Ficheiro XML de Pedidos de Reembolso 37* ≡

```
<pedidos>
  <projecto codigo="INF01" >
    <pedido "lista de atributos" >
      <comprovativo "lista de atributos" />
      <comprovativo "lista de atributos" />
    </pedido>
  </projecto>
  <projecto codigo="ACE05" >
    <pedido "lista de atributos">
      <comprovativo "lista de atributos" />
    </pedido>
  </projecto>
</pedidos>
```



Generated with XMLSpy Schema Editor www.altova.com

Figura 5.2: Diagrama da Estrutura de Documentos XML Válidos de PEDIDOS DE REEMBOLSO

Na Figura 5.2 é apresentada a ilustração da estrutura do documentos XML de PEDIDOS DE REEMBOLSO gerada através da definição do *XSD Schema* disponível em anexo.

Existe ainda um *DOM* que apenas é mantido em memória. Este *DOM* contém os dados que são enviados pela aplicação para o servidor e contém a informação sobre FORNECEDORES e PEDIDOS DE REEMBOLSO que tem de ser aprovada pela Associação Porto Digital.

O nó raiz deste *DOM* é `<sincronizacao>` e tem sempre dois nós filhos. O primeiro nó filho é o elemento `<fornecedores>` e este tem como filhos os elementos `<fornecedor>` do *DOM* de FORNECEDORES com o atributo *enviado* igual a "Não". O segundo nó filho de `<sincronizacao>` é o elemento `<pedidos>` e tem como filhos os elementos `<projecto codigo="codigoDoProjecto">`. Este tem como filhos os PEDIDOS DE REEMBOLSO desse projecto que tenham o atributo *enviado* igual a "Não".

38 *<Exemplo: DOM enviado à APD aquando da Sincronização 38>*≡

```
<sincronizacao>
  <fornecedores>
    <fornecedor validado="Não" codigo="6" nome="Telefonica"
      numContribuinte="112333113" enviado="Sim"
      mercado="Intracomunitário" />
    <fornecedor validado="Não" codigo="8" nome="EDP"
      numContribuinte="123123123" enviado="Sim"
      mercado="Nacional"/>
    <fornecedor validado="Não" codigo="11" nome="Clix"
      numContribuinte="123456789" enviado="Não"
      mercado="Nacional"/>
  </fornecedores>
  <pedidos>
    <projecto codigo="SEC10">
      <pedido documento="Cheques" numSequencia="159" contaPOC="Pessoal (dos
        parceiros)" sIVA="100" aprovado="Não" numFornecedor="PT"
        imputacao="5" data="2005.07.14" total="121" enviado="Não"
        iva="21">
        <comprovativo endereco="/home/" codigo="cheque1.pdf" temp="abrir
          explorador" />
        <comprovativo endereco="/home/" codigo="cheque2.pdf" temp="abrir
          explorador" />
      </pedido>
    </projecto>
    <projecto codigo="ACE05">
      <pedido documento="Facturas" numSequencia="158"
        contaPOC="Comunicações" sIVA="100" aprovado="Não"
        numFornecedor="ad4" imputacao="5" data="2005.07.14"
        total="121" enviado="Sim" iva="21" />
      <comprovativo endereco="/home/" codigo="extractoDeConta.pdf"
        temp="abrir explorador" />
    </pedido>
  </projecto>
</pedidos>
</sincronizacao>
```

5.2.2 Verificação da Estrutura do DOM

A verificação da estrutura dos *DOMs* que mantêm os dados da aplicação é realizada pelo componente *Persistência de Dados*. Neste componente, após ter sido feita a leitura, através da *API DOM* do *Qt*, são realizadas diversas pesquisas sobre a estrutura de dados para determinar a correcção da árvore representativa do documento *XML*.

Quando a estrutura *DOM* não está correcta a função *dtdchecker* que implementa a verificação retorna *FALSE* para a função *openFile*. Esta recebendo *FALSE* da *dtdchecker* retorna também *FALSE* para a função que a invocou. A função que pediu a abertura do ficheiro, recebendo como retorno o valor *FALSE* avisa o utilizador que a estrutura do ficheiro não está correcta e encerra a aplicação.

De seguida é apresentado o código de verificação da boa formação do *DOM* de ALERTAS de FORNECEDORES.

39 *(Exemplo: Código de Validação da Estrutura do DOM Alerta de Fornecedores 39)≡*

```
{
    //testa elemento raiz
    if ( root.tagName() != "alertasFornecedores" )
        return FALSE;
    QDomNode n = root.firstChild();
    QDomNodeList nList = root.childNodes();
    for ( int i = 0; i < nList.length(); i++ )
    {
        QDomElement e = n.toElement();
        //testa elemento "operacao"
        if ( ! ( e.tagName() == "editar" ||
                e.tagName() == "adicionar" ||
                e.tagName() == "apagar" ) )
            return FALSE;
        QDomNode n2 = e.firstChild();
        QDomElement e2 = n2.toElement();
        //testa elemento "tipoDeDado"
        if ( e2.tagName() != "fornecedor" )
            return FALSE;
        n = n.nextSibling();
    }
    break;
}
```

De referir, que esta pesquisa não verifica os atributos dos elementos. Caso a estrutura do *DOM* em termos de elementos estiver correcta, mas houver uma falha na lista de atributos, a aplicação irá prosseguir sem gerar nenhum alerta. Depois, quando esses dados forem apresentados, as células da tabela correspondentes a atributos corrompidos ficarão vazias.

5.2.3 Protecção dos Dados

Dado o carácter legível do *XML* e à possibilidade de qualquer utilizador ter acesso de leitura e escrita ao ficheiro, tornou-se obrigatório implementar um mecanismo de protecção dos dados. O mecanismo escolhido foi a codificação dos dados através de um algoritmo de chave secreta.

A utilização de um algoritmo de chave secreta deveu-se à eficiência deste tipo de algoritmo e ao facto de não existir a necessidade de transmitir estes dados para outras entidades, portanto, não se colocarem problemas de acordo, transmissão e renovação de chaves.

Dentro dos algoritmos de chave secreta o leque de possibilidades é vasto, tendo sido analisada a possibilidade de utilizar os algoritmos:

- *DES - Data Encryption Standard*;
- *Triple DES ou 3DES*;
- *AES - Advanced Encryption Standard*;
- *RC4 - Ron's Cipher 4*;
- *Blowfish*.

O *DES* foi um dos algoritmos de chave secreta mais utilizados [Schneier 2004]. Contudo, já nos meados da década de noventa se sabia que as chaves de encriptação do *DES*, de apenas 56 bits, eram demasiado pequenas para a época. Em parte, a confiança depositada no *DES* advinha de este ter passado os testes da NSA¹ e ter sido publicado como uma norma NSA [Schneier 1995]. Além disso, o *DES* já tinha sobrevivido a 20 anos de intensa cripto-análise. No entanto, em 1990 Biham e Shamir aplicaram uma técnica de análise designada por cripto-análise diferencial [Biham e Shamir 1991] e [Biham e Shamir 1993].

Esta técnica apenas necessita de gerar 2^{47} pares de “texto aberto” e “texto cifrado”, consideravelmente menos possibilidades do que nas abordagens anteriores [Bishop 2002]. Em 1998, foi publicado o desenho de um computador e *software* que conseguiam quebrar qualquer mensagem encriptada pelo *DES* em apenas alguns dias [Gilmore 1998]. Assim, e ainda que disponível na biblioteca *openssl/tls*, o algoritmo *DES* foi desvalorizado como possível solução.

No final de 2001, o NIST² apresentou o algoritmo *AES* de Rijndael como o sucessor do *DES*. Ao contrario do *DES*, o *AES* permite a utilização de chaves de 80, 112, 128, 192 e 256 bits e opera com blocos de 128 bits. O *AES* é imune aos ataques que romperam com a confiança no *DES* [de Jesús Angel Angel 2005]. Na última conferência sobre o *AES* o NIST voltou a confirmar a segurança deste algoritmo até 2010 para chaves de 80 bits, 2020 para chaves de 112 bits e chaves de 128 e datas posteriores para mais bits [Kelsey 2004]. Ainda que o *AES* seja um algoritmo seguro, como a biblioteca *openssl/tls* não disponibiliza uma implementação deste algoritmo, a sua possível utilização ficou relegada para segundo plano.

Outra solução apresentada, mais conservadora, foi o *Triple DES*. O *Triple DES* é um algoritmo de cifrado múltiplo, ou seja, é aplicado várias vezes o mesmo algoritmo para fortalecer a chave de encriptação. Neste algoritmo é aplicado três vezes o algoritmo *DES* da seguinte forma: a primeira vez usa-se uma chave K1 com um bloco (B0) obtendo um bloco encriptado, B1. Na segunda passagem parte-se do bloco B1 e volta-se a aplicar o *DES* com a chave K2, inversa da K1, resultando o bloco B2. Finalmente, aplica-se uma chave K3, diferente de K1 e K2, ao bloco B2 obtendo o valor final da encriptação B3. Através deste mecanismo, o *Triple DES* usa chaves de 168 bits aumentando os recursos necessários para ser quebrado. O *Triple-DES* não está desenhado para ser um algoritmo

¹National Security Agency

²National Institute of Software and Technology

de encriptação por blocos óptimo já que é vulnerável a ataques baseados na relação entre chaves (*related-key attacks*) [Kelsey et al. 1999].

O *RC4* é o algoritmo de encriptação de fluxo mais usado. Por exemplo, os protocolos de envio de dados como o *Secure Socket Layers* e o *Wired Equivalent Privacy* utilizam este algoritmo. As transformações neste algoritmo são lineares. De uma forma geral, o algoritmo consiste em utilizar um vector que a cada utilização tem os seus valores permutados e misturados com a chave. Este facto faz com que o algoritmo seja muito dependente da palavra-chave. A chave *RC4* pode ter até 256 bytes (2048 bits) mas à medida que o tamanho da chave aumenta, a eficiência do algoritmo diminui. Contudo este algoritmo não é considerado um bom sistema criptográfico. Em 2001, os criptógrafos Scott Fluhrer, Itsik Mantin e Adi Shamir publicaram um artigo [Fluhrer et al. 2001] descrevendo uma vulnerabilidade do *RC4*. Segundo este artigo pode-se recuperar a chave utilizada se a inicialização do algoritmo cumprir determinadas premissas, muito comuns, e se se interceptarem um número suficiente de mensagens. O artigo descreve dois ataques. O primeiro baseia-se em padrões invariantes. Ou seja, existem padrões que, se existirem na chave, se propagam ao estado interno do algoritmo, debilitando-o consideravelmente. O segundo ataque descreve a recuperação da chave secreta, quando a chave do algoritmo deriva da concatenação da chave secreta e de um vector inicial público e conhecido. A primeira vulnerabilidade faz com que os primeiros bytes gerados pelo *RC4* resultem previsíveis. A segunda vulnerabilidade permite recuperar a parte secreta da chave *RC4* recompilando um número de mensagens e vectores iniciais. Estas vulnerabilidades são graves e minam a confiança no *RC4* como na fiabilidade dos produtos que a utilizam; por exemplo, o protocolo *WEP*. Ainda que disponível na biblioteca *openssl/tls*, as vulnerabilidades deste algoritmo colocam a sua utilização de parte.

O *Blowfish* é um algoritmo de encriptação em blocos de dados de 8 bytes cada. O algoritmo consiste em duas partes distintas: a expansão da chave e a encriptação dos dados. Na fase da expansão da chave, a chave de até 56 bytes, é convertida em várias sub-chaves totalizando 4168 bytes. A fase de encriptação da mensagem engloba 16 voltas. Cada volta consiste numa permutação e substituição em função da chave secreta expandida. Todas as operações são operações do tipo *XOR* e adições de *palavras* de 32 bits. As únicas operações adicionais são quatro *lookups* dos vectores indexados. Não está patenteado, e o seu código é de domínio público, estando disponível a sua implementação na biblioteca *openssl/tls*. É bastante rápido, e não se conhecem cripto-análises satisfatórias sobre este algoritmo. John Kelsey, Serge Vaudenay e Vincent Rijmen's desenvolveram em paralelo cripto-análise ao algoritmo *Blowfish*. John Kelsey desenvolveu um ataque que consegue quebrar três das operações do *Blowfish*. Contudo não foi capaz de estendê-lo às restantes operações [Schneier 1995]. Serge Vaudenay examinou uma variante simplificada do *Blowfish*. Este ataque supunha que fosse conhecida a *S-Box* e a chave fosse fraca. Mesmo neste caso, este ataque não surtia efeito no *Blowfish* de 16 voltas. Não conhecendo a *S-Box*, o ataque de Serge Vaudenay só conseguia determinar se a chave utilizada na variante simplificada do *Blowfish* era fraca ou não, mas nunca conseguia obtê-la [Schneier 1995]. Vincent Rijmen's desenvolveu, na sua tese de doutoramento, um ataque diferencial de segunda ordem que conseguiu romper quatro operações do *Blowfish*. Tal como Serge Vaudenay, não conseguiu estender este ataque às restantes operações [Schneier 2005].

Os níveis de segurança que proporciona e o facto da sua implementação estar disponível através da biblioteca *openssl/tls* foram os critérios que levaram à utilização do algoritmo *Blowfish* em detrimento dos restantes.

No que respeita à definição da chave de encriptação **key** e vector de inicialização **iv**

foram usados os valores de 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 e 1,2,3,4,5,6,7,8, respectivamente.

A chave de encriptação encontra-se definida no próprio código. Desta forma, após compilada a aplicação, não existe possibilidade de consultar a chave utilizada.

5.3 Mecanismos Importantes

Nesta secção é descrito o mecanismo para controlar o acesso à aplicação, o mecanismo para verificar a correcção e coerência dos dados introduzidos pelo utilizador e o mecanismo que garante a recuperação do estado da aplicação em caso de falha geral.

5.3.1 Autenticação

Este é o primeiro mecanismo de segurança da aplicação ao limitar o acesso à aplicação apenas a utilizadores autorizados. Ou seja, é necessário que o utilizador introduza o seu *Nome* e *Palavra Chave*.

Para tornar persistentes os dados dos utilizadores existe um documento *XML* que mantém a lista de utilizadores e respectivas palavras-chaves. Depois de o utilizador introduzir os dados e iniciar o processo de autenticação pressionando o botão *Ok*, é aplicada à *Palavra-Chave* a função *hash Message Digest Algorithm 5 (MD5)*. De seguida, é lido para memória o documento *XML* que guarda os dados dos utilizadores gerando um *DOM* da informação contida no ficheiro. Este *DOM* é percorrido e caso seja encontrado um elemento `<utilizador>` com o atributo *nome* igual ao introduzido na interface gráfica e a *password* igual à chave *MD5* gerada, então é permitido o acesso à aplicação. Caso não encontre um utilizador com os dados introduzidos, é apresentada ao utilizador uma mensagem de erro.

Na geração da chave *MD5* da *palavra-chave* introduzida pelo utilizador recorreu-se à biblioteca *openssl/evp.h*. Esta biblioteca de *C* disponibiliza as estruturas de dados e funções usadas para a implementação do algoritmo *MD5* [Project 2005].

Em termos de estruturas de dados disponíveis na biblioteca *openssl/tls* foram utilizadas as variáveis **EVP_MD_CTX mdctx** e **const EVP_MD *md**. Já em relação a funções utilizaram-se as seguintes:

- **OpenSSL_add_all_digests()** — função para inicialização da tabela que contém informação sobre os algoritmos de funções *hash* e *assinaturas digitais* disponibilizados pela biblioteca *openssl/tls*.
- **EVP_get_digestbyname("md5")** — função que retorna uma estrutura **EVP_MD** quando lhe é passada o nome de uma função *hash* disponibilizada pela biblioteca *openssl/tls*. É necessário que já tenha sido chamada a função **OpenSSL_add_all_digest()**.
- **EVP_MD_CTX_init(&mdctx)** — inicializa a variável de contexto **mdctx**.
- **EVP_DigestInit_ex(&mdctx, md, NULL)** — especifica que a variável **mdctx** será utilizada com o algoritmo especificado **md** e como o terceiro argumento é **NULL**, então é porque se utilizará a implementação deste algoritmo disponibilizada na biblioteca *openssl/evp.h*.

- `EVP_DigestUpdate( ctx, d, cnt )` — realiza o *hash* de `cnt` bytes de `d` para a variável `ctx`. Esta função pode ser chamada diversas vezes durante o mesmo processo de encriptação.
- `EVP_DigestFinal_ex( ctx, md, md_len)` — recupera o valor `ctx` e coloca-o na variável `md`. O número de bytes escritos para a variável `md` é colocado em `md_len`, até ao máximo de `EVP_MAX_MD_SIZE` bytes que serão escritos. Depois da chamada à função `EVP_DigestFinal_ex()` não se poderão efectuar mais chamadas à função `EVP_DigestUpdate()` mas `EVP_DigestInit_es()` pode ser chamado para inicializar uma nova operação de *hash*.
- `EVP_MD_CTX_cleanup(&mdctx)` — limpa a variável `ctx`. Esta função pode ser chamada várias vezes durante o mesmo processo de encriptação.

Em seguida apresenta-se a parte do código implementado para geração das chaves MD5.

