Mestrado em Engenharia Informática
Estágio
Relatório Final

Como criar uma empresa com apenas 1 euro

João Pedro Calixto Ribeiro Claro

jclaro@student.dei.uc.pt

Orientadores:

Pedro Furtado
Jorge Bernardino

Data: 02 de setembro de 2014

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Agradecimentos

Um agradecimento profundo aos meus orientadores, Professor Jorge Bernardino e Professor Pedro Furtado, pela sua orientação no desenvolvimento de competências e habilidades para o alcance de resultados e pela perseverança perante um orientando mais atípico.

À Dra. Márcia Espírito do Santo e aos meus colegas e amigos Sara João, Joana Oliveira e Daniel Barbosa pelo incessante e incondicional apoio e motivação, essenciais para a realização deste trabalho. Também um agradecimento aos meus colegas e amigos Pedro Neves, Ana Lopes, Beatriz Costa, João Ribeiro, João Marques, João Cerveira, entre tantas outras amizades, pelas suas palavras e conselhos durante o processo.

À minha irmã pela sua contribuição para este trabalho e aos meus pais, pela liberdade, apoio e paciência, três circunstâncias fundamentais das quais tive oportunidade de desfrutar durante a minha vida académica.

Resumo

O software *open-source* é uma área emergente que tem ganho maturidade e tem sido crescentemente adoptado pelas empresas como uma alternativa de menor custo e com maior potencial de adaptação de novas tecnologias. Oferece grandes oportunidades para redução de custos e melhoria da qualidade, especialmente para as pequenas e médias empresas que normalmente enfrentam grandes dificuldades devido à limitação dos recursos disponíveis para a selecção e adopção de um novo sistema de software. Na presente conjuntura, nacional e internacional, a importância de ser empreendedor assume-se cada vez mais como uma questão estratégica e já não depende exclusivamente da vocação e perfil individual. Em muitos casos é a única forma de gerar uma fonte de rendimentos, face a um mercado de trabalho fechado, levando muitos portugueses a criar um negócio próprio. Este relatório de projecto apresenta uma prova de conceito da implementação de um sistema integrado de gestão empresarial, pronto a usar, de apoio a micro e pequenas empresas de venda on-line através da integração de aplicações grátis ou *open-source*. O objectivo é minimizar os custos numa fase inicial do negócio aumentando a sua eficiência através do apoio das suas operações básicas essenciais.

Palavras-Chave

accounting, billing, e-commerce, ERP, FLOSS, 'free business software', 'systems integration'

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1
Introdução

Numa publicação do *blog* oficial da Google a 21 de Dezembro de 2009, o Vice-Presidente Sénior responsável pela Gestão de Produtos Jonathan Rosenberg escreveu: "Na Google acreditamos que sistemas abertos triunfam. Eles levam a uma maior inovação, valor e liberdade de escolha para os consumidores, e um ecossistema mais enérgico, lucrativo e competitivo para os negócios". No seu relatório, Open for Business, Jim Norton defende que são vários os benefícios de software open-source nas empresas uma vez que permitem explorar mais rapidamente novas tecnologias, recorrer a uma comunidade global para conhecimento especializado e resolução de problemas, e reduzir custos. O software open-source é grátis na maioria dos casos, por oposição aos custos normalmente pagos para ter acesso a software proprietário. Contudo esta não é a base correcta no que toca a comparação de custos. A medida que seria apropriado considerar é o custo total de posse (TCO[1]) durante o tempo de vida de software. Jim Norton apresenta os resultados de um relatório levado a cabo pela *London School of Economics* (LSE) para o TCO do Gabinete Governamental do Reino Unido em 2011. O relatório, baseado em 32 organizações, descobriu que o TCO era muitas vezes inferior, no geral, com a utilização de software open-source. Ressalvando que a análise deve ser feita caso a caso para diferentes áreas e fases do ciclo de posse do software, a generalização é de que será possível às empresas poupar até 20% dos custos.

A utilização de software *open-source* pode ter um grande impacto no arranque de pequenas empresas e proprietários em nome individual. Na presente conjuntura, nacional e internacional, a importância de ser empreendedor assume-se cada vez mais como uma questão estratégica e já não depende exclusivamente da vocação e perfil de uma pessoa. Em muitos casos é a única forma de gerar uma fonte de rendimentos, face a um mercado de trabalho fechado, levando muitos portugueses a criar um negócio próprio. A taxa de desemprego entre os jovens em Portugal chegou aos 42,1% no primeiro trimestre de 2013, afectando 166 mil pessoas entre os 15 e os 24 anos (Instituto Nacional de Estatística, 2013). No primeiro trimestre de 2014 o desemprego de muito longa duração apontava para 326,8 mil pessoas à procura de trabalho há mais de dois anos, perfazendo 43,9% dos desempregados. Os desempregados que estão sem trabalho há mais de um ano perdem competências e motivação, e apresentam uma acrescida dificuldade em participar numa retoma económica futura caso não lhes seja prestado um apoio adequado. Muitos trabalhadores, incluindo parte dos jovens mais talentosos e qualificados, têm vindo a ser empurrados para a emigração. Sem conseguirem encontrar uma fonte de rendimento, muitos optam por ser patrões de si próprios iniciando uma actividade independente. Esta situação forçou algumas pessoas a tornarem-se empreendedoras, quer o quisessem quer não. É neste contexto que o software *open-source* surge como um elemento potencial para determinar o sucesso ou insucesso das micro e pequenas empresas criadas.

O software *open-source* traz uma alternativa que aborda muitos dos problemas-chave de software comercial proprietário. Começa por oferecer ao utilizador uma base de código como um ponto de partida. O utilizador pode assim experimentá-lo gratuitamente, na maioria dos casos, e verificar se este se adequa às suas necessidades. De simples ferramentas de contabilidade até sistemas ERP plenamente desenvolvidos, o software open-source pode oferecer opções gratuitas para pequenas empresas que não têm o orçamento para aplicações empresariais caras.