43 *(Exemplo: Código de Geração de Chaves MD5 43)≡*

```
//LE VALORES DA INTERFACE
QString userName = userNameLineEdit->text();
QString password = passwordLineEdit->text();
//chave MD5 da palavra-chave introduzida
QString passwordMD5 = "";

//
char mess1[password.length()];

//transforma QString para char array C/C++
for ( int i = 0; i < password.length(); i++ )
    mess1[i] = (password.at( i )).latin1();

EVP_MD_CTX mdctx;
const EVP_MD *md;
unsigned char md_value[EVP_MAX_MD_SIZE];
unsigned int md_len;
int i;

OpenSSL_add_all_digests();

md = EVP_get_digestbyname("md5");

CHECK_PTR( md );

//GERA CHAVE MD5
EVP_MD_CTX_init(&mdctx);
EVP_DigestInit_ex(&mdctx, md, NULL);
EVP_DigestUpdate(&mdctx, mess1, password.length() );
EVP_DigestFinal_ex(&mdctx, md_value, &md_len);
EVP_MD_CTX_cleanup(&mdctx);

//transforma char array de C/C++ em QString
for(i = 0; i < md_len; i++)
```

```

{
    printf ("%02x", md_value[i] );
    int cod = (int)md_value[i] ;
    passwordMD5.operator+=( QString::number (cod) );
}

```

É também dado ao utilizador a possibilidade de alterar a *palavra chave*. Para tal, o utilizador apenas necessita de preencher os campos da janela apresentada na Figura 5.3.

Figura 5.3: Formulário para Mudar *Palavra-chave*

Antes de iniciar a pesquisa sobre o *DOM* de utilizadores autorizados, a aplicação verifica se a nova *Palavra Chave* e a confirmação da *Palavra Chave* são iguais. Se forem iguais é gerada a respectiva chave *MD5* e iniciada a pesquisa. Caso exista um utilizador que verifique os dados introduzidos é efectuada a alteração da *Palavra Chave*. Não sendo possível fazer a alteração, o utilizador é informado do erro que ocorreu. O utilizador também é informado se a palavra-chave tiver sido alterada com sucesso.

5.3.1.1 Algoritmo MDA

Proposto em 1991 por Ron Rivest, o *MD5* é uma versão melhorada da função *hash MD4*. Ainda que mais complexo, o desenho e arquitectura do *MD5* é similar à do *MD4* e, como este, gera *hash's* de 128 bits [Schneier 1994]. O *MD5* processa mensagens de tamanho arbitrário através de blocos de 512 bits. Cada bloco de 512 bits é dividido em seis sub-blocos de 32 bits.

Em primeiro lugar, a mensagem a encriptar é completada com um bit 1 e n bits 0 até que seja divisível por 64 e tenha o menor número de bits (*padding*). Depois, uma representação da mensagem sem *padding* é adicionada ao resultado do *padding*. Estes dois passos fazem com que a mensagem resultante tenha um comprimento múltiplo de 512, necessário já que o *MD5* encripta a mensagem em blocos de 512 bits.

Depois, são inicializadas quatro variáveis de 32 bits:

$$A = 01234567; B = 89ABCDEF; C = FEDCBA98; D = 76543210; \quad (5.1)$$

Para o processamento do algoritmo são definidas quatro funções auxiliares que tomam como entrada três palavras de 32 bits e a sua saída é também uma palavra de 32 bits.

```

{
    printf ("%02x", md_value[i] );
    int cod = (int)md_value[i] ;
    passwordMD5.operator+=( QString::number (cod) );
}

```

É também dado ao utilizador a possibilidade de alterar a *palavra chave*. Para tal, o utilizador apenas necessita de preencher os campos da janela apresentada na Figura 5.3.

Figura 5.3: Formulário para Mudar *Palavra-chave*

Antes de iniciar a pesquisa sobre o *DOM* de utilizadores autorizados, a aplicação verifica se a nova *Palavra Chave* e a confirmação da *Palavra Chave* são iguais. Se forem iguais é gerada a respectiva chave *MD5* e iniciada a pesquisa. Caso exista um utilizador que verifique os dados introduzidos é efectuada a alteração da *Palavra Chave*. Não sendo possível fazer a alteração, o utilizador é informado do erro que ocorreu. O utilizador também é informado se a palavra-chave tiver sido alterada com sucesso.

5.3.1.1 Algoritmo MDA

Proposto em 1991 por Ron Rivest, o *MD5* é uma versão melhorada da função *hash MD4*. Ainda que mais complexo, o desenho e arquitectura do *MD5* é similar à do *MD4* e, como este, gera *hash's* de 128 bits [Schneier 1994]. O *MD5* processa mensagens de tamanho arbitrário através de blocos de 512 bits. Cada bloco de 512 bits é dividido em seis sub-blocos de 32 bits.

Em primeiro lugar, a mensagem a encriptar é completada com um bit 1 e n bits 0 até que seja divisível por 64 e tenha o menor número de bits (*padding*). Depois, uma representação da mensagem sem *padding* é adicionada ao resultado do *padding*. Estes dois passos fazem com que a mensagem resultante tenha um comprimento múltiplo de 512, necessário já que o *MD5* encripta a mensagem em blocos de 512 bits.

Depois, são inicializadas quatro variáveis de 32 bits:

$$A = 01234567; B = 89ABCDEF; C = FEDCBA98; D = 76543210; \quad (5.1)$$

Para o processamento do algoritmo são definidas quatro funções auxiliares que tomam como entrada três palavras de 32 bits e a sua saída é também uma palavra de 32 bits.

$$F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z) \quad (5.2)$$

$$G(X, Y, Z) = (X \wedge Z) \vee (X \wedge \neg Z) \quad (5.3)$$

$$H(X, Y, Z) = X \oplus Y \oplus Z \quad (5.4)$$

$$I(X, Y, Z) = Y \oplus (X \vee \neg Z) \quad (5.5)$$

Tom Berson, através de cripto-análise diferencial, conseguiu corromper uma das voltas do MD5 mas este ataque continuou longe de ser eficaz contra as quatro em conjunto [Schneier 1994]. Outro ataque bem sucedido, realizado por Bert den Boer e Antoon Bosselaers foi capaz de produzir colisões utilizando o MD5 como função de compressão. Ou seja, um dos princípios básicos do MD5 — não existir duas mensagens a gerar a mesma *message digest* — é quebrado. Contudo, estes dois ataques não são críticos e não devem afastar a utilização do MD5 [Schneier 1994].

O MD5 ainda é amplamente utilizado. Contudo, as últimas investigações descobriram vulnerabilidades que fazem com que se questione a sua utilização. Em Agosto do ano de 2004 Xiaoyun Wang, Dengguo Feng, Xuejia y Hongbo Yu anunciaram que tinham conseguido descobrir mais colisões de *hash* no MD5 [Wang et al. 2004]. Este ataque apenas levou uma hora num *cluster* IBM P690.

Apesar de não ser tão eficiente como algumas funções *hash* anteriores, o MD5 continua a ser bastante eficiente e as falhas de segurança não são críticas para a utilização empregue nesta aplicação já que chave MD5 é encriptada através do algoritmo *blowfish* quando é copiada para o ficheiro. Já como assinatura digital os criptoanalistas recomendam migrar os algoritmos de *hash* para funções RIPEMED-160 (recomendação ETSI³) ou SHA-2 [Medina e Pasamar 2004].

5.3.2 Verificação de Dados

A verificação de dados é um dos mecanismos mais importantes da aplicação já que um dos objectivos do seu desenvolvimento é a diminuição da taxa de erros na introdução dos dados.

Seguindo o definido nos casos de utilização do actor Parceiro, secção 3.2.1.3, foram implementadas dois tipos de verificações, “*duras*” e “*suaves*”, que correspondem aos dois tipos de restrições definidas.

Sobre os dados dos FORNECEDORES apenas foram implementadas restrições “*duras*”. Nos campos *Nome* e *NIF* foi implementada a restrição “*dura*” de não permitir que seja introduzido um FORNECEDOR com o nome ou número de contribuinte igual a um FORNECEDOR já existente. Sobre o *NIF* existe outra restrição “*dura*” que testa se o número de contribuinte é válido.

Já sobre os dados dos PEDIDOS DE REEMBOLSO existem verificações “*duras*” e verificações “*suaves*”. As verificações “*duras*” são feitas sempre que o utilizador altera o valor de uma célula. Quando o utilizador altera uma célula, a aplicação gera um evento que é capturado pela função `changedValueItemTable1`. Esta função verifica se o dado é válido. Se o dado não for válido, é apresentada ao utilizador uma mensagem de aviso e a regra que deve obedecer para cumprir a restrição. Depois de encerrada esta mensagem, a aplicação devolve ao utilizador a possibilidade de continuar a completar esse PEDIDO DE REEMBOLSO. No entanto, se o utilizador não corrigir o erro, a aplicação volta a mostrar a

³ *European Telecommunications Standard Institute*

mesma mensagem quando o utilizador tentar adicionar ou consultar outro PEDIDO DE REEMBOLSO. Nesta situação a aplicação não permite ao utilizar prosseguir com a introdução de mais PEDIDOS DE REEMBOLSO enquanto o utilizador não corrigir os dados.

De seguida apresenta-se uma tabela que indica que restrições “duras” foram implementadas para cada campo de um PEDIDO DE REEMBOLSO.

Atributo	Restrição “Dura”
<i>Fornecedor</i>	Número inteiro, maior que zero e que correspondam ao código de um FORNECEDOR existente.
<i>Documento</i>	Um ou mais caracteres que corresponda ao código de um dos DOCUMENTOS de quitação permitido no projecto Porto Digital.
<i>Conta POC</i>	Número inteiro de cinco dígitos que corresponda a uma das RÚBRICAS do Programa Oficial de Conta permitidas no projecto Porto Digital.
<i>Imputação</i>	Número entre 0 e 100 correspondendo à percentagem da despesa afecta ao projecto Porto Digital.
<i>Valor</i>	Número maior que zero.
<i>IVA</i>	Número maior que zero.
<i>Total</i>	Número maior que zero.

Tabela 5.1: Descrição das Restrições “Duras” Implementadas

Para verificar se os campos FORNECEDOR, DOCUMENTO e RÚBRICA estão correctos, a aplicação percorre os respectivos DOMs. Quando o valor introduzido é encontrado no respectivo DOM, a célula da tabela é actualizada para o nome do FORNECEDOR, DOCUMENTO ou RÚBRICAS. Optou-se por pedir a introdução do código em vez do nome dos atributos porque deste modo diminui-se a probabilidade de erro inerente à introdução pelo utilizador de uma cadeia de caracteres. Além disso torna-se mais rápido, sobretudo para utilizadores experientes.

A possibilidade de uma despesa ter uma *Imputação* de 0% serve para contemplar o caso de um PEDIDO DE REEMBOLSO de deduções à Segurança Social aquando do PEDIDO DE REEMBOLSO de um ordenado. Este tipo de despesa não é passível de reembolso, mas é obrigatório que o Parceiro apresente o comprovativo das obrigações sociais para que a despesa do ordenado seja aprovada.

As verificações “suaves” implementadas sobre os dados dos PEDIDOS DE REEMBOLSO são feitas quando o utilizador as solicita. Este tipo de verificações alerta para dados que não se enquadram nos casos comuns, mas ainda assim podem estar correctos. Por este facto a aplicação apenas solicita ao utilizador que confirme que não há nenhum equívoco nos dados. Este tipo de verificação só é feita sobre os PEDIDOS DE REEMBOLSO ainda não enviados para a Associação Porto Digital. Sobre os outros pedidos assume-se que esta verificação já foi realizada e, como o Parceiro não tem possibilidade de alterar dados já enviados à Associação Porto Digital, o aviso gerado seria inconsequente.

A função implementada para realizar as verificações “suaves” percorre os elementos do DOM e, para os elementos não enviados à APD, analisa os valores de *Imputação*, *Valor*, *IVA* e *Total*; verifica se os valor de *Imputação* se enquadra num dos três regimes de IVA e se a soma do *valor* com o *IVA* é igual ao *Total*. Sempre que um pedido não enviado à APD

não cumpre uma destas regras, é adicionada uma mensagem a uma cadeia de caracteres que será apresentada ao utilizador no final do processo de verificações.

5.3.3 Consistência e Persistência de Dados

Para garantir que em caso de falha a aplicação é capaz de recuperar o estado em que se encontrava, é mantido em ficheiro o conjunto de operações realizadas, e não gravadas, sobre os dados dos FORNECEDORES e dos PEDIDOS DE REEMBOLSO. Assim, sempre que o utilizador realiza uma operação são realizadas duas tarefas. Uma consiste em reflectir na interface gráfica a alteração efectuada. A outra, o cerne da mecanismo de recuperação, consiste em adicionar um nó que representa a alteração efectuada a um *DOM*. Este *DOM* representa o conjunto de alterações efectuadas aos dados e que ainda não foram guardadas. De seguida, este *DOM* é guardado em ficheiro. Para cada tipo de dados, FORNECEDORES e PEDIDOS DE REEMBOLSO, existe um *DOM* e respectivo ficheiro de alterações.

Quando a aplicação é iniciada verifica o número de elementos que possui o elemento raiz dos *DOMs* que mantêm as alterações não gravadas. Se os elementos raiz tiverem filhos é porque existem operações efectuadas na última execução que não foram gravadas. Então o utilizador é avisado que a aplicação não terá terminado correctamente a última execução e que possui dados para recuperar. Após apresentação do ficheiro *XML* com a informação das operações não gravadas, é dada a possibilidade ao utilizador de recuperar os dados ou ignorar as alterações. Caso opte por ignorar, serão perdidas as alterações e não há qualquer alteração aos *DOMs* dos FORNECEDORES e PEDIDOS DE REEMBOLSO lidos do ficheiro. Caso o utilizador opte por recuperar as alterações, é utilizado o mecanismo de *merge* dos *DOMs* de conteúdo e de alterações para obter o *DOM* dos dados actualizados. Este *DOM* é então gravado para ficheiro e passa a ser considerado o *DOM* dos dados inicialmente lidos de ficheiro. Quanto aos *DOMs* de alterações, estes são gravados para ficheiro apenas com o elemento raiz.

Para a apresentação das alterações foi criada uma *tabWidget*, permitindo ao utilizador mudar de *widget* e consultar as alterações de um ou outro ficheiro. Nesta *tabWidget* também é possível consultar os ALERTAS enviados pela Associação Porto Digital para alteração de dados dos FORNECEDORES ou PEDIDOS DE REEMBOLSO. O modo de incorporação da informação contida nos ALERTAS da APD é gémeo do utilizado para recuperação em caso de falha da aplicação. Assim é realizado um *merge* entre o *DOM* que representa os ALERTAS e o *DOM* que representa o ficheiro de dados a que se refere o ALERTA. A única diferença é que o Parceiro, não poderá descartar os ALERTAS sendo forçado a executar o *merge* para aceder à aplicação.

5.3.3.1 Gravação

O conjunto de operações realizadas para executar os ALERTA, realiza-se de igual forma quando, durante a execução, o utilizador guarda os dados da aplicação para ficheiro.

Quando o utilizador grava, ou descarta as alterações não gravadas antes de sair da aplicação, esta não gerará nenhuma mensagem de aviso na próxima utilização, porque os *DOMs* das alterações dos dados dos FORNECEDORES e dos PEDIDOS DE REEMBOLSO estarão vazios.

5.3.3.2 Estrutura do DOM de Alterações

O *DOM* que representa as alterações sobre os FORNECEDORES tem o elemento raiz <alteracoes> e o que representa os PEDIDOS DE REEMBOLSO tem o elemento raiz <alteracoesPP>.

Além do elemento raiz <alteracoes> ou <alteracoesPP>, o ficheiro terá 0, 1 ou mais elementos do tipo:

48 <Exemplo: Elemento “operacao” de um DOM de Alterações 48>≡

```
<"operacao">
  <"tipoDeDado" "atributos"/>
  <"tipoDeDado" "atributos"/>
  <"tipoDeDado" "atributos"/>
</"operacao">
```

A etiqueta <"operacao"> pode ser <adicionar>, <editar> ou <eliminar> dependendo da operação efectuada; “tipoDeDado” pode ser FORNECEDOR ou PEDIDO DE REEMBOLSO dependendo do tipo de dado e, “atributos” é a lista de atributos necessária para realizar a operação “operacao” sobre o dado “tipoDeDado”. Nos casos das operações *adicionar* e *editar*, “atributos” é a lista de todos os atributos do “tipoDeDado” em questão. No caso da operação *eliminar*, “atributos” é apenas o código identificador do “tipoDeDado”.

5.4 Funcionalidades

Nas secções seguintes são descritas as principais funcionalidades da aplicação:

- pesquisas e filtros, descritos na secção 5.4.1;
- sincronização, descrita na secção 5.4.2;
- relatórios disponíveis, descritos na secção 5.4.3.

5.4.1 Pesquisas e Filtros

As funcionalidades de *Pesquisa* e *Filtros* estão disponíveis sobre os dados dos FORNECEDORES.

5.4.1.1 Pesquisas

Para realizar uma pesquisa, o utilizador deve definir os campos de pesquisa:

- *coluna* sobre a qual se irá processar a pesquisa;
- *critério* de pesquisa — igual, contém ou diferente;
- *valor a pesquisar*;
- *case sensitive* (S/N).

O utilizador inicia o processo de pesquisa pressionando o botão *Pesquisar*.

Para responder ao pedido do utilizador, a aplicação abre o ficheiro de alterações dos FORNECEDORES para consultar possíveis alterações aos dados que tem em memória. Os dados sobre FORNECEDORES que possui em memória são os dados que inicialmente foram

lidos de ficheiro. Assim, para ter em memória uma estrutura *DOM* que corresponda aos dados apresentados na interface gráfica, a aplicação efectua um *merge* do *DOM* inicialmente carregado para memória e o *DOM* das alterações realizadas e ainda não gravadas. No caso do *DOM* das alterações estar vazio, a função de *merge* devolve, automaticamente, o *DOM* dos fornecedores que recebe.

Após ter em memória um *DOM* que representa os dados apresentados ao utilizador através da interface gráfica, a aplicação percorre essa estrutura de dados e, sempre que encontra um elemento que satisfaça as condições definidas para a pesquisa, selecciona esse FORNECEDOR na interface gráfica. Ao terminar de percorrer este *DOM*, estarão seleccionados visualmente todos os FORNECEDORES que satisfazem a pesquisa.

Na Figura 5.4 pode ver-se este mecanismo esquematizado em diagrama de estados.

Diagrama de Estados de uma Pesquisa

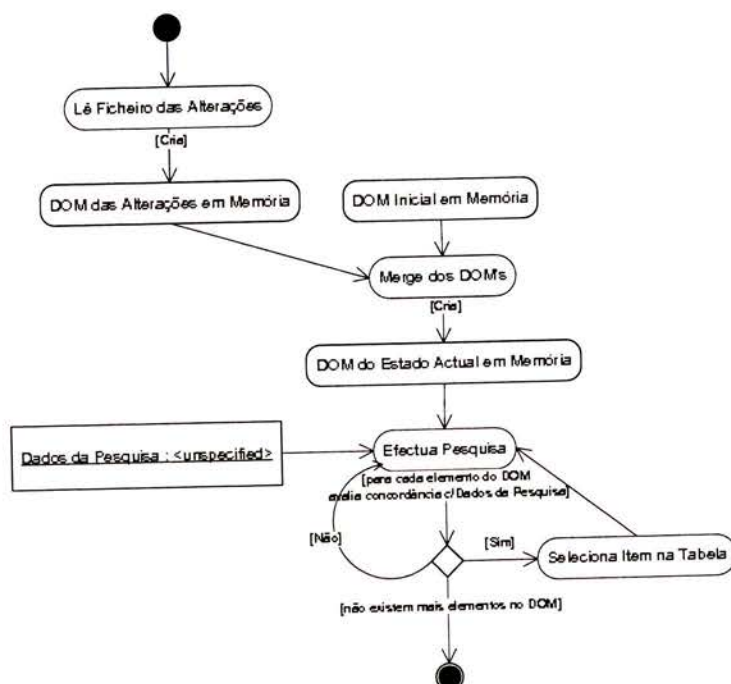


Figura 5.4: Diagrama de Estados de uma Pesquisa

Não há, como se verá no caso dos filtros, a necessidade de desactivar a pesquisa já que os *items* são automaticamente deseleccionados quando outro objecto da interface gráfica é seleccionado.

De seguida é apresentado o código que executa uma pesquisa. Por existir um relatório interno à empresa contento todo o código desenvolvido, aqui apenas é apresentado o código necessário para fazer uma pesquisa cujo critério é *Igual*, já que para os outros critérios o procedimento é semelhante.

49 *(Exemplo: Código de Pesquisa de Fornecedores 49)*≡

```

void dlgFornecedores::executaPesquisa( QString coluna, QString criterio,
                                     QString valor, bool sensitive )
{
    //DOM Inicial
    QDomElement rootFor = fornecedores->documentElement();

```

```
//DOM Alterações
QDomElement rootAlt = altFornecedor->documentElement();

//merge
merge ( fornecedores, altFornecedor, FALSE );
//DOM Actual
QDomElement root = fornecedores->documentElement();

// Coluna sobre a qual vai ser efecutada a pesquisa
coluna = traduccaoColuna( coluna );
QDomNode n = root.firstChild();
QDomElement e = n.toElement();

//Deseleciona todos os items da QListView para o caso de ainda estar activa
outra pesquisa
deselecionaItems();

// IGUAL
if ( criterio == "Igual" )
{
    //pesquisa case-sensitive
    if ( sensitive )
    {
        //enquanto houverem mais elementos a percorrer
        while ( !n.isNull() )
        {
            //se comparação verdadeira
            if ( !QString::compare( valor, e.attribute( coluna, "" ) ) )
                //seleciona item na qlistview
                selecionaItem( e.attribute( "codigo", "" ) );

            //proximo elemento do DOM
            n = n.nextSibling();
            e = n.toElement();
        }
    }
    //pesquisa não case-sensitive
    else
    {
        while ( !n.isNull() )
        {
            if ( !QString::compare( valor.lower(), e.attribute( coluna, "" ).lower() ) )
                selecionaItem( e.attribute( "codigo", "" ) );
            n = n.nextSibling();
            e = n.toElement();
        }
    }
}

// CONTEM
if ( criterio == "Contem" )
```

```

[...]
```

```

// DIFERENTE
if ( criterio == "Diferente" )
[...]
```

```

}
```

5.4.1.2 Filtros

O filtro funciona como a conjunção das pesquisas com o critério *contém* dos campos preenchidos. Ou seja, enquanto a pesquisa com critério *contém* é feita sobre um campo, num filtro é possível fazer este tipo de pesquisa, mas para vários campos simultaneamente. Matematicamente, um filtro retorna os elementos que satisfazem a condição:

$$R1 \wedge R2 \wedge R3 \wedge R4 \wedge R5 \wedge R6 \quad (5.6)$$

Onde:

$$R1 = \text{elemento.attribute("codigo").contains("codigoIntroduzidoPeloUtilizador")} \quad (5.7)$$

$$R2 = \text{elemento.attribute("nome").contains("nomeIntroduzidoPeloUtilizador")} \quad (5.8)$$

$$R3 = \text{elemento.attribute("nif").contains("nifIntroduzidoPeloUtilizador")} \quad (5.9)$$

$$R4 = \text{elemento.attribute("mercado").contains("mercadoEscolhidoPeloUtilizador")} \quad (5.10)$$

$$R5 = \text{elemento.attribute("enviado").contains("enviadoEscolhidoPeloUtilizador")} \quad (5.11)$$

$$R6 = \text{elemento.attribute("validado").contains("validadoEscolhidoPeloUtilizador")} \quad (5.12)$$

Quando o utilizador activa um filtro, é realizado um *merge* entre o *DOM* dos dados carregados inicialmente para memória e o *DOM* que guarda as alterações ainda não guardadas em ficheiro. No final do *merge* obtemos o *DOM* que representa os dados que estão a ser apresentados ao utilizador. Este passo de *merge* é igual ao que é aplicado na pesquisa.

É sobre este *DOM* que são feitas as operações necessárias para criar o *DOM* que resulta da passagem do filtro. Para tal, a aplicação percorre o *DOM* e, para cada elemento verifica se este cumpre a condição 5.6. Caso este não cumpra a condição é removido do *DOM*. No final deste processo fica-se com um *DOM* que contém apenas os elementos coerentes com o filtro activado. Finalmente, é limpa a *QListView* que apresenta os dados dos fornecedores e chamada a função implementada para a preencher. Esta função tem como argumento a raiz do *DOM* que se pretende apresentar.

51 <Exemplo: Código de Actualização do Objecto de Visualização de Fornecedores 51>≡

```

//limpa estrutura QListView
//que será posteriormente preenchida com os valores do DOM Filtrado
listaFornecedores->clear();

// carrega QListView com o QDomDocument representativo do filtro aplicado
carregaQListView ( rootFor )
```

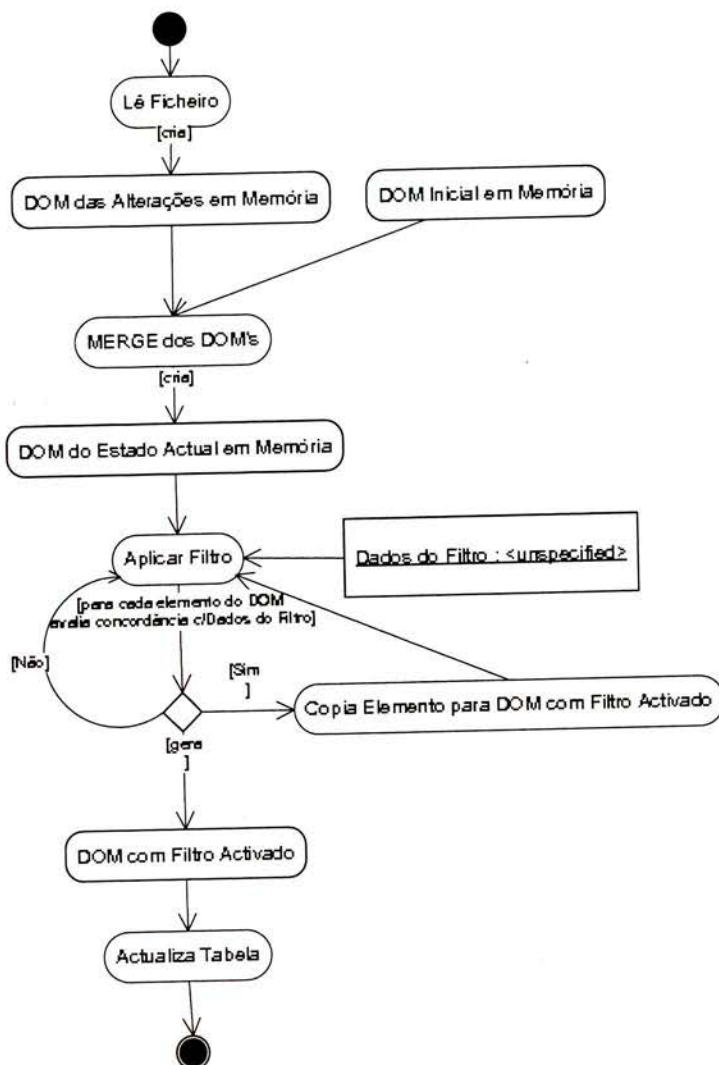


Figura 5.5: Diagrama de Estados de um Filtro

Este processo é apresentado na Figura 5.5.

No caso dos filtros é necessário desactivá-los. Nesta situação a aplicação recorre ao *merge* do *DOM* dos dados inicialmente lidos para memória com o *DOM* das alterações realizadas e não gravadas para voltar a obter o *DOM* do estado actual. Para terminar apenas troca os dados contidos na *QListView* pelo *DOM* resultante do *merge*.

5.4.2 Sincronização

O processo de sincronização de dados é baseado num sistema de comunicação através de canais de comunicação (*socket*). Para tal foi utilizado um pequeno servidor disponibilizado no sítio *Web* da Trolltech.

Esta aplicação tem a classe **ServerInfo** que disponibiliza a interface gráfica da aplicação e cria um objecto da classe **SimpleServer**. Esta classe trata novos pedidos de conexão e para cada cliente que se conecta cria um novo objecto **ClientSocket**. É a classe **ClientSocket** que é responsável pela comunicação entre cliente e servidor definindo os

sinais para leitura dos dados enviados e para eliminação do objecto `ClientSocket` após a recepção do sinal `connectionClosed`.

Toda a informação recebida pelo servidor é apresentada na sua interface.

Na aplicação desenvolvida, o módulo de cliente do servidor foi implementado numa classe à parte. Esta classe disponibiliza a interface e toda a lógica necessária para a sincronização dos dados. Primeiro são preparados os dados dos FORNECEDORES e PEDIDOS DE REEMBOLSO que têm de ser enviados para a Associação Porto Digital. Para tal, a aplicação percorre as estruturas *DOMs* dos FORNECEDORES e PEDIDOS DE REEMBOLSO criando uma cópia de todos os elementos que ainda não tenham sido enviados. Depois de feita a cópia, esta é movida para um *DOM* temporário, e apenas em memória, que é finalmente enviado pelo canal de comunicação. No caso de alguma das funções de preparação de dados retornar um *booleano FALSE* a aplicação avisa o utilizador da ocorrência e cancela o processo de sincronização.

53a *<Exemplo: Código de Construção dos dados dos Fornecedores para Sincronização 53a>≡*

```
if ( !preparaDadosFornecedores() )
{
    QMessageBox::warning ( this,
                           trUtf8("Sincronização - Erro"),
                           trUtf8("Ocorreu um erro na preparação de dados dos
                                   Fornecedoress\n
                                   Não existirá sincronização para estes
                                   dados"),
                           "Ok" );
}
```

Tendo preparada a estrutura de dados para enviar, a aplicação inicia o processo de criação de um canal de comunicação. Para abrir o canal de comunicação é chamada a função `abreSocket()`. Nesta função também é feita a ligação entre os eventos disponibilizados pela classe `QSocket` que garantem que o canal de comunicação é correctamente aberto, encerrado e que não há qualquer erro durante o envio dos dados, com a função `socketSlot`. Para garantir que as funções serão realmente executadas aquando da execução dos eventos, é testado o retorno das funções que criam esta ligação; retornado *FALSE* em caso de alguma falha. Desta forma é garantido que o canal de comunicação está aberto e que retornará qualquer erro que ocorra durante a aplicação. A função `socketSlot` altera a variável global `wait` para *FALSE* indicando à função principal que o último passo foi executado sem erro.

53b *<Exemplo: Código de Abertura de um Canal de Comunicação 53b>≡*

```
bool dlgSincronizacao::abreSocket()
{
    if ( ( socket = new QSocket( this ) ) == FALSE )
        return FALSE;
    if ( ! connect( socket, SIGNAL(connected()), SLOT( socketSlot() ) ) )
        return FALSE;
    if ( ! connect( socket, SIGNAL(connectionClosed()), SLOT( socketSlot() ) ) )
        return FALSE;
    if ( ! connect( socket, SIGNAL(error(int)), SLOT( socketSlot() ) ) )
        return FALSE;
    socket->connectToHost( QString("localhost"), Q_UINT16( 4242 ) );
    return TRUE;
}
```

No caso de uma chamada da função principal retornar *FALSE* ou a variável global *wait* estiver a *TRUE* a aplicação apresenta uma mensagem de erro respectiva e aborta a sincronização.

54 *(Exemplo: Erro na Abertura de um Canal de Comunicação 54) ≡*

```
if ( abreSocket() == FALSE || wait == TRUE )
{
    QMessageBox::warning( this,
                          trUtf8("Sincronização - Erro"),
                          trUtf8("Não foi possível estabelecer contacto com o servidor
                                da Associação Porto Digital.
                                \nPor favor, tente mais tarde.
                                \nSe o problema persistir contacte o Suporte Técnico."
                                Ok" );
    startPushButton->setDisabled( TRUE );
    return;
}
```

Ainda que ocorra um erro ao fechar o canal de comunicação, a aplicação continua com o processo de sincronização apenas enviando ao utilizador uma mensagem de aviso.

Após a aplicação ter enviado todos os dados correctamente, procede à actualização dos dados que acabou de enviar. Esta actualização resume-se à alteração do campo *enviado* no campo dos elementos enviados. Para tal a aplicação lê o código dos FORNECEDORES e PEDIDOS DE REEMBOLSO enviados. Para cada leitura, vai procurar esse elemento na estrutura *DOM* dos FORNECEDORES ou PEDIDOS DE REEMBOLSO, dependendo do elemento, e altera o campo *enviado* para *"Sim"*.

Com a conclusão deste procedimento é concluído o processo de sincronização.

5.4.3 Relatórios

As decisões dos profissionais exigem cada vez mais e melhor informação. Esta informação é gerada a partir de um volume de dados cada vez maior que excede os recursos temporais e financeiros necessários para que se realize através de recursos humanos. Os Sistema de Apoio à Decisão (SAD) permitem lidar com problemas de decisão cuja dimensão ultrapassa a capacidade cognitiva normal. Com o objectivo de introduzir a base de um Sistema de Apoio à Decisão disponível quer aos Parceiros, quer à Associação Porto Digital, foi implementado um relatório e definidos outros dois.

O relatório implementado é destinado aos Parceiros do projecto Porto Digital e agrega informação sobre PEDIDOS DE REEMBOLSO. Através deste relatório o Parceiro tem acesso a seis indicadores:

- número de PEDIDOS DE REEMBOLSO por enviar;
- número de PEDIDOS DE REEMBOLSO já enviados;
- número de PEDIDOS DE REEMBOLSO aprovados;
- valor do conjunto de PEDIDOS DE REEMBOLSO por enviar;
- valor do conjunto de PEDIDOS DE REEMBOLSO enviados;
- valor do conjunto de PEDIDOS DE REEMBOLSO aprovados.

Com estes indicadores pretende-se que o Parceiro saiba o valor de reembolso que já recebeu e possa prever o valor que será reembolsado no curto e médio prazo.

Quanto aos relatórios definidos, mas ainda não implementados, são descritos na secção 6.6 do capítulo destinado à descrição de melhoramentos da aplicação.

5.5 Interface

A interface gráfica foi desenvolvida recorrendo à ferramenta *QtDesigner*. Através desta ferramenta são adicionados vários objectos gráficos, desde caixas de selecção, texto não editável, campos de inserção de dados ou botões a uma janela *QWidget*. Quando o projecto é gravado é criado um conjunto de ficheiros *XML* que descrevem a interface definida. Durante a compilação esta informação é processada e gerada a interface gráfica.