[1] Tocal Cost of Ownership

Este trabalho apresenta uma prova de conceito da implementação de um sistema integrado de gestão empresarial, pronto a usar, de apoio a micro e pequenas empresas de vendas on-line, através da integração de aplicações grátis ou *open-source*. O objectivo é minimizar os custos numa fase inicial do negócio, aumentando a sua eficiência através do apoio das suas operações básicas essencias. A solução final, sem custos, contempla a integração de aplicações já desenvolvidas e permitirá ao utilizador fazer a gestão de produtos, aceitar encomendas on-line, emitir facturas, gerir a contabilidade da empresa e criar demonstrações financeiras. A integração cria um espaço de comunicação entre as aplicações de forma a evitar acções redundantes para o utilizador e a automatizar operações.

Capítulo 2
Estado da Arte

A implementação de um sistema de gestão empresarial que integre múltiplas aplicações impõe uma análise das actuais soluções disponíveis no mercado no que respeita às suas funcionalidades, suporte e capacidade ou potencial de integração com outras aplicações. Além disso, diferentes abordagens podem ser tomadas de acordo com as exigências gerais do sistema e características particulares de cada aplicação. Este capítulo apresenta um conjunto de aplicações de negócio e as vantagens e desvantagens de diferentes tipos de integração.

2.1. Sistema Integrado de Gestão Empresarial

Os ERP (do inglês Enterprise Resource Planning, em português Sistema Integrado de Gestão Empresarial, SIGE ou SIG) em termos gerais, são plataformas de software desenvolvidas para integrar os diversos departamentos de uma empresa, possibilitando a automação e armazenamento de todas as informações de negócios. Os ERP tradicionais eram sinónimo de grandes empresas. No entanto esta tendência tem mudado e as PMEs começaram a usar sistemas ERP para se tornarem mais competitivas, com melhor capacidade de resposta às exigências do mercado e de forma a melhorarem o seu desempenho operacional (Deep, et al.,2008). O trabalho de Iskanius et al. (2009), explora a experiência do uso de sistema ERP em pequenas empresas e é um exemplo de investigação feito nesta área para empresas de menor dimensão. O estudo baseado na motivação para usar sistemas ERP em pequenas empresas resultou na conclusão de que as pequenas empresas se encontram motivadas para melhorar os seus planos de procedimentos e melhorar a flexiblidade perante os clientes. (Iskanius et al., 2009:9) (Adam, Kotzé, & Merwe, 2011). Os sistemas de ERP têm a sua origem no planeamento de requisitos de produção e gestão de inventário (MRP) (Anderegg, 2007). Os sistemas actuais contêm módulos não apenas para gestão de material mas também para suporte a vendas, contabilidade, recursos humanos e outras funções que apoiam as operações de negócio.

Os sistemas ERP Livres de Código Aberto (FOS-ERP) têm tido uma aceitação crescente e, consequentemente, estão a aumentar a sua quota de mercado. Este crescimento é ainda recente, mas já era claramente evidente no final da década passada, (LeClaire, 2006), (Kissinger, 2008). Embora a sua importância em termos de mercado seja crescente, o FOS-ERP não está ainda suficientemente estudado em bases científicas académicas. Contudo é notório que o panorama está a mudar. De acordo com Kim e Boldyreff (2005) até setembro de 2005 apenas tinha sido publicado um artigo sobre o ERP *open-source* "considerando todos os periódicos e eventos da Association for Computing Machinery – ACM e da IEEE Computer Society". O artigo em questão (Smets-Solanes & Carvalho, 2003), visava uma quinta geração de ERP baseada em arquitecturas *open-source*. E, na verdade, muitos dos trabalhos relevantes sobre o tema estão publicados em revistas não académicas, como são os casos de LeClaire (2006), Kissinger (2008), acima referidos. Ainda assim, é já possível encontrar trabalhos de revisão sobre o tema, como, por exemplo (Carvalho & Campos, 2009) e teses de mestrado (Huq, 2010).

Existem actualmente vários sistemas de gestão empresarial *open-source* como Openbravo[1] e Adempiere[2]. Apesar de alguns destes sistemas disponibilizarem um vasto conjunto de

[1] http://www.openbravo.com/
[2] http://www.adempiere.org/

módulos de gestão empresarial, como gestão de recursos humanos e gestão de clientes. No entanto ficam aquém na disponibilização de integração com plataformas *e-commerce* e em cumprir as condições necessárias para facturação segundo a legislação portuguesa, pouco contribuindo assim para a solução final.

2.2. Aplicações de Gestão Empresarial

Existem milhares de aplicações úteis para empresas disponibilizadas sob uma licença *open-source*. Muitas vezes estas aplicações competem entre si e a melhor escolha para o utilizador pode não ser evidente. Pode depender de várias características como a linguagem de programação utilizada, alguma funcionalidade em específico ou o suporte oferecido pela comunidade. Esta secção apresenta algumas das aplicações mais populares para cada área de vendas, facturação e contabilidade e os motivos que levaram às escolhas das aplicações que farão parte da solução final.

2.2.1. Vendas

A importância do *e-commerce* continua a crescer. Mais e mais pessoas compram os seus produtos e serviços on-line e fazem-no de diferentes lugares, como em casa ou no trabalho. O *e-commerce* tornou-se numa grande influência sobre a economia europeia. Em 2013 o Produto Interno Bruto (PIB) europeu foi superior a 16,4 triliões de euros com o e-PIB a representar 2,2% deste número (Ecommerce Europe, 2014). Segundo o relatório apresentado pela "Ecommerce Europe" é esperado que este valor duplique em 2016 e triplique em 2020. Existe um número considerável de soluções *e-commerce open-source* (Webnet Hosting, 2013) sendo as plataformas Magento[1], Prestashop[2] e OpenCart[3] as mais populares de momento. Na **Error! Reference source not found.** é possível ver os termos mais populares para plataforma. A popularidade é calculada pela ferramenta Google Trends e baseia-se na frequência com que os termos são pesquisados.