Os únicos objectos gráficos implementados via código *Qt* resumem-se a mensagens de aviso ou informação ao utilizador, como por exemplo, no caso em que o utilizador tenta fechar a janela de FORNECEDORES e tem dados que ainda não foram gravados. Neste caso é apresentado uma mensagem de aviso perguntando ao utilizador se pretende cancelar a operação e gravar ou rejeitar os dados não gravados.

```
55  <Exemplo: Código de uma Mensagem Apresentada ao Utilizador 55>≡
    QMessageBox gravarAoSair( "Fornecedores - Sair",
        trUtf8("Existem alterações aos fornecedores que
                ainda não foram gravadas.\n"
                "Gravar antes de sair?"),
        QMessageBox::Question,
        QMessageBox::Yes | QMessageBox::Default,
        QMessageBox::No,
        QMessageBox::Cancel | QMessageBox::Escape );
    gravarAoSair.setButtonText( QMessageBox::Yes, "Sim" );
    gravarAoSair.setButtonText( QMessageBox::No, trUtf8("Não") );
    gravarAoSair.setButtonText( QMessageBox::Cancel, "Cancelar" );
    switch( gravarAoSair.exec() )
    {
    case QMessageBox::Yes:
    {
        gravaFornecedores();
        ce->accept();
        break;
    }
    case QMessageBox::No:
    {
        eliminaAlteracoes();
        ce->accept();
        break;
    }
    case QMessageBox::Cancel:
    {
        ce->ignore();
        break;
    }
    }
```

Tipicamente, durante a aplicação apenas são alteradas as características de um tipo de objecto. Por exemplo, colocar a linha da *QTable* que apresenta os dados de PEDIDOS DE REEMBOLSO não editável para um PEDIDO DE REEMBOLSO anteriormente enviado para a Associação Porto Digital.

Com o *QtDesigner* também é possível ligar determinados eventos a funções implementadas pelo programador, a funções de determinados objectos, ou mesmo a outros eventos. Ou seja, é possível indicar graficamente que função deve ser processada quando ocorre um determinado evento. Este mecanismo é utilizado, por exemplo, para conectar o evento de pressionar um botão à execução de uma determinada função.

Durante o desenvolvimento da interface teve-se especial atenção às questões de usabilidade. Por este motivo foi implementado um navegador que permite ao utilizador aceder às principais funções activando-as através de um simples *click*. Este navegador é apresentado na Figura 5.6.



Figura 5.6: Navegador da Aplicação

Para satisfazer utilizadores mais experientes criou-se um conjunto de teclas de atalho que permitem navegar rapidamente entre menus. Na Tabela 5.2 é apresentada a lista de teclas de atalho e respectiva função.

Em termos de implementação, apenas foi desenvolvido código para os casos das teclas *F1*, *Esc* e botão *X* no canto superior direito de todas as janelas. Nos dois últimos casos foi desenvolvido código específico porque era necessário analisar se existia informação não gravada e apresentar ao utilizador a situação. No primeiro caso era chamada a própria ajuda do *QtDesigner* quando se pretendia adicionar esta tecla como tecla de atalho. Nos restantes casos foi possível definir no *QtDesigner* a propriedade *accel* que guarda a tecla

Tecla de Atalho	Função
F1	Abre janela de <i>Ajuda</i>
F2	Abre janela de <i>Rúbricas</i>
F3	Abre janela de <i>Documentos</i>
F4	Abre janela de <i>Fornecedores</i>
F5	Abre janela de <i>Alertas</i>
F6	Abre janela de <i>Sincronização</i>
Esc	Gera sinal para encerramento da janela activa
Alt-letra	Activa o botão que possua a letra <i>letra</i> sublinhada

Tabela 5.2: Tabela das Teclas de Atalho

de atalho para esse objecto, um botão, por exemplo, que depois de activado chama uma função.

Como acontece na maioria das aplicações com interface gráfica, foi disponibilizado uma barra de ferramentas na zona superior da janela principal que se apresenta na Figura 5.7.



Figura 5.7: Barra de Ferramentas

De salientar que houve um grande cuidado na estética e usabilidade de todos os objectos gráficos tendo-se criados *icons* que permitem ao utilizador aperceber-se intuitivamente da função executada quando activado um determinado elemento gráfico. É disso exemplo a janela de gestão de FORNECEDORES apresentada na Figura 5.8.

5.6 Resumo

Este capítulo descreve a decisão sobre o tipo de base de dados *XML* utilizada, a decisão sobre a estrutura e validação dos documentos de *XML*. São também descritos os principais mecanismos e as principais funcionalidades implementadas.

Em detrimento de uma base de dados *XML* foi implemetando uma repositório de documentos *XML*.

Quanto à estrutura dos documentos de *XML* optou-se por colocar as características de um objecto da aplicação como atributos de um elemento do *DOM*. Deste modo pretendeu-se acelerar o processo de pesquisa já que se obtém a lista de características de um objecto da aplicação em níveis da pesquisa superiores. No entanto, e dado que a verificação da estrutura do *DOM* apenas testa a correcção das etiquetas dos elementos, caso ocorra algum erro na lista de atributos, a aplicação não gerará nenhum ALERTA. Apenas não apresentará os dados dos atributos que estejam incorrectos.

Para a autenticação dos utilizadores foi utilizada a função de *hash MD5*. A encriptação dos documentos *XML* foi conseguida através do algoritmo de chave secreta *Blowfish*.

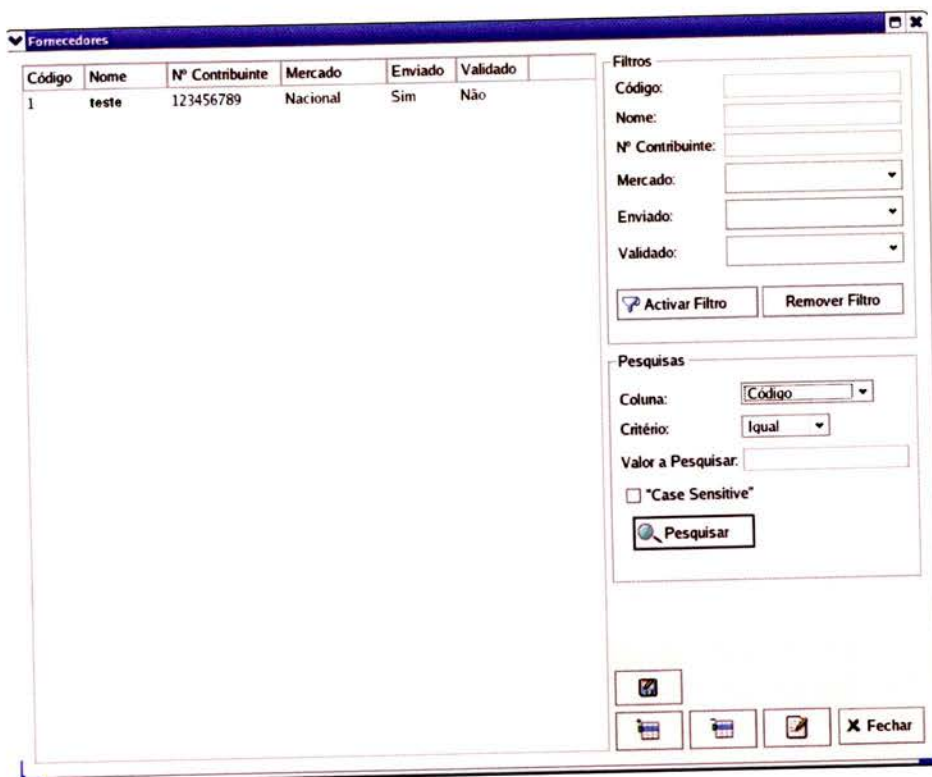


Figura 5.8: Janela de FORNECEDORES

Neste momento é já possível obter uma visão geral da aplicação estando disponíveis várias verificações de dados, pesquisas, filtros e relatórios. Também está implementado o mecanismo de envio de dados para o servidor da Associação Porto Digital e o mecanismo de tratamento de ALERTAS recebidos. Contudo, a recepção de ALERTAS ainda não está implementada.

Toda a interface e código escrito foram desenvolvidos através da plataforma de desenvolvimento *QtDesigner*. Esta ferramenta tecnológica gera automaticamente os ficheiros necessários para a compilação do projecto. Neste projecto apenas foi necessário pequenas alterações ao *Makefile* para incorporar as bibliotecas *lssl* e *lcrypto* necessárias para a utilização da função *hash* MD5 e o algoritmo de criptografia *Blowfish*.

Capítulo 6

Melhoramentos

Neste capítulo são apresentados possíveis melhoramentos da aplicação desenvolvida.

6.1 Introdução

Nas seguintes secções são apresentadas possíveis melhorias na eficiência e segurança da aplicação, assim como são descritos dois novos relatórios a incluir na aplicação. Também são analisadas melhorias ao código implementado e soluções para a realização de testes unitários.

6.2 Gestão de Memória

Para melhoramento futuro é necessário avaliar a memória consumida pela aplicação. Uma forma de melhorar a utilização de recursos da aplicação é, quando o utilizador tem acesso aos PEDIDOS DE REEMBOLSO, não manter em memória o *DOM* dos FORNECEDORES, RÚBRICAS e DOCUMENTOS. Esta informação é necessária para a implementação das verificações “*duras*” aquando da introdução de novos PEDIDOS DE REEMBOLSO. Contudo, em vez de manter em memória estes três *DOMs*, poder-se-á implementar um filtro *SAX* que percorra os ficheiros *XML* e gere eventos sempre que passe pelo código identificador do FORNECEDOR, RÚBRICA ou DOCUMENTO. Depois, seriam construídos três vectores onde se guardaria a lista dos códigos permitidos. Estes vectores seriam utilizados nas verificações “*suaves*”.

Outro modo de minorar a utilização de memória, seria criar um mecanismo semelhante a este mas para os PEDIDOS DE REEMBOLSO. Em vez da aplicação carregar para a memória todos os PEDIDOS DE REEMBOLSO manteria em memória apenas os pedidos relativos ao projecto activo. Ainda que a solução permita diminuir a utilização de memória, é preciso notar que se recorreria mais vezes à leitura do ficheiro. Enquanto que actualmente apenas é necessário fazer uma leitura, com a proposta agora apresentada tornar-se-ia preciso ler o ficheiro dos PEDIDOS DE REEMBOLSO sempre que o utilizador mudasse de projecto. Por este motivo são necessários testes de performance para assegurar uma melhoria global da eficiência da aplicação.

6.3 Mecanismos de Segurança

Na implementação do algoritmo *Blowfish* não existiu cuidado na definição da chave e do vector de inicialização **iv**. No entanto, a segurança de uma mensagem encriptada depende, não só do algoritmo escolhido, como também da chave de encriptação elegida. Um dos factores que determina de forma mais importante a força de encriptação de uma chave é a qualidade dos números aleatórios usados para a gerar. Aliás, a geração de verdadeiros números aleatórios continua a ser um dos processos mais difíceis na Ciência da Computação. Assim, antes de a aplicação ser colocada em produção, tem de ser analisada e revista a escolha destes valores. Por exemplo, poder-se-á utilizar uma função geradora de números aleatórios que tenha como entrada o *checksum* dos dados de uma secção da memória dedicada aos processos em execução. Deste modo, é quase impossível descobrir o valor utilizado para gerar o número aleatório, o que poderia acontecer se a entrada da função fosse, por exemplo, o formato interno da hora interna do sistema operativo.

Neste momento, ainda persiste um erro durante o processo de cifra/decifra dos dados com o algoritmo *Blowfish*. O problema reside no processo de escrita do ficheiro. Durante a encriptação para ficheiro de uma cadeia de caracteres encriptada é possível que esteja a ser guardado um ou mais bits que impedem a correcta decifra. Se após a encriptação e persistência para ficheiro de uma cadeia de caracteres, for executado o comando `openssl bf -in cipher.bin -K`

`000102030405060708090A0B0C0D0E0F -iv 0102030405060708 -d` o resultado da decifração será um conjunto variável de espaços brancos, seguidos do resto da mensagem correctamente decifrada. Já se essa decifração for realizada pela aplicação através da leitura dos dados que se encontram no ficheiro, então a decifração falha. No entanto, se se executar a encriptação, e de seguida, se executar a decifração com a estrutura resultante da encriptação, então o processo de encriptação e decifração ocorre sem erros. Por este motivo se aponta que a falha se encontra no processo de escrita ou leitura do ficheiro.

Já durante o processo de sincronização de dados entre um Parceiro e a APD e a APD e o POS.C é também obrigatória a protecção dos dados enquanto atravessam um meio não seguro como é a Internet. Para solucionar este problema existem diversos caminhos que devem ser analisados. Pode-se optar por uma solução de envio dos dados em texto aberto através de canais seguros ou então recorrer a soluções criptográficas.

A implementação de canais seguros entre os vários intervenientes na comunicação é bastante simplificada com a biblioteca *QtSSL* do *Qt* e substituiria os canais de comunicação agora implementados através da biblioteca *QtSocket*. Em termos de soluções criptográficas pode-se optar por uma solução baseada em encriptação de chave pública ou secreta. Em ambos os casos, sempre que alguma das aplicações fosse enviar dados pela Internet, teria de ler a informação de ficheiro, decifrá-la e voltar a cifrá-la antes de enviar para o destinatário. Este processo de encriptação e decifração é obrigatório porque a aplicação, antes de encriptar e enviar os dados, necessita de definir qual é a informação a sincronizar.

Na criptografia com chave pública é usado um esquema assimétrico com um par de chaves para encriptação: uma chave pública que é utilizada para encriptar os dados; e uma chave secreta correspondente para decifração dos dados. A chave secreta apenas é conhecida pelo receptor da mensagem. As duas chaves não estão relacionadas entre si, e para segurança do sistema, deve ser computacionalmente impossível derivar a chave secreta a partir da chave pública. Assim, qualquer aplicação pode enviar dados para o destinatário bastando para tal conhecer a chave pública do destinatário e o algoritmo utilizado. A disponibilização das chaves dos intervenientes pode ser feita através de um

serviço acedido automaticamente por um canal de comunicação ou *webservice*. Deste modo, poder-se-ia alterar as chaves públicas de encriptação regularmente e sem grande esforço. Apenas seria necessário alterar simultaneamente a chave pública facultada no servidor e a secreta guardada em ficheiro no proprietário da chave pública alterada. Alguns exemplos de sistemas de criptografia com chave pública são o *ElGamal*, *RSA*, *Diffie-Hellman* e *DSA*. Para encriptação e assinatura digital o *RSA* é fácil de implementar em *C* com a biblioteca *openssl*. O *ElGamal* é uma boa solução para encriptação e o *DSA* é usado para assinatura digitais [Schneier 1994].

Mesmo soluções de encriptação em chave secreta podem ser consideradas para solucionar o problema de transmissão de dados através da Internet. O problema é que neste tipo de solução é necessário que o emissor e receptor partilhem a chave de encriptação/decifração. Esta negociação pode ocorrer através de canais seguros e depois de acordada a chave de encriptação, a comunicação passar-se-ia a fazer sobre canal aberto. Outra hipótese na área de encriptação simétrica é implementar um mecanismo de chave pública para a negociação da chave secreta e depois, utilizar encriptação simétrica para troca de dados. No caso de se optar por esta solução, o algoritmo de *Diffie-Hellman*, pelas suas características, é o mais indicado.

Quanto à definição de chaves é evidente que chaves maiores serão criptograficamente mais seguras, mas farão com que o processo de encriptação/decifração se torne mais demorado. Uma solução é combinar chaves não excessivamente longas com a utilização de canais seguros ou assinatura digital.

Antes de ser tomada qualquer decisão sobre o modo como proteger os dados que serão transmitidos via *Web* é necessário identificar quem poderá estar interessado em ler essa informação, se estão determinados no seu objectivo e que recursos têm disponíveis.

6.4 Mecanismos para Depuração de Erros

O *Qt* vem com algumas funcionalidades que permitem facilitar a depuração de erros da aplicação. Através da utilização das funções *qDebug*, *qWarning* e *qFatal* é fácil determinar em que ponto do programa ocorreu uma falha. É possível, por exemplo, usar uma destas funções para determinar, num determinado momento, se uma declaração *if* é *TRUE* ou quantas vezes foi executado um ciclo *while*. Com a utilização destas funções é mais simples evitar e corrigir erros.

A função *qDebug()* imprime, para o monitor em *Linux* e para o depurador de erros (*debugger*) no *Windows*, a mensagem do erro que ocorreu o que permite obter informação sobre o estado do programa. A função *qDebug()* recebe um ou mais argumentos; o primeiro é uma cadeia de caracteres, e os restantes são argumentos que devem ser inseridos nessa cadeia de caracteres. O funcionamento é idêntico ao da função *printf()* de *C++*.

61 *<Exemplo: Função qDebug() 61>*≡
`qDebug( "This is the widget's ID = %x", myWidget->is() );`

Contudo, para uma utilização eficiente da função *qDebug()* é necessário que as chamadas a esta função sejam feitas no local certo do programa.

A função *qWarning()* funciona como a função *qDebug()*. Todavia, da perspectiva do programador existe uma diferença importante entre as duas. Ao utilizar a função *qDebug()* para relatar informação sobre um evento e o *qWarning()* para informar sobre uma falha do programa, cria-se uma clara distinção entre informações sobre eventos e falhas da aplicação.

De seguida apresenta-se um exemplo da utilização do *qWarning()*.

62a *(Exemplo: Função qWarning() 62a)*≡
 QPushButton b1("Hello!", this);
 if (!QPushButton)
 qWarning("O objecto QPushButton b1 não foi criado correctamente!");

Assim, se houver algum problema na criação do objecto *b1* será apresentada a mensagem de aviso.

A última função para depuração de erros disponibilizada pelo *Qt* é o *qFatal()*. Funciona como as funções *qDebug()* e *qWarning()* mas quando chamada também provoca o encerramento da aplicação. Por este motivo só deve ser usada quando ocorre uma falha que comprometa a continuação da execução da aplicação.

Para mais fácil identificação da função de depuração de erro que foi chamada, deve-se iniciar a mensagem de retorno com o nome da respectiva função, ou seja, deve iniciar-se a mensagem com: "*DEBUG:* ", "*WARNING:* " ou "*FATAL:* ".

Além destas três funções, o *Qt* oferece várias *macros* para depuração de erros através da classe *qglobal.h*. As duas *macros* de depuração de erros mais importantes são a *ASSERT()* e a *CHECK_PTR()*. Ambas, tal como as restantes *macros* disponíveis em *qglobal.h*, utilizam as funções de depuração de erros para gerar a informação de um determinado evento. Usando *macros* em detrimento das funções de depuração de erros pode-se poupar tempo no desenvolvimento da aplicação.

A macro *ASSERT()* recebe um booleano como argumento. Se o resultado é *FALSE*, imprime a informação que segue o padrão da *macro*. Por exemplo, para a *macro*:

62b *(Exemplo: Macro ASSERT() 62b)*≡
 ASSERT(x!=0) //linha 30 do ficheiro teste.cpp;

O resultado, para o caso de *x=0*, é "*ASSERT: x == 0 teste.cpp (30)*".

A macro *CHECK_PTR()* verifica se o programa vai tentar aceder a uma posição de memória que não lhe foi atribuída, isto é, se excedeu a memória atribuída ou se ocorreu um erro na criação de algum objecto. A *CHECK_PTR()* recebe um apontador como argumento, e se o apontador for nulo, chama a função *qFatal* para informar que a aplicação tentou aceder a uma posição de memória que não lhe foi atribuída.

62c *(Exemplo: Macro CHECK_PTR() 62c)*≡
 char *string = new char[100];
 CHECK_PTR(string);

Neste exemplo, *CHECK_PTR()* invocará função *qFatal()* se a variável *string* não for criada correctamente.

Ainda que estes mecanismos não tenham sido utilizados, a aplicação testa as mesmas condições mas por mecanismos próprios. No entanto, a utilização quer das funções, quer das *macros* disponibilizadas pelo *Qt* permitiria um desenvolvimento mais rápido, e uma melhor compreensão do código por outro programador. Por este motivo, não só se deve passar a utilizar estes mecanismos, como alterar os que existem no código desenvolvido.

6.5 Testes Unitários

É também importante definir e implementar teste unitários a serem desenvolvidos através da ferramenta *QtUnit*. O *QtUnit* é uma tecnologia para desenvolvimento de testes unitários em C++ que foi baseada no *CppUnit*. Todo o código do *CppUnit* foi re-implementado para utilizar exclusivamente as bibliotecas *Qt 3.x* [Need 2002]. Assim o *QtUnit* tornou-se tão portátil como qualquer outra aplicação desenvolvida sobre *Qt*. O *QtUnit* encontra-se organizado em três partes distintas:

- a ferramenta de teste; onde são descritos, organizados e executados os casos de teste;
- a ferramenta de execução em modo texto; onde são executados e imprimidos os resultados de avaliação dos testes;
- a ferramenta de execução em ambiente gráfico; onde, através de uma interface gráfica, são executados e apresentados os resultados dos testes.

É de salientar que a versão 4.0 do *Qt* já possui uma solução para realização de testes unitários. A biblioteca *qtestlib* permite o desenvolvimento de teste unitários em C++ para aplicações *Qt* [Trolltech 2005b]. No caso de se portar o código para *Qt4* não será necessário utilizar o *QtUnit*. Assim, poder-se-á manter o mesmo ambiente de desenvolvimento para a aplicação e para os teste unitários.

6.6 Relatórios

O primeiro relatório tem como objectivo apresentar a evolução das despesas e PEDIDOS DE PAGAMENTO de um sub-projecto. Para tal é criada uma tabela de entrada dupla *Parceiros X Pedidos de Pagamento*. Para cada PEDIDO DE PAGAMENTO existirão duas linhas na tabela. Uma para a despesa executada nesse PEDIDO DE PAGAMENTO e outra com a verba a ser reembolsada devido a essa despesa. O valor da despesa executada obtém-se pelo somatório do valor de cada PEDIDO DE REEMBOLSO aprovado referente ao PEDIDO DE PAGAMENTO em questão. O valor do reembolso é calculado usando o:

1. o valor da participação da entidade financiadora:

$$E_e * CPOS_C \quad (6.1)$$

2. o valor da participação dos outros Parceiros na despesa de um Parceiro:

$$E_e * CP_p + (0.25\% - CP_p) \quad (6.2)$$

3. o valor da participação de um Parceiro nas despesas dos outros Parceiros do projecto:

$$\sum_{e=1, p=1}^n E_e * P_p * CP_p \quad (6.3)$$

Obtidos estes três valores basta somar o valor da Equação 6.1 ao valor da Equação 6.2 e retirar o valor da Equação 6.3. Matematicamente, temos:

$$(E_e * CPOS_C) + (E_e * CP_p + (0.25\% - CP_p)) - \sum_{e=1, p=1}^n E_e * P_p * CP_p \quad (6.4)$$

Em que E_e é o valor de despesa apresentada pelo parceiro no PEDIDO DE PAGAMENTO e, $CPOS_C$ é o valor da comparticipação da entidade financiadora. Este valor é 71.25% porque é necessário subtrair os 3.75% retidos pelo POS_C até à conclusão do projecto. A variável CP_p é a comparticipação do Parceiro p .

Existe ainda mais uma coluna e duas linhas para além das já referidas. A coluna POS_C indica o valor a reembolsar pela entidade financiadora e é o somatório dos reembolsos de todos os Parceiros desse sub-projecto. A primeira e última linha indicam a comparticipação e reembolso final de cada Parceiro, respectivamente. O somatório da participação de cada Parceiro no projecto tem de ser 25%, sendo os outros 75% reembolsados pelo POS_C. O valor de reembolso no final diz respeito a uma retenção de 5% feita pelo POS_C sobre a verba que está a financiar. Esta retenção será reembolsada no final do projecto quando for entregue e aprovado o relatório final do projecto Porto Digital. Estes 5% retidos sobre 75% da despesa, corresponde a 3.75% do valor elegível.

Na Tabela 6.1 apresenta-se um exemplo de uma saída deste relatório.

<i>Despesa Efectuada = 20000 euros</i>	<i>Parceiro 1</i>	<i>Parceiro 2</i>	<i>Parceiro 3</i>	<i>POS_C</i>
Comparticipação	12%	10%	3%	
Execução 1	-4000.00	-5800.00	-1200.00	-7837.50
Reembolso 1	2530.00	4482.50	825.00	
Execução 2	-1200.00	-3000.00	-2500.00	-4773.75
Reembolso 2	351.00	2217.50	2205.25	
Execução 3	-300.00	-800.00	-300.00	-997.50
Reembolso 3	120.75	630.00	246.75	
Execução 4	-500.00	-400.00	0.00	-641.25
Reembolso 4	373.25	295.00	-27.00	
Reembolso Final (3.75%)	225.00	375.00	150.00	-750.00

Tabela 6.1: Exemplo de um Relatório para um Sub-projecto

A primeira célula da tabela tem o somatório das despesas de todos os Parceiros. O valor negativo do reembolso da Execução 4, Parceiro 3, deve-se ao facto de o Parceiro 3 não ter apresentado despesas e ter de pagar o reembolso das despesas realizadas pelos outros Parceiros.

A este relatório poder-se-ão adicionar ALERTAS de gestão de projectos. Por exemplo, é possível que a aplicação, dado o orçamento e a data prevista para conclusão do projecto, indique que:

- o desenvolvimento se encontra atrasado;
- o orçamento previsto está perto de ser atingido e, se não for perto do final da data de conclusão, que poderá indicar desvios no orçamento;
- um determinado Parceiro já excedeu ou está próximo de exceder o limite orçamentado para as despesas naquele sub-projecto.

O outro relatório já definido tem como objectivo prever o valor que será reembolsado a cada parceiro por cada PEDIDO DE PAGAMENTO.

Na Tabela 6.2 encontra-se um exemplo do extracto provisório de um Parceiro que participa em dois sub-projectos. O Parceiro 1 participa nos sub-projectos “INF-01” e “ACE-05”, tendo uma taxa de comparticipação de 12% e 18%, respectivamente. No sub-projecto

“INF-01” não apresentou despesas mas os restantes Parceiros sim. No sub-projecto “ACE-05” teve uma execução financeira de 25228.09 Euros. A comparticipação de 2443.08 Euros resulta da multiplicação do orçamento executado pelos restantes Parceiros (20275.66) pela taxa de participação do Parceiro (12%). No âmbito do sub-projecto “ACE-05” tem a receber 19373.19 Euros, dos quais 11333.85 serão reembolsados pelo FEDER, 6641.17 pelo Orçamento de Estado, 452.12 pelos restantes parceiros, e 946.05 será reembolsado pelo POS.C somente após a conclusão do projecto.

Este relatório, apenas disponível para a Associação Porto Digital, deve ser gerado sempre que seja enviado uma FOLHA DE PEDIDOS DE PAGAMENTO da APD ao POS.C e posteriormente, enviado ao Parceiro em questão.

6.7 Resumo

Em termos de possíveis melhorias destacam-se melhorias:

- na gestão de memória - com uma a utilização da *API SAX* em detrimento da *API DOM*. No entanto, a utilização da *API SAX* implicará um maior número de leitura de ficheiros. Por este motivo é necessário avaliar o impacto desta alteração no desempenho global da aplicação;
- nos mecanismos de segurança - com a implementação de um mecanismo de protecção dos dados durante a comunicação entre os intervenientes no projecto Porto Digital;
- nos mecanismos de depuração de erros - com a utilização das funções de depuração de erros e *macros* disponibilizadas pelo *Qt*;
- no conjunto de relatórios disponibilizados aos vários actores da aplicação - com a implementação dos relatórios definidos na secção 6.6.

Antes do lançamento da aplicação é também importante definir e aplicar testes unitários. Para tal poder-se-á utilizar a aplicar *QtUnit* para testes em aplicações desenvolvidas em *Qt 3.x* ou a biblioteca *QtTest* disponibilizada pela *Qt* a partir da versão *Qt4.0* [Trolltech 2005a].

<i>Parceiro 1</i>	<i>20/08/2005</i>	<i>INF-01</i>	<i>ACE-05</i>	<i>Total</i>
Acumulado Anterior	Quantidade de Pedidos de Pagamento	0	0	
	Orçamento Global Executado	0.00	0.00	
	Orçamento Executado pelo Parceiro	0.00	0.00	
	Comparticipação % do Parceiro	12%	18%	
	Conta Corrente do Parceiro	0.00	0.00	
	Reembolso pelo FEDER	0.00	0.00	
	Reembolso pelo OE	0.00	0.00	
	Conta Corrente na APD	0.00	0.00	
	Retido até Conclusão do Projecto	0.00	0.00	
Pedido de Pagamento	Nº do Pedido de Pagamento ao POS_C	1	1	
	Orçamento Global Executado	20275.66	32527.24	
	Orçamento Executado pelo Parceiro	0.00	25228.09	
	Comparticipação % do Parceiro	12%	18%	
	Comparticipação do Parceiro	2433.08	5854.90	
	Comparticipação FEDER	0.00	11333.85	
	Comparticipação OE	0.00	6641.17	
	Comparticipação dos Restantes Parceiros	-2443.08	452.12	
	Retido até Conclusão do Projecto	0.00	946.05	
Reembolso	POS_C FEDER	0.00	11333.85	
	POS_C OE	0.00	0.00	
	Restantes Parceiros	0.00	452.12	
Transferência	Associação Porto Digital	-2433.08	11785.96	9352.89
Estado Actual	Quantidade de Pedidos de Pagamento	1	1	
	Orçamento Global Executado	20275.66	32527.24	
	Orçamento Executado pelo Parceiro	0.00	25228.09	
	Comparticipação % do Parceiro	12%	18%	
	Conta Corrente do Parceiro	0.00	-7587.22	
	Reemboolso FEDER	0.00	0.00	
	Reembolso OE	0.00	6641.17	
	Conta Corrente APD	0.00	0.00	
	Retido até Conclusão do Projecto	0.00	946.05	

Tabela 6.2: Exemplo de um Relatório da Conta Corrente do *Parceiro 1*

Capítulo 7

Conclusões e Trabalho Futuro

O projecto de estágio na ParadigmaXis teve como pano de fundo o projecto Porto Digital. A fase inicial do estágio foi dedicada ao estudo da documentação desenvolvida para a candidatura do projecto Porto Digital ao programa governamental Portugal Digital. Como se verificaram alguns atrasos no arranque do projecto Porto Digital, durante esta fase, foi-me atribuída a responsabilidade de actualizar o sítio *Web* do *CityHelp*. Para esta fase foi levado a cabo um trabalho de recolha de informação sobre as tecnologias utilizadas na implementação do sítio *Web*: *MySQL*, *PHP* e *Javascript*. As alterações solicitadas foram todas realizadas e testadas em condições de produção, tendo sido cumpridos todos os objectivos desta fase.

Concluída esta primeira fase, foram especificados os requisitos da aplicação a desenvolver no âmbito do estágio, bem como a arquitectura e tecnologias a utilizar na sua implementação. De entre as tecnologias utilizadas são de referir o *XML*, as linguagens de programação *C/C++* e *Qt*, a *API DOM* e a ferramenta *QtDesigner*. A aprendizagem destas tecnologias foi muito aliciante e esta foi a fase mais interessante do estágio pois abarcou várias áreas da informática. Foram feitas pesquisas sobre soluções de persistência, encriptação e transmissão de dados; sobre processos de desenvolvimento de *software*, especificação de requisitos e aplicações para definição de testes unitários.

Quanto à aplicação desenvolvida, esta ainda não se encontra numa fase em que seja possível testá-la no cliente. Para tal é necessário terminar a implementação de alguns requisitos e, sobretudo, terminar alguns mecanismos de segurança que são críticos.

Entre os requisitos destaca-se a necessidade de:

1. implementar a geração dos relatórios definidos na secção 6.6;
2. permitir a um utilizador criar outro utilizador. Actualmente só é possível “adicionar” um utilizador trocando o ficheiro que contém os dados sobre os utilizadores;
3. implementar mais verificações, quer verificações “*duras*” quer verificações “*suaves*”.

Os mecanismos de segurança, para os quais é necessário terminar o desenvolvimento, encontram-se descritos na secção 6.3.

Também é necessário realizar adaptações à aplicação desenvolvida para possibilitar a sua instalação na Associação Porto Digital e POS_C já que o desenvolvimento efectuado até este momento segue a perspectiva do Parceiro. Isto é, a aplicação está preparada para ler *XML* dos Parceiros. A estrutura do *XML* da Associação Porto Digital é basicamente a mesma que a do Parceiro, mas na APD, quando a aplicação recebe o *DOM* de sincronização, é adicionado o código do Parceiro como atributo dos nós *<fornecedores>* e *<pedidos>*.

Toda a aplicação da APD deve trabalhar com este atributo permitindo consultas não só por projecto mas também por Parceiro.

Também relativo ao processo de sincronização, é necessário implementar o sistema de criação de ALERTAS por parte da Associação Porto Digital. Ou seja, tem de haver a possibilidade do utilizador da aplicação em funcionamento na APD criar ALERTAS. Para isso, depois da APD receber os dados do Parceiro, a aplicação deverá apresentá-los ao utilizador e permitir que este os aprove ou altere. Nas alterações sobre um FORNECEDOR, a operação de alteração consiste em indicar o código de um FORNECEDOR já aprovado e por isso, no repositório da APD. Por exemplo, um parceiro envia um FORNECEDOR com o nome *PT Comunicações* e com um determinado *NIF*. A aplicação deve indicar que já existe um FORNECEDOR com esse *NIF* mas nome diferente. Então o utilizador indica que o FORNECEDOR enviado pelo Parceiro deve ser alterado pelo que já se encontra na base de dados da APD. Neste momento é gerado um ALERTA para o Parceiro com os dados do FORNECEDOR da APD. Em nenhuma situação, o utilizador pode alterar manualmente os dados de um FORNECEDOR pois poderia vir a causar a introdução de erros nas próprias alterações. A aplicação também deverá aprovar automaticamente um FORNECEDOR recebido por um Parceiro em que todos os campos coincidam com um cliente já existente nos ficheiros. Quando a aplicação não aprovar automaticamente um FORNECEDOR recebido do Parceiro, ou não conseguir identificar um FORNECEDOR no Repositório de Dados da APD com o qual consiga estabelecer um padrão de identidade, então a decisão será tomada exclusivamente pelo utilizador. Este deverá aprovar o FORNECEDOR criando um novo elemento na lista de FORNECEDORES ou considerar que o FORNECEDOR recebido se trata de outro já existente no Repositório de Dados e marcar essa relação. Consoante as operações efectuadas é gerado o *DOM* de ALERTAS a enviar para o Parceiro. No repositório de FORNECEDORES na APD, um nó `<fornecedor listaDeAtributos>` terá de ter uma lista de filhos que indique quais os Parceiros que têm nos seus repositórios aquele FORNECEDOR. Esta lista permitirá que na necessidade de se alterar os dados de um FORNECEDOR aprovado, se possa gerar automaticamente um conjunto de ALERTAS que repliquem essa alteração para os Parceiros.