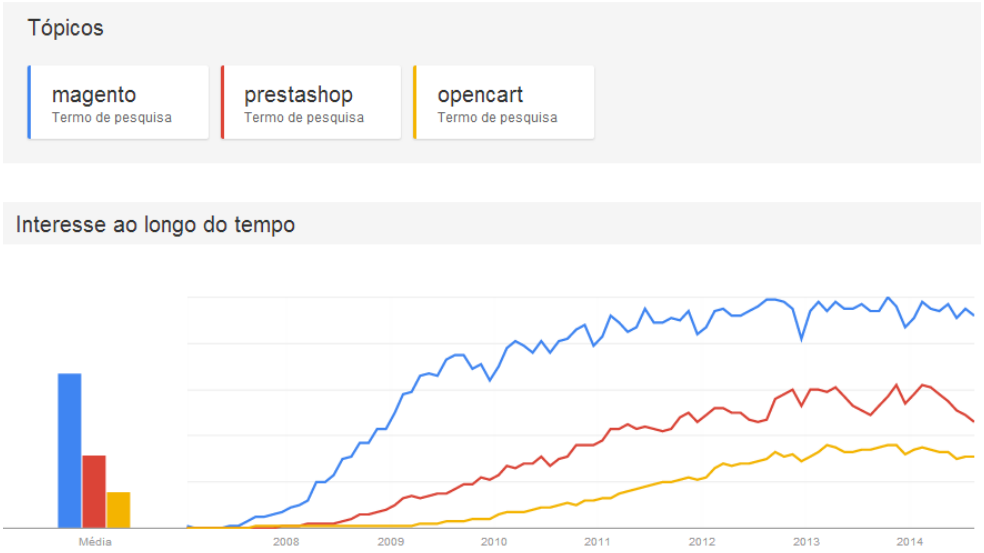

Figura 2.1. Comparação de tendências de software e-commerce open-source (Google Trends)

[1] http://magento.com/
[2] http://www.prestashop.com/
[3] http://www.opencart.com/

Magento é uma plataforma *web* de *e-commerce open-source* baseado em PHP e MySQL e utiliza o *framework* Zend. Foi desenvolvido pela Magento Inc que hoje é uma divisão da eBay. Empresas como a Nike, Olympus e Gant[1] impulsionam actualmente o seu negócio on-line através do Magento. O Magento é uma plataforma robusta, sob a licença OSL[2], que permite configurar várias das funcionalidades já existentes assim como adicionar novas. Além das funcionalidades básicas para uma loja on-line como gestão da página e catálogo, carrinho de compras, pesquisa e *checkout* apresenta também características como SEO[3], análises e relatórios, e suporte para dispositivos *mobile*. O Magento Community Edition é a versão gratuita da plataforma. Além de ser um sistema de *e-commerce* bastante completo, é possível instalar novas funcionalidades criadas por colaboradores de todo o mundo. Estes complementos são chamados de módulos.

Prestashop é uma plataforma *e-commerce open-source* e está sob a licença OSL. É escrito em PHP e usa uma base de dados MySQL. Originalmente criado em França, é actualmente usado em mais de 165 mil lojas em todo o mundo e foi considerado "Best Open-source Business Aplication" em 2010 e 2011. PrestaShop também oferece uma impressionante variedade de funcionalidades de SEO, opções de gestão de cliente e opções de oferta. Como uma solução *open-source*, a aplicação oferece na sua base todas as características que uma loja on-line necessita numa fase inicial. Conta com mais de 300 extensões que podem ser adicionadas à loja.

OpenCart é uma plataforma *e-commerce open-source* semelhante ao Magento e Prestashop. Com uma menor comunidade, tem-se tornado popular por ser uma versão mais leve, intuitiva e fácil de administrar. São variadas as opiniões dos utilizadores e as soluções apresentadas e é difícil estabelecer a solução *de facto* para plataformas e-commerce. Na Tabela 2.1 são apresentadas algumas características para cada plataforma. Esta análise foi feita a partir da recolha de dados das páginas oficiais, instalação e estudo das três plataformas e recolha de opiniões mais consensuais da comunidade on-line. Existe algum consenso que a plataforma Magento é a solução e-commerce mais popular do momento. Contudo pode não ser necessariamente aquela que melhor serve as necessidades de todos os interessados em implementar um negócio on-line. Apesar de ser uma das soluções mais robustas, altamente personalizável e com maior comunidade para apoio é também considerada menos intuitiva do ponto de vista do proprietário da loja e com uma curva de aprendizagem íngreme para programadores.

	Magento	**Prestashop**	**OpenCart**
Licença	OSL	OSL	GPL
Linguagem	PHP	PHP	PHP
Base de dados	MySQL	MySQL	MySQL
Última versão	1.9.0.1 16 Maio 2014	1.6.0.9 31 Julho 2014	1.5.6.4. 23 Abril 2014
Utilização	Médio	Fácil	Fácil

[1] http://magento.com/customers/customer-showcase
[2] Open Software License
[3] Search Engine Optimization

Página	Profissional	Profissional	Profissional
Lojas	> 240 mil	> 185 mil	NA
SEO	Muito bom	Bom	Bom
API	REST	REST	REST (extensão)
Recomendado para	Proprietários de lojas	Empresas, Proprietários de lojas	Proprietários de lojas

Tabela 2.1. Comparação de plataformas e-commerce

As soluções apresentadas têm diferentes níveis de maturidade mas apresentam na sua base uma solução que cumpre as necessidades básicas de uma loja on-line. No entanto, no âmbito da implementação de uma solução de gestão empresarial, é necessário ter em conta a sua capacidade de extensão e integração com outras aplicações. A plataforma Magento é não só a solução mais popular mas também a que tem mais tempo de mercado e marca a existência de uma maior comunidade. Apesar de uma curva de aprendizagem mais exigente e que poderá no futuro ser uma ameaça face a soluções mais intuitivas tanto do ponto de vista do utilizador como do programador, é neste momento a plataforma e-commerce mais estável e que mais pode oferecer de funcionalidades extra e integração com aplicações já existentes. É por isso a escolha como sistema de vendas.