Para análise da fiabilidade dos dados apresentados por cada Parceiro pode-se, numa fase mais adiantada do projecto, implementar um mecanismo que monitorize o número de alterações feitas pela APD aos dados desse Parceiro.

O desenvolvimento verificado até à data demonstra que, quer as tecnologias escolhidas, quer a arquitectura definida, são adequadas e respondem às necessidades do problema proposto. Neste momento, apenas é necessário um maior esforço na implementação para se obter uma aplicação que satisfaça todos os requisitos do cliente. Ou seja, a aplicação desenvolvida pode ser considerada como um protótipo que valida as escolhas tecnológicas e a solução seguida.

Nesta fase de iniciação ao mundo de trabalho, foi sentida diferença quanto aos métodos de trabalhos praticados durante a fase anterior da licenciatura. Ao contrário do que acontece durante um semestre curricular, em que existem vários pontos de avaliação, durante o estágio poucas vezes foi sentida a pressão de ter de terminar uma actividade, relatório ou apresentação. O trabalho é mais contínuo e sistemático. Na faculdade, como se lida com vários projectos simultaneamente, e cada um com vários pontos de avaliação ao longo do semestre, existe um maior retorno da evolução do trabalho.

De salientar que para a avaliação do sucesso de um trabalho curricular desta natureza devem ser também valorados aspectos relacionais, como por exemplo, a evolução do estagiário ao longo do tempo e a sua adaptação ao mercado de trabalho. Também nesta

perspectiva, o estágio na ParadigmaXis só pode ser avaliado positivamente.

Outro aspecto positivo neste estágio foi o convívio com profissionais experientes e altamente qualificados que incentivavam a pesquisa e descoberta de novas tecnologias e soluções informáticas.

Apesar de algumas dificuldades, naturais para uma primeira experiência laboral, o estágio decorreu sem grandes percalços tendo sido atingidos os principais objectivos.

Bibliografia

- Associação Porto Digital (2004). Porto Digital — Candidatura ao Programa Portugal Digital. Relatório técnico.
- Barton, John J. e Lee R. Nackman (1994). *Scientific and Engineering C++*. Addison-Wesley.
- Beck, Kent (2000). *Extreme Programming Explained*. Addison-Wesley.
- Biham, E. e A. Shamir (1991). *Differential Cryptanalysis of DES — Like Cryptosystems*. Journal of Cryptology.
- Biham, E. e A. Shamir (1993). *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag.
- Bishop, Matt (2002). *Computer Security: Art and Science*. Addison Wesley.
- Bradley, Neil (2002). *The XML Companion*. Addison-Wesley.
- Bray, Tim, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, e John Cowan (2004). XML1.1. Obtido de <http://www.w3.org/TR/xml11/> em Abril de 2005.
- Brownell, David (2002). SAX. Obtido de <http://www.saxproject.org/> em Abril de 2005.
- Connolly, Dan (Ed.) (1997). *XML Principles, Tools, and Techniques*. O'Reilly & Associates, Inc.
- Dalheimer, Matthias Kalle (1999, Maio). *Programming with Qt*. O'Reilly & Associates, Inc.
- Daza, Guillermo Prieto (2005). Metodología para el Desarrollo de Aplicaciones en la plataforma: Linux PDA. Relatório técnico. Obtido de <http://es.tldp.org/almacen/Manuales-LuCAS/doc-tesis-desarrollo-linux-PDAs/tesis-navegable/thesis.html> em Março de 2005.
- de Jesús Angel Angel, José (2005). AES — Advanced Encryption Standard. Relatório técnico. Obtido de http://computacion.cs.cinvestav.mx/~jjangel/aes/AES_v2005_jjaa.pdf em Julho de 2005.
- Deitel, H. M. e P. J. Deitel (2001). *C++ How to Program*. Prentice Hall.
- Fluhrer, Scott, Itsik Mantin, e Adi Shamir (2001). Weaknesses in the Key Scheduling Algorithm of RC4. Relatório técnico. Obtido de http://www.crypto.com/papers/others/rc4_ksaproc.ps em Julho de 2005.

- Gamma, Erich, Richard Helm, Ralph Johnson, e John Vlissides (1995). *Design Patterns — Elements of reusable object-oriented software*. Addison-Wesley.
- Gilmore, J. (1998). *Cracking the DES*. O'Reilly.
- Jacobson, Ivar, Grady Booch, e James Rumbaugh (1999). *The Unified Software Development Process*. Addison-Wesley.
- KDE (2005). History: The Qt Issue. Obtido de <http://www.kde.org/whatiskde/qt.php> em Abril de 2005.
- Kelsey, John (2004). NIST and the AES in 2004. Relatório técnico, NIST. Obtido de http://www.aes4.org/english/events/aes4/downloads/kelsey_aes4.pdf em Julho de 2005.
- Kelsey, John, Bruce Schneier, e David Wagner (1999). Key-Schedule Cryptanalysis of 3-Way, IDEA, G-DES, RC4, SAFER, and Triple-DES. Relatório técnico. Obtido de <http://www.schneier.com/paper-key-schedule.pdf> em Julho de 2005.
- Marchal, Benoît (2001, Agosto). SAX, The Power API. Relatório técnico, ibm.com/developerworks. Obtido de <http://www-106.ibm.com/developerworks/xml/library/x-saxapi/?dwzone=xml> em Abril de 2005.
- Medina, Manel e Abraham Pasamar (2004). Ruptura del SHA-1? Los de firme-e podrían comprometerse en breve. Relatório técnico.
- MySQL (2001). MySQL Reference Manual: 1.6.1.1 Features Available in MySQL 4.0. Obtido de <http://dev.mysql.com/doc/mysql/en/nutshell-4-0-features.html> em Agosto de 2005.
- MySQL (2004a). Chapter5: Joins, Temporary Tables, and Transaction. Obtido de <http://dev.mysql.com/tech-resources/articles/mysql-db-design-ch5.html> em Agosto de 2005.
- MySQL (2004b). MySQL 5.0 Stored Procedures. Obtido de <http://dev.mysql.com/tech-resources/articles/mysql-storedprocedures.pdf> em Agosto de 2005.
- Need, Use What You (2002). QtUnit — Software development testing framework. Obtido de <http://www.uwyn.com/projects/qtunit/> em Agosto de 2005.
- Pereira, Alexandre e Carlos Poupa (2004). *Linguagens Web*. Edições-Sílabo.
- Project, OpenSSL (2005). EVP_DigestInit(3). Obtido de http://www.openssl.org/docs/crypto/EVP_DigestInit.html# em Julho de 2005.
- Ramalho, José C. e Pedro Henriques (2002). *XML and XSL — Da teoria à prática*. FCA — Editora Informática.
- Ramsey, Norman (1994, Setembro). Literate Programming Simplified. *IEEE Software* 11(5), 97–105. <http://www.eecs.harvard.edu/~nr/noweb>.
- Rumbaugh, James, Ivar Jacobson, e Grady Booch (1999). *The Unified Modeling Language Reference Manual*. Addison-Wesley / ACM Press.

- Russel, Chad e Jon Stephens (2004). *Beginnig MySQL Database Design and Optimization: From Novice to Professional*. Capítulo 5: Joins, Temporary Tables, and Transaction legalmente disponível em <http://dev.mysql.com/tech-resources/articles/mysql-db-design-ch5.pdf> em Setembro de 2005.
- Schneier, Bruce (1994). *Applied Cryptography - Protocols, algorithms and source code in C*. John Wiley & Sons.
- Schneier, Bruce (1995, Outubro). Factoring in Public-Key's Future. *Byte* 20(10), 78-78.
- Schneier, Bruce (2004). Saluting The Data Encryption Legacy. Obtido de <http://www.schneier.com/essay-064.html>, em Julho de 2005.
- Schneier, Bruce (2005). The Blowfish Encryption Algorithm. Obtido de <http://www.schneier.com/blowfish.html> em Julho de 2005.
- Serrão, Carlos e Joaquim Marques (2001). *Programação com PHP4*. FCA — Editora de Informática.
- Silva, Alberto Manuel Rodrigues e Carlos Alberto Escaleira Videira (2001). *UML, Metodologias e Ferramentas CASE*. Centro Atlântico PT.
- Solin, Daniel (2000). *Qt Programming*. SAMS.
- Sommerville, Ian (1995). *Software Engineering*. Addison-Wesley.
- Stroustrup, Bjarne (1997). *The C++ Programming Language*. Addison-Wesley.
- TrollTech (2005). <http://www.trolltech.com/products/qt/designer.htm>.
- Trolltech (2005a). Qt Solutions Catalog — Components for Qt version 4. Obtido de <http://www.trolltech.com/products/solutions/catalog/4/index.html> em Agosto de 2005.
- Trolltech (2005b). Unit Test Library. Obtido de <http://www.trolltech.com/products/solutions/catalog/4/Tools/qttestlib/> em Agosto de 2005.
- Vakali, Athena e Barbara Catania Maddalena (2005). XML Data Stores: Emerging Practices. Relatório técnico. Obtido de <http://paginas.fe.up.pt/~jlopes/teach/2005-06/DAPI/iee-native-xml.pdf> em Setembro de 2005.
- W3C (2005). W3C XML Activities. Obtido de <http://w3schools.com/xml/default.asp> em Abril de 2005.
- Wang, Xiaoyun, Dengguo Feng, Xuejia, e Hongbo Yu (2004). Collisions for Hash Functions. Relatório técnico. Obtido de <http://eprint.iacr.org/2004/199.pdf> em Julho de 2005.
- Wood, Lauren et al. (1998, Outubro). Document Object Model (DOM) Level 1 Specification. Relatório técnico, W3C. <http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/>.
- Yarger, Randy Jay, George Reesse, e Tim King (1999). *MySQL and mSQL*. O'Reilly.

Glossário

Actor. É a idealização de uma entidade externa à aplicação — pessoa, processo ou coisa — que interage com o sistema, sub-sistema ou classe.

Alerta. Mensagem digital enviada da Associação Porto Digital para o Parceiro que contém a informação relativa ao conjunto de operações que o Parceiro deve proceder na sua base de dados. O sistema de recepção de ALERTAS foi implementado de maneira a que o Parceiro seja obrigado a executar estas alterações.

Analizador Gramatical. Aplicação que divide a informação em pedaços mais pequenos de acordo com um conjunto de regras, validando assim a estrutura e integridade do documento.

ANSI. Acrónimo para *American National Standards Institute*. A ANSI é uma organização americana, sem fins lucrativos, que tem por objectivo coordenar o desenvolvimento de normas em várias áreas da economia. Na área da informática são conhecidos inúmeros normas ANSI, entre eles o ANSI C que é a descrição de compiladores e programas em C.

API. Acrónimo para *Application Program Interface*. Especificação de um interface de um pacote de *software*, através do qual outros programas podem utilizar esse pacote.

CityHelp. O *CityHelp* é uma aplicação que funciona como roteiro digital. Permite procurar e localizar qualquer local ou ponto de interesse de uma cidade, disponibilizando uma enorme quantidade de informação descritiva e fotográfica de toda a cidade. O *CityHelp* foi desenvolvida pela InfoPortugal SA e a Ubicuos SA .

Case-sensitive. Sensível ao facto de as letras estarem em maiúsculas ou minúsculas.

Caso de Utilização. Descreve uma interacção coerente entre um actor e o sistema sem revelar a estrutura interna do sistema.

Combo box. Caixa de selecção.

CppUnit. Ferramenta para testes unitários em C++. Criado por Michael Feathers, o *CppUnit* surgiu como o resultado da re-implementação do *JUnit* para C++.

Cron. Pequeno módulo de Linux que permite executar comandos a um determinado momento ou entre intervalos de tempo definido. Em Windows esta ferramenta chama-se agenda de tarefas (*Scheduled Tasks*).

CSS. Acrónimo para *Cascading Style Sheet*. Um documento CSS, ou folha de estilo, tem como objectivo especificar a formatação de elementos HTML ou XML.

Diagrama de Casos de Utilização. Captura os requisitos de comportamento do sistema, sub-sistema ou classe do ponto de vista do utilizador, particionando funcionalmente o sistema em transacções significantes para o actor.

Diffie-Helman. Primeiro algoritmo de chave pública. Usado para distribuição de chaves, não pode ser utilizado para encriptação e descriptação de mensagens. A sua segurança é baseada na dificuldade de calcular logaritmos discretos finitos, comparativamente à facilidade de calcular exponenciais finitos.

Documento. Objecto da aplicação que qualifica o comprovativo da realização ou quitação de uma despesa efectuada pelo Parceiro.

Documento XML. Um documento *XML* é uma estrutura lógica com uma determinada hierarquia, identificando-se um elemento especial: a raiz. Um documento *XML* não tem de corresponder a uma entidade física, ficheiro ou conjunto de folhas impressas.

DOM de um Navegador. Acrónimo para *Document Object Model*. Especificação do modo como os objectos de uma página *Web* são representados. O *DOM* também define os atributos que estão associados aos objectos e o modo como os atributos e objectos são manipulados.

DOM XML. Acrónimo para *Document Object Model*. API orientada a objectos, utilizada para o acesso tipo árvore a documentos *XML*.

DSA. Acrónimo para *Digital Signature Algorithm*. Proposto em 1991 pelo *NIST (National Institute of Standards and Technology)*, o *DSA* é uma variante dos algoritmos digitais *Schnorr* e *ElGamal*.

DTD. Acrónimo para *Document Type Declaration*. O *DTD* é um conjunto de declarações que definem a estrutura de um tipo de documento, estando também especificados o domínio dos atributos, valores por omissão, estrutura de conteúdo de cada elemento, etc.

ECMA. Acrónimo para *European Computer Manufacturers Association*. Associação fundada em 1961 dedicada à padronização de sistemas de informação. Desde 1994 passou a denominar-se *ECMA Internacional* para reflectir as suas actividades internacionais.

Elemento. Bloco de informação caracterizado por uma etiqueta, que pode ser vazio ou englobar outros elementos.

ElGamal. Algoritmo de encriptação de chave pública. Pode ser utilizado para assinatura digital e para cifrar/decifrar mensagens. A sua segurança é baseada na dificuldade de calcular logaritmos discretos.

Etiqueta. Define um nó de *XML* caracterizando-o com meta-informação. São delimitadores de blocos de informação, tipicamente, utilizadas em linguagens de marcação para estruturar os dados.

ETSI. Acrónimo para *European Telecommunications Standards Institute*. Organização não governamental, sem fins lucrativos, que tem como missão produzir normas na área das telecomunicações.

code that needs work. The point of doing so is that a grep or a similar pattern-matching tool can find all such places quickly. This is common in GNU code.

Flash. Formato de animações vectoriais da Macromedia.

Folha de Pedidos de Pagamento. Conjunto de PEDIDOS DE PAGAMENTO enviados pela Associação Porto Digital para o Programa Operacional da Sociedade do Conhecimento.

Fornecedor. Objecto da aplicação que representa uma entidade prestadora de serviços ou productos a um Parceiro.

GenCode. Linguagem de marcação desenvolvida pela *GCA (Graphic Communications Association)* para desenvolver anotações para os documentos dos seus clientes. Esta linguagem permitiu a *GCA* integrar os vários documentos provenientes desses cliente.

GML. Acrónimo para *Generalized Markup Language*. Linguagem desenvolvida no seio da IBM com o objectivo de resolver os seus problemas internos de publicação electrónica. O *GML* permitiu que a produção de um livro, relatório ou edição electrónica pudesse ser processada a partir do mesmo documento.

Gtk+. Ferramenta multi-plataforma para desenvolvimento de componentes de interface gráfica com o utilizador. Licenciado sob *GNU LGPL*, o *Gtk+* é um *software* livre e é parte integrante do projecto *GNU*.

Hardware. Componentes electrónicos físicos de um sistema computadorizado. Um componente de *hardware* recebe instruções de *software* e desempenha a tarefa de que foi incumbido por tais instruções. Assim, um sistema computadorizado funciona correctamente apenas se o *hardware* e *software* forem convenientemente conjugados.

Hashing. Função criptográfica que gera uma saída de tamanho fixo independentemente do tamanho da entrada.

HTML. Acrónimo para *HyperText Markup Language*. Linguagem de marcação usada para disponibilizar documentos na Internet. A entidade coordenadora do desenvolvimento desta linguagem é o *W3C*. Como linguagem de marcação que é, o *HTML* utiliza etiquetas para definir informação. Os navegadores *Web* interpretam essa informação e apresentam os vários componentes de uma dada página.

IBM P690. Supercomputador da marca IBM. Possui 32 processadores de 62 bits a 1.3Ghz e com 32GB de RAM. Este supercomputador é capaz de executar 8.9 triliões de operações por segundo.

Interface. Descreve o conjunto de potenciais pedidos a uma aplicação. Pode aparecer como um conjunto de comandos que invocam métodos da aplicação ou como um conjunto de componentes gráficos com os quais o utilizador interage directamente.

Internet. É uma rede de computadores global que comunicam entre si usando protocolos *TCP/IP*.