2.2.2. Facturação

Através da Portaria n.º 22-A/2012 de 24 de Janeiro, a Autoridade Tributária e Aduaneira[1] (ex-DGCI) redefiniu os requisitos que visam implementar mecanismos de controlo e auditoria nos programas de facturação utilizados pelos contribuintes, originalmente definidos na portaria n.º 363/2010. Esta medida tem como finalidade reforçar este instrumento de combate à fraude e evasão fiscal, facilitando o cruzamento de dados. Nesse sentido, os sujeitos passivos de IRC[2] e IRS[3] que utilizam programas informáticos de facturação ficam obrigados a utilizar programas que tenham sido previamente certificados pela Autoridade Tributária e Aduaneira (AT).

De acordo com o art.º 2.º da Portaria n.º 363/2010, de 23 de junho, com a redacção dada pela Portaria n.º 22-A/2012, de 24 de janeiro, todos os sujeitos passivos de IRS ou IRC, com as excepções constantes do n.º 2 do artigo 2.º, passam a estar obrigados a utilizar, exclusivamente, um programa de facturação certificado. Deixou, portanto, de ser possível, após 1 de abril de 2012, o uso de máquina registadora ou a facturação manual emitida em documentos impressos por tipografias autorizadas, passando a sistema universal de faturação a utilização de programa certificado. A lei passou a impor a utilização de programa certificado como forma, exclusiva, de emissão de faturas (Portal das Finanças).

Esta obrigatoriedade de utilização de programa certificado vigora desde 1 de abril de 2012 para sujeitos passivos com volume de negócios superior a € 125 000 e desde 1 de janeiro de 2013 para sujeitos passivos com volume de negócios superior a € 100 000.

[1] http://info.portaldasfinancas.gov.pt/pt/at/at_index.htm
[2] Imposto sobre Rendimento das Pessoas Colectivas
[3] Imposto sobre Rendimentos Singulares

Para o contribuinte utilizador de soluções obrigadas à certificação, a componente mais visível do processo de certificação está nos procedimentos de facturação e na informação representada nos documentos de facturação impressos e entregues aos clientes. Nos documentos de venda e transporte impressos (facturas, guias de remessa e transporte, talões de venda e documentos equivalentes) existe uma alteração evidente e explicita para os utentes da informação: a expressão "Documento processado por computador", um hábito nos últimos anos, será substituída de acordo com a legislação agora em vigor, pela expressão "Processado por programa certificado nº <Número do certificado atribuído pela AT>/AT" antecedida de 4 caracteres da assinatura digital do documento em causa (softwarecertificado.com) como apresentado na Figura .2.

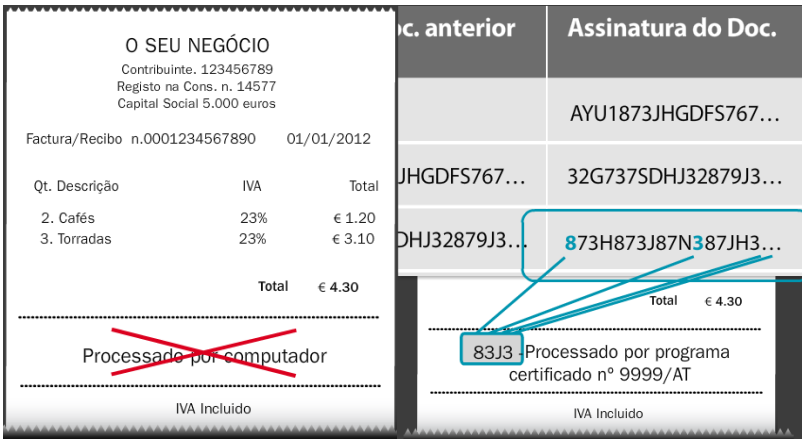

Figura 2.2. Comparação Alterações Factura/Recibo

A Autoridade Tributária disponibiliza no Portal das Finanças a lista de todos os programas que se encontram devidamente certificados[1]. As aplicações *open-source* mais populares do momento são: OpenK[2] e Evaristo[3].

O software OpenK da empresa KTC é baseado no Openbravo um dos softwares *open-source* de gestão e facturação mais usados em todo o mundo. Com o OpenK é possível ter acesso a uma solução mundial adaptado à realidade nacional e certificado pela Autoridade Tributária portuguesa (nº 1345). No entanto apenas as instalações do OpenK efectuadas e mantidas pela KTC estão certificadas pela AT (Manual de Utilização do OpenK POS, 2013). Assim, após o termino do contracto de serviços entre a KTC e o Cliente a aplicação deixa de se considerar certificada e deve ser removida permanentemente. A instalação tem um custo base anual de 100€, mais IVA.

O Evaristo é um programa de facturação *open-source* da empresa Memória Persistente. No entanto a versão certificada pela AT (nº1624) é a versão de código fechado. Tem também uma licença anual de utilização base de 100€.