- ISO.** Acrónimo para *International Software Organization*. Fundada em 1946, a *ISO* é uma organização internacional formada pelas instituições nacionais de normas de mais de 75 países. A *ISO* tem por objectivo a aprovação de normas internacionais em todos os campos técnicos com excepção das áreas de Electricidade e Electrónica. A *ANSI* e o IPQ (Instituto Português de Qualidade) são associados *ISO*.
- Linguagem de marcação.** O conceito de linguagem de marcação surgiu na década de 60 com a ideia de que os documentos estruturados poderiam ser trocados e manipulados, se fossem editados num formato normalizado e estruturado.
- Linguagem procedimental.** Método de resolução de um problema computacional através da sua divisão em problemas mais pequenos (*top-down design*).
- Marcação.** Forma de definir informação recorrendo a etiquetas.
- MD5.** Algoritmo de *hashing* desenvolvido por Ronald Rivest no MIT. Utilizado para assinatura digital e autenticação em sistemas de informação. O algoritmo foi projectado para ser rápido, simples e seguro.
- Modelo de Objectos de Domínio.** Modelo *UML* estático porque não descreve o comportamento do sistema em função do tempo. Os principais constituintes deste modelo são objectos de domínio e as suas relações, ou seja, os elementos do Modelo de Objectos de Domínio são os conceitos que são significativos para a aplicação, incluindo conceitos do “mundo-real”, abstractos, de implementação ou da ciência da computação.
- multi-processamento.** Capacidade de um sistema de executar várias tarefas simultaneamente como se múltiplas instâncias da aplicação estivessem a correr ao mesmo tempo.
- Navegador.** Aplicação que permite a visualização e navegação através de conteúdos da Internet, tipicamente documentos *HTML* hospedados em servidores *Web*.
- Namespace(em XML).** Um *namespace* associa um conjunto de nomes a um *URI*, de forma a distinguir entre elementos com o mesmo nome.
- Nó.** No contexto de documentos *XML* tem o mesmo significado que *Elemento*.
- Objecto de Domínio.** Descrição de um conceito do domínio da aplicação da aplicação. Os objectos de domínio são o centro a partir do qual é desenhado o Modelo de Objectos do Domínio.
- openSSL.** *Software* livre baseado na biblioteca *SSLeay* desenvolvida por Eric A. Young and Tim J. Hudson. A *openSSL* é desenvolvida no seio do projecto *OpenSSL* no qual é também desenvolvida a ferramenta *TTL* (*Transport Layer Security*).
- Pacote.** É um conjunto de classes relacionadas.
- PDF.** Acrónimo para *Portable Document Format*. Formato de arquivo desenvolvido pela *Adobe Systems* para representar documentos de modo independente da aplicação, *hardware* e sistema operativo usados para os criar. O *PDF* é um formato aberto, pelo que qualquer pessoa pode escrever aplicações que leiam ou escrevam *PDFs*. Há

aplicações gratuitas para leitura de *PDFs* para Windows, Linux e MacOS, alguns distribuídos pela própria *Adobe*.

Pedido de Reembolso. Conjunto de informação que descreve e comprova a existência de uma despesa por parte do Parceiro no âmbito de um projecto do Porto Digital.

Pedido de Pagamento. Conjunto de PEDIDOS DE REEMBOLSO que o Parceiro envia à Associação Porto Digital em cada sincronização.

Personal Home Page Tools. Ferramenta para páginas pessoais. Linguagem que precede o *PHP* (Pré-processador de hipertexto) utilizada por Rasmus Lerdorf para obter informações sobre o número de visitas à sua página pessoal.

Texto Aberto. Dados que podem ser lidos e entendidos sem nenhuma modificação especial. Informação não encriptada.

Texto Cifrado. Texto encriptado.

Programação Orientada a Objectos. Nesta abordagem, o primeiro passo para a resolução de um problema computacional é a definição de classes. Estas classes são abstracções das entidades/objectos físicos.

OOP. Acrónimo para *Object-Oriented Programming*. O mesmo que Programação Orientada a Objectos.

Requisitos não funcionais. Os requisitos não funcionais especificam aspectos gerais do sistema como eficiência, fiabilidade, tempos de respostas e restrições, tais como dependência de plataformas, capacidade dos dispositivos de entrada e saída, facilidade de manutenção, escalabilidade e reutilização.

RIPEMD-160. Algoritmo para o cálculo de assinaturas digitais desenvolvido na Europa por Hans Dobbertin, Antoon Bosselars y Bart Preneel e publicado em 1996.

RSA. Acrónimo para *Rivest Shamir Aldeman*, iniciais dos sobrenomes dos seus inventores: Ron Rivest, Adi Shamir e Leonard Aldeman. Apresentado em 1978, a segurança do *RSA* é baseada na dificuldade de realizar a factorização de números grandes.

Rúbrica. Classificação da despesa segundo as rubricas do Plano Oficial de Contabilidade permitidas no projecto.

SAX. Acrónimo para *Simple API for XML*. API orientada a eventos, utilizada para o acesso sequencial a documentos *XML*.

S-Box. Acrónimo para *Substitution-Box*. Componente básico na criptografia simétrica. Nos algoritmos de encriptação de blocos, são usadas para ocultar a relação entre o texto aberto e o texto cifrado.

Scripting. Pequenos programas embebidos numa determinada linguagem para processamento de informação na interacção do sistema com o utilizador.

- SGML.** *Standard Generalized Markup Language*. O *SGML* surgiu da conjunção dos esforços paralelos da *GCA* (*Graphic Communications Association*) no desenvolvimento da *GenCode* e da *IBM* no desenvolvimento do *GML* (*Generalized Markup Language*). Lançada publicamente em 1986 como a norma *ISO 8870*, a *SGML*, foi desenhada com o objectivo de permitir a definição e utilização de formatos de documentos. É suficientemente formal para permitir validar os documentos, tem estrutura suficiente para permitir a especificação e o manuseamento de documentos complexos e é extensível de modo a suportar a gestão de grandes repositórios de informação [Ramalho e Henriques 2002].
- SHA.** Acrónimo *Secure Hash Algorithm*. Conjunto de algoritmos criptográficos, desenhados pela *NSA National Security Agency* e publicados pelo *NIST*. O primeiro membro desta família de algoritmos foi publicado em 1993 sobre o nome de *SHA*. Actualmente este algoritmo é conhecido por *SHA-0* para evitar confusões com os seus sucessores. O *SHA-1* saiu dois anos mais tarde.
- SHA2.** Algoritmo da família *SHA*. Resume-se ao *SHA-1* com mais bits.
- Socket.** Meio pelo qual dois processos comunicam entre si trocando um conjunto de dados de forma fiável, segura e ordenada. Os dois processos comunicantes tanto podem estar em máquinas distintas como na mesma máquina.
- Software.** Instruções computáveis por determinado *hardware* com a finalidade de levar a cabo uma determinada tarefa.
- Software Livre.** Refere-se a um programa cujo código fonte se encontra disponível ao público em geral, para uso ou alteração sem qualquer custo. Aplicações ou código livre são, tipicamente, desenvolvidos de forma colaborativa através de comunidades de programadores.
- SQL.** Acrónimo para *Structured Query Language*. Linguagem de base de dados relacionais em que facilmente se grava, altera e recupera informação. Desenvolvida pela *IBM* nos anos 70. Em 1996 a *ANSI* publicou a norma *SQL*.
- Teste Unitário.** Programa escrito para correr sobre um determinado procedimento ou classe. Tipicamente, envia uma mensagem fixa ao código sob análise e verifica se retorna o valor previsível.
- URI.** Acrónimo para *Uniform Resource Indicator*, antes *Universal Resource Indicator*. O *Uniform Resource Indicator* é um identificador único. Não é garantido, ao contrário de um *URL*, que seja uma localização.
- URL.** Acrónimo para *Uniform Resource Locator*, antes *Universal Resource Locator*. Identificador único que aponta a localização de um ficheiro ou recurso. Muitas vezes designado como endereço *Web*, a sua utilização não se confina à Internet.
- W3C.** Acrónimo para *World Wide Web Consortium*. Consórcio internacional fundado em 1994 por Tim Berners-Lee. Engloba mais de 120 organizações com o objectivo de desenvolver normas para a Internet.
- Web.** O mesmo que *World Wide Web*. Ver *WWW*.

Web-services. Permitem que aplicações de diferentes plataformas e sistemas operativos possam trocar informação entre si utilizando *XML* na definição das suas interfaces públicas.

WWW. Acrónimo para *World Wide Web*. Rede de computadores que fornece informação em forma de hipertexto (*HTML*), comunicando sobre o protocolo *HTTP*.

XML. Acrónimo para *Xntensible Markup Language*. Linguagem de marcação utilizada para estruturação de dados num determinado documento. Através do *XML* também é possível definir outras linguagens e por este motivo é considerada uma meta-linguagem.

XML bem formado. Documento *XML* que cumpre todas as regras de boa formação de um documento *XML*.

XML Schema. Com a mesma função de um *DTD*, o *XML Schema* escreve-se na sintaxe *XML* não sendo necessário ao utilizador *XML* aprender qualquer sintaxe nova. Além dos tipos de dados comuns à maioria de aplicações, o *XML Schema* também permite a definição de outro tipos de dados, a extensão e reutilização de estruturas e a utilização de *Namespaces*.

XML válido. Documento *XML* que além de cumprir os requisitos dos documentos *XML* bem formados, adicionalmente respeita as regras de estrutura definidas no *DTD* ou *XML Schema*.

Apêndice A

Listagem dos Ficheiros XSD Schema

Neste anexo é disponibilizado o conteúdo dos ficheiros *XSD Schema* que validam os documentos *XML* com que a aplicação está preparada para trabalhar. São também apresentados os diagramas gerados pela ferramenta *Spy* da Altavo.

A.1 XSD Schema: rubricas.xsd

De seguida é apresentado o ficheiro *XSD Schema* que define um documento *XML* válido de RÚBRICAS.

80

(*XSD: XSD para Documentos XML de Rúbricas* 80)≡

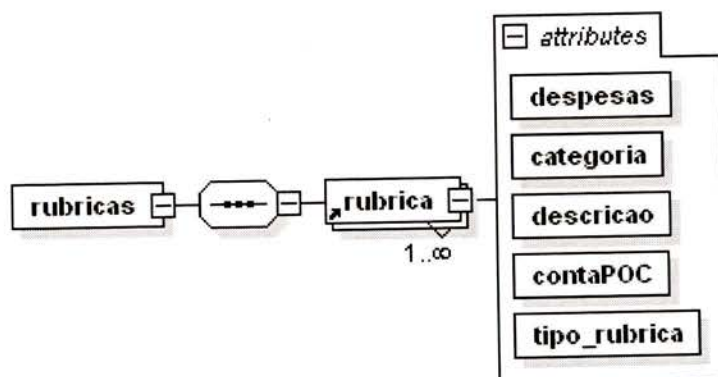
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2005 rel. 3 U (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="rubrica">
    <xs:complexType>
      <xs:attribute name="despesas" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="categoria" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="descricao" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="contaPOC" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:int"/>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="tipo_rubrica" use="required">
        <xs:simpleType>
```

```

        <xs:restriction base="xs:string"/>
    </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="rubricas">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="rubrica" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>

```

Na Figura A.1 pode consultar-se a estrutura de um documento *XML* válido de RÚBRICAS.



Generated with XMLSpy Schema Editor www.altova.com

Figura A.1: Diagrama da Estrutura de Documentos *XML* Válidos de RÚBRICAS

A.2 XSD Schema: documentos.xsd

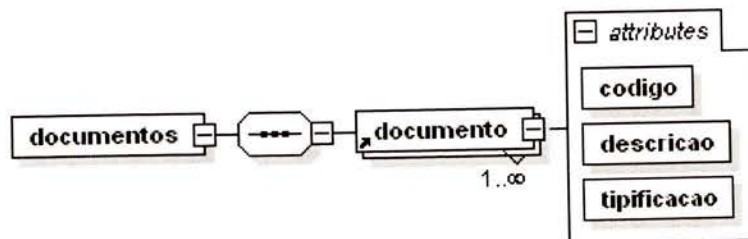
De seguida é apresentado o ficheiro *XSD Shema* que define um documento *XML* válido de *DOCUMENTOS*.

```

82  <XSD: XSD para Documentos XML de Documentos 82>≡
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by tiago (d) -->
    <!--W3C Schema generated by XMLSpy v2005 rel. 3 U (http://www.altova.com)-->
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
        <xs:element name="documento">
            <xs:complexType>
                <xs:attribute name="codigo" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="descricao" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                <xs:attribute name="tipificacao" type="xs:string" use="required"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="documentos">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="documento" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:schema>

```

Na Figura A.2 pode consultar-se a estrutura de um documento *XML* válido de *DOCUMENTOS*.



Generated with XMLSpy Schema Editor www.altova.com

Figura A.2: Diagrama da Estrutura de Documentos *XML* Válidos de *DOCUMENTOS*

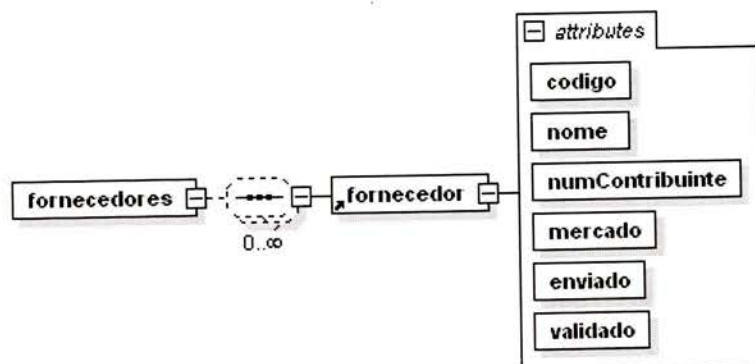
A.3 XSD Schema: fornecedor.xsd

De seguida é apresentado o ficheiro *XSD Shema* que define um documento XML válido de FORNECEDORES.

```
83  <XSD: XSD para Documentos XML de Fornecedores 83>≡
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <!--W3C Schema generated by XMLSpy v2005 rel. 3 U (http://www.altova.com)-->
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
        <xs:element name="fornecedor">
            <xs:complexType>
                <xs:attribute name="codigo" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:int"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="nome" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="numContribuinte" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:int"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="mercado" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="enviado" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="Sim"/>
                            <xs:enumeration value="Não"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="validado" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="Sim"/>
                            <xs:enumeration value="Não"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:complexType>
        </xs:element>
        <xs:element name="fornecedores">
            <xs:complexType>
                <xs:sequence minOccurs="0" maxOccurs="unbounded">
```

```
<xs:element ref="fornecedor"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Na Figura A.3 pode consultar-se a estrutura de um documento *XML* válido de FORNECEDORES.



Generated with XMLSpy Schema Editor www.altova.com

Figura A.3: Diagrama da Estrutura de Documentos *XML* Válidos de FORNECEDORES

A.4 XSD Schema: pedido.xsd

De seguida é apresentado o ficheiro *XSD Shema* que define um documento XML válido de PEDIDOS DE REEMBOLSO.

```

85  <XSD: XSD para Documentos XML de Pedidos de Reembolso 85>≡
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <!--W3C Schema generated by XMLSpy v2005 rel. 3 U (http://www.altova.com)-->
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
        <xs:element name="pedido">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="comprovativo" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="documento" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="numSequencia" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:int">
                            <xs:minInclusive value="1"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="contaPOC" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="sIVA" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:double">
                            <xs:minInclusive value="0"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="aprovado" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="Sim"/>
                            <xs:enumeration value="Não"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="numFornecedor" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="imputacao" use="required">

```

```

    <xs:simpleType>
      <xs:restriction base="xs:int">
        <xs:minInclusive value="0"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="data" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string"/>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="total" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:double">
        <xs:minExclusive value="2.5"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="enviado" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="Sim"/>
        <xs:enumeration value="Não"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="iva" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:double">
        <xs:minInclusive value="0"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="pedidos">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="projecto" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="projecto">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="pedido" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="codigo" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="INF01"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

```

```
<xs:enumeration value="INF02"/>
<xs:enumeration value="DIN03"/>
<xs:enumeration value="DIN04"/>
<xs:enumeration value="ACE05"/>
<xs:enumeration value="GOV06"/>
<xs:enumeration value="GOV07"/>
<xs:enumeration value="GOV08"/>
<xs:enumeration value="SEC09"/>
<xs:enumeration value="SEC10"/>
<xs:enumeration value="SEC11"/>
<xs:enumeration value="COO12"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="comprovativo">
  <xs:complexType>
    <xs:attribute name="endereco" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string"/>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="codigo" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:int"/>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="temp" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string"/>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
</xs:schema>
```


A.5 XSD Schema: alteracoesFor.xsd

De seguida é apresentado o ficheiro *XSD Shema* que define um documento *XML* válido de alterações não gravadas sobre os FORNECEDORES.