Outras soluções são programas certificados que têm versões grátis. Nestas destacam-se Primavera Express[4] e Projecto Colibri[5]. Ambas as aplicações permitem executar as

[1] https://www.portaldasfinancas.gov.pt/pt/consultaProgCertificadosM24.action
[2] http://www.openk.pt/
[3] https://t5.m16e.com/wiki/pageviewer/view/6
[4] http://www.primaverabss.com/
[5] http://projectocolibri.com/

funcionalidades básicas de gestão de uma empresa no que respeita às vendas/POS, inventário e conta corrente de clientes. Além disso permitem a configuração do nome e logotipo da empresa para aplicar nos documentos necessários à actividade comercial como facturas, notas de crédito/débito, e guia de remessa e de transporte. Também garantido pelas duas aplicações é a emissão do ficheiro SAFT-PT[1], um documento de emissão obrigatória desde 1 de Janeiro 2010. A aplicação Primavera Express permite efectuar a facturação e gestão de empresas com um volume de facturação até 30 mil euros.

O Projecto Colibri iniciou o desenvolvimento uma API em Java[2] *open-source* para integração com o Magento. Suporta também a instalação da plataforma em mais do que uma base de dados, incluindo MySQL, como se pode ver na Tabela.3. A preocupação do Projecto Colibri em integrar com lojas on-line, e através da disponibilização de código *open-source*, faz do Projecto Colibri o melhor candidato para integração no sistema em consideração.

	Primavera Express 7.60	**Projecto Colibri RCP 9**
Certificado	nº1450 (versão 7.50)	nº38 (versão 6)
Programado em	.NET Framework	JAVA
Base de Dados	SQL Server	H2, MySQL e PostgreSQL
Limite volume de facturação	até 30 mil euros	-

Tabela 2.2. Comparação Primavera Express e Projecto Colibri

2.2.3. Contabilidade

A gestão contabilística de uma empresa abrange mais do que manter um conjunto de registos equilibrar a conta corrente do negócio. Este registo permite fazer uma melhor gestão da empresa identificando a fonte de despesas assim bem como distribuição de lucros. A gestão financeira implica também planear e criar um orçamento de acordo com as metas financeiras e estratégias definidas para o negócio. Para auxiliar nesta tarefa existem várias aplicações especializadas que possibilitam, entre outras funcionalidades, registar as despesas e receitas sob diferentes categorias e gerir relatórios e gráficos personalizáveis e que dão diferentes perspectivas do negócio. Esta análise não compreende o estudo das obrigações de uma empresa perante os diferentes regimes de contabilidade. A escolha foi feita com base na instalação das aplicações GnuCash[3], Adempiere e OpenBravo (ver Figura 2.3) e respectivo estudo. Na ausência de características distintivas essenciais para o sistema a apresentar a escolha baseou-se em segunda instância nos comentários de utilizadores e a sua satisfação global com as aplicações apresentadas. A aplicação escolhida foi o GnuCash, um software de finanças e contabilidade sob a licença GPL e disponível para várias plataformas. Apesar de estar escrito na linguagem de programação C tem interfaces em desenvolvimento para python e perl e permite armazenar os seus dados em XML, SQLite e MySQL.

[1] Standard Audit File for Tax Purposes
[2] https://code.google.com/p/projectocolibri/
[3] http://www.gnucash.org/

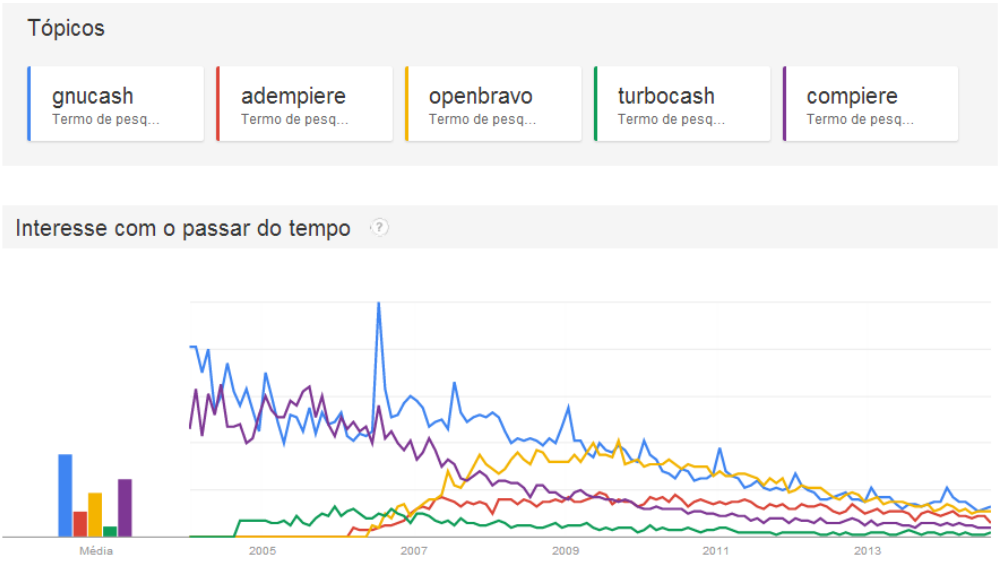

Figura 2.3. Comparação de tendências de software de contabilidade open-source (Google Trends)

2.3. Integração de Aplicações

Numa situação normal o sistema de informação de uma empresa necessita recolher dados de diferentes sectores. Esses sectores são geralmente geridos por sistemas diferentes tornando os dados dissociados. A integração das várias componentes levanta vários desafios e deve ser por isso devidamente analisada. Ao longo do tempo, estes desafios foram ultrapassados com quatro abordagens diferentes: transferência de ficheiros, base de dados partilhada, invocação remota de métodos e *messaging* (Hohpe & Woolf, 2004). Um ponto chave é que todas as integrações têm que lidar com o facto que as aplicações são diferentes – ou seja, com diferentes linguagens de programação, plataforma ou formato de dados. É também esta a realidade para as aplicações sob consideração neste sistema. A solução de integração tem por isso que criar uma interface que junte diferentes tecnologias. E, acima de tudo, a mudança é inevitável. A integração entre aplicações dificilmente poderá ser desenhada para se adaptar a todas as mudanças nas aplicações ou à introdução de novas componentes. No entanto, a robustez da integração e a sua sensibilidade a adaptações será um factor a ter em conta no desenvolvimento deste projecto. Esta secção pretende informar sobre alguns padrões normalmente implementados e como estes poderão ou não servir o propósito do projecto. De notar que os tipos de integração mencionados são genéricos e a sua implementação pode variar de várias maneiras, sendo que a solução final pode mesmo incluir mais do que um tipo.

2.3.1. Transferência de ficheiros

Transferências de ficheiros é um tipo de integração que pode fazer sentido aplicar quando é necessário partilhar informações entre aplicações que usam diferentes linguagens, que se baseiam em diferentes plataformas e/ou as aplicações têm noções diferentes de como o negócio opera. Os ficheiros são um mecanismo universal de armazenamento e podem assim servir como base de comunicação independentemente da linguagem. Uma das decisões importantes a fazer é o formato de ficheiro a utilizar. O formato actualmente com mais consenso é o formato XML, um standard que a maioria das tecnologias de hoje em dia consegue não só ler como também processar para a sua linguagem natural. Outra escolha a

fazer é o instante em que são produzidos e consumidos. Uma vez que existe alguma exigência na criação e manipulação de ficheiros, a frequência com que esta operação é feita deve ser reduzida. Uma grande vantagem de utilizar ficheiros é que estes não necessitam ter conhecimento do que se passa na aplicação. Os módulos integradores que manipulam os ficheiros podem escrevê-los de maneira diferente consoante a aplicação que o irá ler ou escrevê-lo num formato único ficando à responsabilidade da aplicação que o irá consumir decidir como extrair e processar a informação. Como resultado, as aplicações encontram-se bastante desacopladas uma das outras.

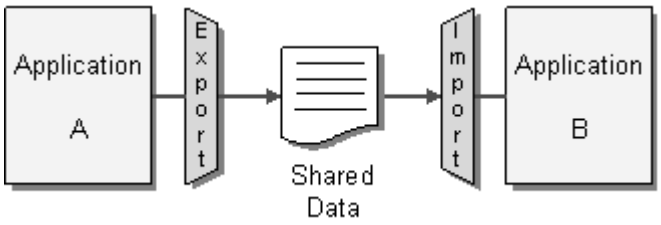

Figura 2.4. Esquema de integração por transferência de ficheiros

Este tipo de integração é uma opção evidente para o problema em questão, considerando um leque de opções open-source que permite alterar as aplicações de modo a que sejam capazes de ler e escrever para um formato de ficheiro a definir. Além disso, independentemente de como o software é distribuído, não é incomum que o próprio tenha capacidade de importar e exportar dados numa estrutura standard do negócio, ou numa estrutura própria mas que pode ser eventualmente replicada. No entanto, existem algumas contrariedades no desenvolvimento de uma integração que use ficheiros. Há que ter em atenção requisitos como convenção de formatos, estrutura, nomes do ficheiros, garantia da sua unicidade, que aplicações devem apagar ficheiros antigos, e quando. Adicionalmente, é preciso garantir a existência de um mecanismo que bloqueie o ficheiro enquanto está a ser escrito e outro que seja responsável por transferir o ficheiro remotamente, se necessário. Todos estes requisitos têm um custo acrescido no desenvolvimento.

Outra questão importante a abordar é o sincronismo. Uma vez os ficheiros são actualizados com uma certa frequência, os dados podem não fazer correspondência durante um determinado período. A questão do sincronismo pode não ser necessariamente um problema para o nosso problema em particular, dependendo do sentido em que é feita a sincronização dos dados. Por exemplo, a actualização dos dados relativamente a encomendas no software de contabilidade pode ocorrer apenas mediante pedido do utilizador. Este trabalho está limitado ao contexto de três aplicações. Não significa, portanto, que não deve ser analisado tendo em consideração a implementação de outras operações.

2.3.2. Partilha de Base de Dados

A transferência de ficheiros permite às aplicações partilharem dados, mas a actualização dos dados pode não ocorrer com frequência suficiente para muitos dos negócios modernos. Além disso, o formato e estrutura dos ficheiros podem nem sempre ser suficientes para garantir a mesma interpretação e leitura por diferentes aplicações. Nesse caso, uma solução pode ser um local de armazenamento centralizado onde as aplicações partilham informação e cada uma tem acesso para aceder aos dados quando necessitar. Desta forma, se todas as

aplicações dependerem da mesma base de dados é garantido que os dados se encontram consistentes entre as mesmas, sem diferentes interpretações ou semânticas.

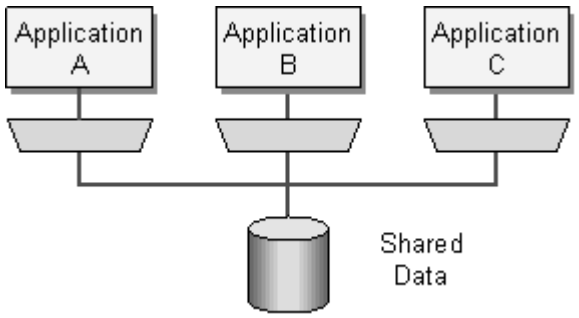

Figura 2.5. Esquema de integração por partilha de base de dados

Uma das dificuldades neste tipo de integração é encontrar um modelo estrutural único que sirva as necessidades de múltiplas aplicações. Muitas vezes a decisão de integrar aplicações aparece apenas mais à frente, quando as aplicações foram desenvolvidas separadamente com as suas base de dados e modelo relacional próprio. Nesta fase a integração com um modelo relacional comum pode mostrar-se um grande desafio. Outro problema a ter em atenção é que múltiplas aplicações a ler e a actualizar dados da mesma base de dados podem causar problemas de performance e dificultar ou bloquear operações umas das outras. Para aplicações que se encontram em locais diferentes, aceder a uma única base de dados remota não garante muitas vezes um tempo de resposta adequado. A distribuição de base de dados para uma ligação local com as várias aplicações resolve este problema mas dificulta a resolução de conflitos entre base de dados remotas entre si. No caso particular deste projecto, não são previstos problemas de desempenho com alta concorrência no acesso aos dados dado que as operações consideradas são minimamente rotineiras.