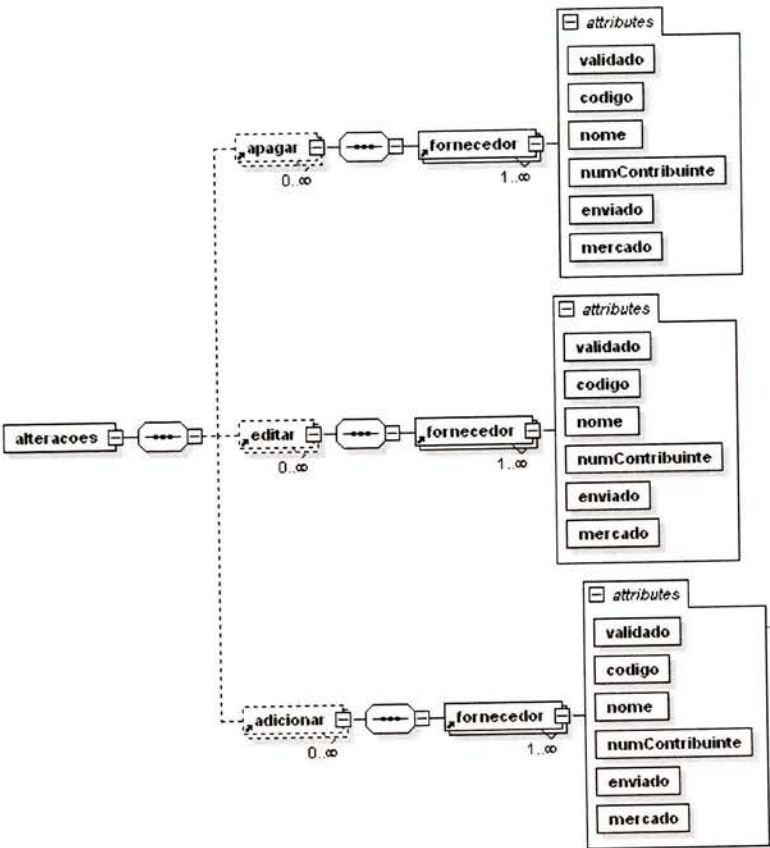
```

89  <XSD: XSD para Documentos XML de Alterações sobre Fornecedores 89>≡
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by tiago (d) -->
    <!--W3C Schema generated by XMLSpy v2005 rel. 3 U (http://www.altova.com)-->
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
        <xs:element name="adicionar">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="fornecedor" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="alteracoes">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="apagar" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="editar" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="adicionar" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="apagar">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="fornecedor" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="editar">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="fornecedor" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="fornecedor">
            <xs:complexType>
                <xs:attribute name="validado" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="Sim"/>
                            <xs:enumeration value="Não"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="codigo" use="required">
                    <xs:simpleType>

```

```
<xs:restriction base="xs:int">
  <xs:minExclusive value="0"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="nome" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="numContribuinte" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:int"/>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="enviado" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Sim"/>
      <xs:enumeration value="Não"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="mercado" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Nacional"/>
      <xs:enumeration value="Intracomunitário"/>
      <xs:enumeration value="Externo"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:schema>
```

Na Figura A.5 pode consultar-se a estrutura de um documento *XML* válido de alterações não gravadas sobre os FORNECEDORES.



Generated with XMLSpy Schema Editor www.altova.com

Figura A.5: Diagrama da Estrutura de Documentos *XML* Válidos de Alterações sobre FORNECEDORES

A.6 XSD Schema: alteracoesPP.xsd

De seguida é apresentado o ficheiro *XSD Shema* que define um documento *XML* válido de alterações sobre os PEDIDOS DE REEMBOLSO

```

92  <?XSD: XSD para Documentos XML de Alterações sobre Pedidos de Reembolso 92>≡
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <!--W3C Schema generated by XMLSpy v2005 rel. 3 U (http://www.altova.com)-->
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
        <xs:element name="alteracoesPP">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="projecto" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="pedido">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="comprovativo" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="numSequencia" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:int">
                            <xs:minExclusive value="0"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="documento" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="sIVA" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:int">
                            <xs:minExclusive value="0"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="descricao" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="aprovado" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="Sim"/>
                            <xs:enumeration value="Não"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:complexType>
        </xs:element>
    </xs:schema>

```

```

</xs:attribute>
<xs:attribute name="data" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="aaa" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="total" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:double">
      <xs:minInclusive value="2.5"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="enviado" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Sim"/>
      <xs:enumeration value="Não"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="iva" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:double">
      <xs:minExclusive value="0"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="numDocumento" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="projecto">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="pedido" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="codigo" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string"/>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>

```

```

</xs:element>
<xs:element name="comprovativo">
  <xs:complexType>
    <xs:attribute name="endereco" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string"/>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="codigo" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:int">
          <xs:minExclusive value="0"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="temp" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string"/>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
</xs:schema>

```

Na Figura A.6 pode consultar-se a estrutura de um documento *XML* válido de alterações não gravadas sobre os PEDIDOS DE REEMBOLSO.

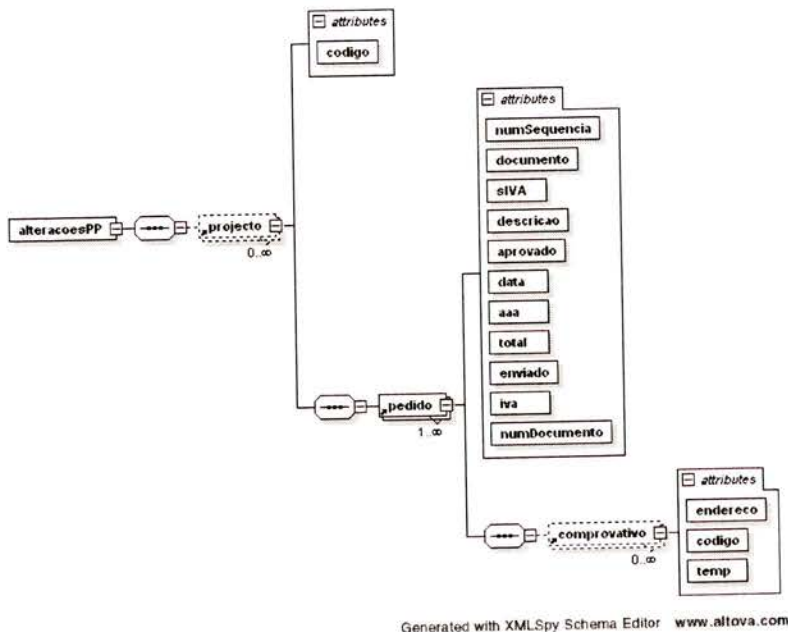


Figura A.6: Diagrama da Estrutura de Documentos *XML* Válidos de Alterações sobre PEDIDOS DE REEMBOLSO

Índices

I.8 Índice de Fragmentos de Código

- ⟨Exemplo: Código de Abertura de um Canal de Comunicação 53b⟩ [53b](#)
- ⟨Exemplo: Código de Actualização do Objecto de Visualização de Fornecedores 51⟩ [51](#)
- ⟨Exemplo: Código de Construção dos dados dos Fornecedores para Sincronização 53a⟩ [53a](#)
- ⟨Exemplo: Código de Geração de Chaves MD5 43⟩ [43](#)
- ⟨Exemplo: Código de Pesquisa de Fornecedores 49⟩ [49](#)
- ⟨Exemplo: Código de uma Mensagem Apresentada ao Utilizador 55⟩ [55](#)
- ⟨Exemplo: Código de Validação da Estrutura do DOM Alerta de Fornecedores 39⟩ [39](#)
- ⟨Exemplo: Documento XML bem Formado 15⟩ [15](#)
- ⟨Exemplo: DOM enviado à APD aquando da Sincronização 38⟩ [38](#)
- ⟨Exemplo: Elemento Folha Genérico dos Documentos XML 35a⟩ [35a](#)
- ⟨Exemplo: Elemento “operacao” de um DOM de Alterações 48⟩ [48](#)
- ⟨Exemplo: Elemento Rúbrica 35b⟩ [35b](#)
- ⟨Exemplo: Erro na Abertura de um Canal de Comunicação 54⟩ [54](#)
- ⟨Exemplo: Ficheiro XML de Pedidos de Reembolso 37⟩ [37](#)
- ⟨Exemplo: Ficheiro XML de Rúbricas 36⟩ [36](#)
- ⟨Exemplo: Função qDebug() 61⟩ [61](#)
- ⟨Exemplo: Função qWarning() 62a⟩ [62a](#)
- ⟨Exemplo: Macro ASSERT() 62b⟩ [62b](#)
- ⟨Exemplo: Macro CHECK_PTR() 62c⟩ [62c](#)
- ⟨XSD: XSD para Documentos XML de Alterações sobre Fornecedores 89⟩ [89](#)
- ⟨jXSD: XSD para Documentos XML de Alterações sobre Pedidos de Reembolso 92⟩ [92](#)
- ⟨XSD: XSD para Documentos XML de Documentos 82⟩ [82](#)
- ⟨XSD: XSD para Documentos XML de Fornecedores 83⟩ [83](#)
- ⟨XSD: XSD para Documentos XML de Pedidos de Reembolso 85⟩ [85](#)
- ⟨XSD: XSD para Documentos XML de Rúbricas 80⟩ [80](#)
- ⟨XSD: XSD para Documentos XML de Utilizadores da Aplicação 95⟩ [95](#)

I.9 Índice de Citações

Barton e Nackman [1994], 12
Beck [2000], 2
Biham e Shamir [1991], 40
Biham e Shamir [1993], 40
Bishop [2002], 40
Bradley [2002], 14
Bray et al. [2004], 16
Brownell [2002], 17
Connolly [1997], 17
Dalheimer [1999], 13
Daza [2005], 10
Deitel e Deitel [2001], 11
Fluhrer et al. [2001], 41
Gamma et al. [1995], 2
Gilmore [1998], 40
Jacobson et al. [1999], 22
KDE [2005], 12
Kelsey et al. [1999], 41
Kelsey [2004], 40
Marchal [2001], 17
Medina e Pasamar [2004], 45
MySQL [2001], 11
MySQL [2004a], 11
MySQL [2004b], 11
Need [2002], 63
Pereira e Poupa [2004], 9, 11
Project [2005], 42
Ramalho e Henriques [2002], 14–16, 35, 78
Ramsey [1994], 2
Rumbaugh et al. [1999], 20
Russel e Stephens [2004], 11
Schneier [1994], 44, 45, 61
Schneier [1995], 40, 41
Schneier [2004], 40
Schneier [2005], 41
Serrão e Marques [2001], 10
Silva e Videira [2001], 21
Solín [2000], 12, 13
Sommerville [1995], 5, 6
Stroustrup [1997], 11, 12
TrollTech [2005], 14
Trolltech [2005a], 65
Trolltech [2005b], 63
Vakali e Maddalena [2005], 16
W3C [2005], 16
Wang et al. [2004], 45
Wood et al. [1998], 17
Yarger et al. [1999], 10
de Jesús Angel Angel [2005], 40
Associação Porto Digital [2004], 3, 5

I.11 Índice Remissivo

- Actor, 20, 73
 - Associação Porto Digital, 24
 - Parceiro, 23
 - POS_C, 24
 - Utilizador Geral, 24
- Alerta, 73
- Analizador Gramatical, 73
- ANSI, 11, 73
- API, 73
- Case-sensitive, 73
 - pesquisa, 48
- Caso de Utilização, 20, 73
 - Diagrama, 22, 74
 - Módulo Consultas, 26
 - Módulo Gerir Fornecedores Porto Digital, 25
 - Módulo Gerir Fornecedores, 24
 - Módulo Gerir Pedidos de Pagamento, 25
 - Módulo Pedidos de Reembolso, 23
- CityHelp, 73
- Combo box, 73
- CppUnit, 73
- Criptografia de Chave Privada
 - S-Box, 77
- Criptografia de Chave Pública, 60
 - Diffie-Helman, 61, 74
 - DSA, 61, 74
 - ElGamal, 61, 74
 - RSA, 61, 77
- Criptografia de Chave Secreta, 40
 - AES, 40
 - Blowfish
 - Definição, 41
 - Escolha de Chaves, 41
 - Melhoramentos à Implementação, 60
 - DES, 40
 - RC4, 41
 - Triple DES, 40
- Cron, 73
- CSS, 73
- Documento, 74
- DOM, 74
 - Definição, 17
 - Estrutura implementada, 35, 48
 - Influência na Arquitectura, 30
- Validação, 39
- ECMA, 11, 74
- ETSI, 45, 74
- Figuras
 - A.5 Diagrama da Estrutura de Documentos *XML* Válidos de Alterações sobre FORNECEDORES, 91
 - A.6 Diagrama da Estrutura de Documentos *XML* Válidos de Alterações sobre PEDIDOS DE REEMBOLSO, 94
 - A.2 Diagrama da Estrutura de Documentos *XML* Válidos de DOCUMENTOS, 82
 - 5.5 Diagrama de Estados de um Filtro, 52
 - 5.8 Janela de FORNECEDORES, 58
 - A.3 Diagrama da Estrutura de Documentos *XML* Válidos de FORNECEDORES, 84
 - 4.1 Modelo da Arquitectura, 29
 - 4.2 Modelo de Camadas Lógicas, 30
 - 1.2 Modelo de Desenvolvimento em Cascata, 7
 - 3.2 Diagrama de Casos de Utilização, 22
 - 3.7 Diagrama de Casos de Utilização Disponíveis Através do Módulo Consultas, 26
 - 3.4 Diagrama de Casos de Utilização Disponíveis no Módulo Fornecedores, 24
 - 3.5 Diagrama de Casos de Utilização Disponíveis no Módulo Gerir Fornecedores Porto Digital, 25
 - 3.6 Diagrama de Casos de Utilização Disponíveis no Módulo Gerir Pedidos de Pagamento, 25
 - 3.3 Diagrama de Casos de Utilização Disponíveis no Módulo Pedidos de Reembolso, 23
 - 4.3 Diagrama de Componentes, 31
 - 4.4 Diagrama de Distribuição, 32
 - 3.1 Modelo de Objectos do Domínio, 21
 - 5.3 Formulário para Mudar *Palavra-Chave*, 44
 - 5.6 Navegador da Aplicação, 56

- A.4 Diagrama da Estrutura de Documentos *XML* Válidos de PEDIDOS DE REEMBOLSO, 88
- 5.2 Diagrama da Estrutura de Documentos *XML* Válidos de PEDIDOS DE REEMBOLSO, 37
- 5.4 Diagrama de Estados de uma Pesquisa, 49
- 1.1 Planeamento do Estágio, 6
- A.1 Diagrama da Estrutura de Documentos *XML* Válidos de RÚBRICAS, 81
- 5.1 Diagrama da Estrutura de Documentos *XML* Válidos de RÚBRICAS, 36
- 5.7 Barra de Ferramentas, 57
- A.7 Diagrama da Estrutura de Documentos *XML* Válidos de UTILIZADORES, 95
- Flash, 75
- Folha de Pedidos de Pagamento, 75
- Fornecedor, 75
- Funções Hash
 - Definição, 75
 - MD5, 76
 - Cripto-Análise, 45
 - Definição, 44
 - Implementação, 42
 - RIPEMD-160, 45, 77
 - SHA, 78
 - SHA2, 45, 78
- GenCode, 75
- GML, 75
- Gtk+, 75
- Hardware, 75
- HTML, 75
- IBM P690, 75
- Interface, 75
- Internet, 75
- ISO, 11, 76
- Linguagem de marcação, 76
- Linguagem procedimental, 76
- Marcação, 76
- Modelo de Objectos de Domínio, 76
- Modelo de Objectos do Domínio, 21
- multi-processamento, 76
- Namespace, 76
- Navegador, 76
 - DOM, 11, 74
- Nó, 76
- Objecto de Domínio, 76
- OOP, 77
- openSSL, 76
- Pacote, 76
- PDF, 76
- Pedido de Pagamento, 77
- Pedido de Reembolso, 77
- Personal Home Page Tools, 77
- Programação Orientada a Objectos, 77
- Requisitos Funcionais, 22
- Requisitos não funcionais, 26, 77
 - Desempenho, 26
 - Escalabilidade, 27
 - Fiabilidade, 26
 - Portabilidade, 26
 - Segurança, 27
 - Usabilidade, 27
- Rúbrica, 77
- SAX, 77
 - Definição, 17
- Scripting, 77
- SGML, 78
- Socket, 78
- Software, 78
- Software Livre, 78
- SQL, 78
- Teste Unitário, 78
- Texto Aberto, 40, 77
- Texto Cifrado, 40, 77
- URI, 78
- URL, 78
- W3C, 16, 78
- Web, 78
- Web-services, 79
- WWW, 79
- XML, 79
 - Descrição, 15
 - Documento, 14, 74
 - Bem formado, 14, 79
 - Válido, 15, 79
 - DTD, 74
 - Elemento, 14, 74
 - Etiqueta, 14, 74
 - Exemplo, 15
 - Solução de Armazenamento, 16
- XML Schema, 79

Colofon

↗ Este documento foi editado\
electronicamente usando GNU Emacs
21.3.1 em Windows XP Home Edition 2002
e GNU Emacs 21.3.1 em Linux Fedora Core 3.
As figuras foram obtidas através da captura dos
ecrãs ou desenhadas com o Microsoft Office Visio Pro-
fessional 2003. A ima- gem do planeamento
foi obtida através da ferramenta Mi-
crosoft Project Professional 2003.
Utilizou-se o Mi- crosoft Paint 5.1
para edição de algu- mas das imagens ob-
tidas imagens. Pos- teriormente, estas
imagens foram con- vertidas para En-
capsulated PostS- cript através da
impressão para fi- cheiro Postscript.
Usando dotNoWeb, uma ferramenta de
programação literária, o relatório foi escrito em
L^AT_EX2e e os fragmentos de código em Qt, C e
XML, sendo depois processado de modo a ex-
trair e compilar o código e criar a versão
final da documentação em PostScript,
↖ em PDF e em HTML. ↗





FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

BIBLIOTECA



0000081461