2.3.3. Invocação Remota de Métodos

A transferência de ficheiros e partilha de base de dados permite às aplicações partilharem informação entre si, uma característica importante da integração de aplicações. Alterações nos dados significam muitas vezes acções que podem afectar mais do que uma aplicação. Ter aplicações a invocar esses processos directamente na estrutura de outras aplicações implica que seria necessário terem conhecimento do estado interno uma das outras. Esta questão é resolvida através de um mecanismo de encapsulamento em que as aplicações escondem o seu estado interno mas oferecem uma interface pública para que sejam invocadas certas acções. Desta forma, se uma aplicação necessita de informação que é controlada por outra pode requisitá-la directamente.

Uma base de dados partilhada é o antónimo de encapsulamento. Mudanças na aplicação podem implicar mudanças na estrutura da base de dados. Mudanças essas que se podem repercutir noutras aplicações. O resultado disso é que empresas que usam bases de dados partilhadas tenham alguma relutância em alterar a sua estrutura, o que significa que a adaptação às mudanças do negócio é menos fluída.

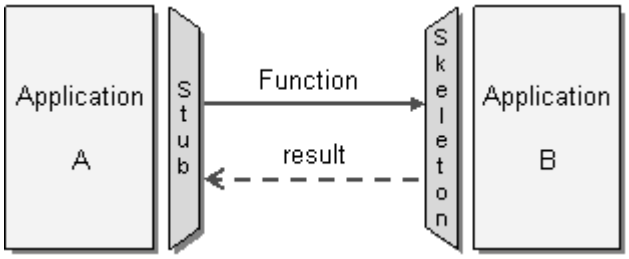

Figura 2.6. Esquema de integração por invocação remota de métodos

Apesar de estes métodos serem invocados como normalmente o veríamos num código de uma aplicação única, este tipo de integração não restringe a um tipo de linguagem. Algumas das tecnologias são centradas numa linguagem (e.g. Java RMI) mas a aplicação deste tipo de integração reside em vários módulos que comunicam entre si usando um protocolo pré-estabelecido. Algumas dessas tecnologias são CORBA e Web Services como SOAP e REST. A comunicação na maior parte dos casos é por isso independente da linguagem ou plataforma. Apesar da semelhança com a chamada normal de uma função local, a chamada remota levanta muitas outras questões de desempenho no sistema. Este tipo de encapsulamento ajuda a desacoplar as aplicações eliminando a necessidade de ter uma estrutura de armazenamento de dados partilhada. No entanto, existe ainda dependência entre as aplicações podendo um pedido ser passado ao longo de várias aplicações. Este fluxo ordenado de pedidos pode dificultar a alteração de um sistema independentemente, sem quebrar o sistema como um todo.

No âmbito deste projecto, este tipo de integração para software não open-source pode estar limitada pela sua capacidade em facultar APIs que permitam invocar as acções necessárias. Caso contrário, o desenvolvimento de métodos remotos está fechado.

2.3.4. *Messaging*

Transferência de ficheiros e bases de dados partilhadas permitem às aplicações partilhar os seus dados mas não a sua funcionalidade. A Invocação Remota de Métodos permite às aplicações partilharem funcionalidades mas essa partilha implica um acoplamento maior entre as aplicações.

Face à dificuldade ou mesmo inaptidão que a partilha de ficheiros e base de dados têm em prestar um serviço colaborativo entre aplicações, o RPC[1] parece ser a solução mais apelativa. No entanto, apesar de o RPC se assemelhar à chamada de métodos locais, na verdade a invocação dos métodos é lenta e estes estão muito mais aptos a falhar - e podem corromper o normal-funcionamento da aplicação à qual foi feita a chamada. Além disso, podemos não querer que as aplicações tenham informações umas das outras mesmo que seja só uma interface. Esta é uma particularidade encontrada na integração por partilha de ficheiros e na qual o *Messaging* se baseia, mas de uma forma muito mais rápida com o envio de pequenos pacotes rapidamente produzidos e transmitidos. Neste tipo de integração o receptor é imediatamente notificado quando um novo pacote estiver pronto para ser consumido.

[1] Remote Procedure Call

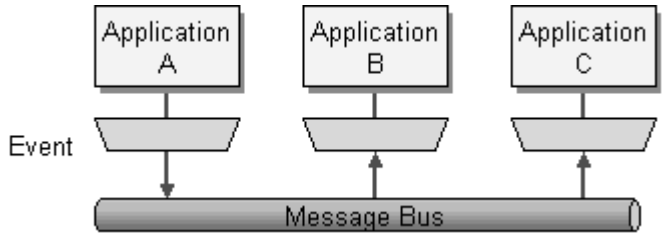

Figura 2.7. Esquema de integração por mensagens

Este tipo de comunicação assíncrona permite invocar um comportamento noutra aplicação, como um RPC, mas não requere, por exemplo, que ambas as aplicações estejam a funcionar ao mesmo tempo. Este tipo de integração tem até um desacoplamento maior que a partilha de ficheiros, pois as mensagens podem ser alteradas durante a transmissão sem o emissor ou receptor terem conhecimento desse aspecto. Por exemplo, se o emissor e o receptor não usarem o mesmo tipo de formato de dados, esta pode ser alterada no canal para poder ser lida pelo receptor.

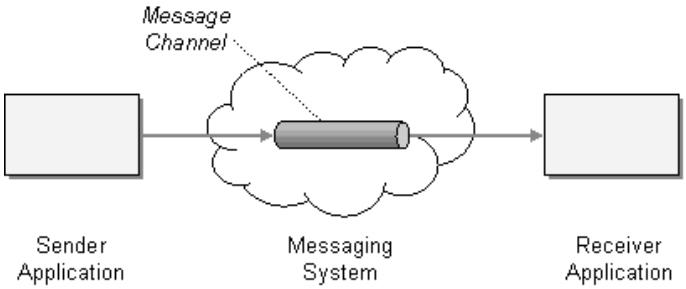

Figura 2.8. Esquema de um canal de sistema de mensagens

Apesar deste comportamento colaborativo entre aplicações não ser tão rápido como RPC, a aplicação que requisita informação não precisa ficar bloqueada em espera pela resposta.

A alta frequência de mensagens reduz mas não elimina os problemas de sincronismo referidos na integração para partilha de ficheiros. A principal desvantagem deste tipo de integração é que requere um componente extra na arquitectura, o agente que transfere e processa mensagens. Adicionar mais um componente num sistema pode levar a uma redução no desempenho e confiança no sistema. Além disso, muitos sistemas em que existe comunicação entre aplicações têm um comportamento síncrono por natureza com o emissor a querer necessariamente esperar pela resposta antes de tomar uma acção. Como a comunicação por mensagens tem uma natureza assíncrona, pode não ser uma boa solução para essas situações. Esta é no entanto uma opção viável no âmbito deste projecto pois as operações têm essencialmente o objectivo de exportação de dados não sendo esperada qualquer tipo de resposta do receptor

Capítulo 3
Método de Abordagem

A implementação passou pelo desenvolvimento de dois módulos para a plataforma Magento que exportam os dados para as bases de dados das aplicações Projecto Colibri e GnuCash (ver Figura .1). Cada módulo é responsável por proceder à sincronização unidireccional de dados para as respectivas aplicações, tendo como *frontend* do sistema a plataforma Magento.

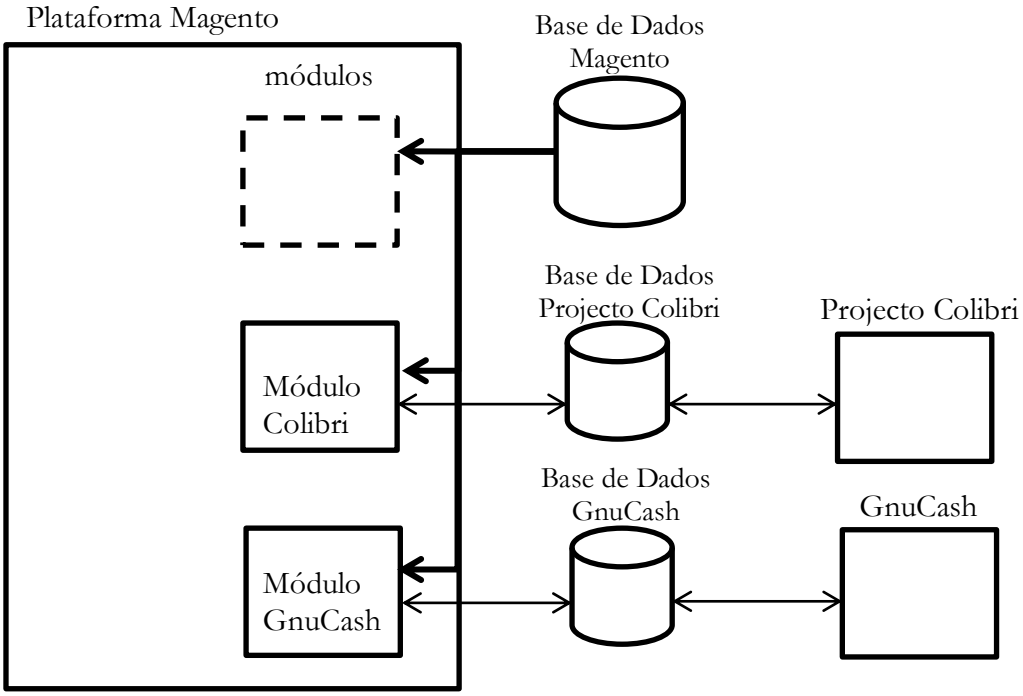

Figura 3.1. Arquitectura da solução implementada

A integração através das bases de dados das aplicações é a solução mais eficiente face à ausência de uma interface nas aplicações Projeto Colibri e GnuCash. Outra solução seria inverter a abordagem, passando as aplicações a pedir os dados através da API REST da plataforma e-commerce. No entanto esta abordagem iria requerer a alteração do controlo do sistema para as aplicações, o que é um obstáculo para o Projecto Colibri, como aplicação de código fechado. Esta abordagem podia no entanto ser tomada para a aplicação GnuCash. Foi tomada a decisão de aplicar o mesmo tipo de integração a ambas aplicações pela exigência de aprendizagem do *framework* Zend da plataforma-ecommerce.

As tecnologias usadas têm um papel importante na selecção das aplicações, nomeadamente o armazenamento dos dados. MySQL é uma base de dados open-source sob a licença GPL[1] e pode ser usada por software comercial numa empresa sem qualquer custo. Existe alguma confusão neste tema quanto à validade da licença quando utilizada num âmbito comercial. Só existe uma violação da licença se o motor do MySQL for alterado de maneira a só funcionar com o software em específico e este for vendido sem distribuição do código alterado (GNU General Public License, version 2). MySQL é uma das bases de dados mais

[1] GNU General Public License

populares e não sendo a única solução open-source é uma característica comum nas aplicações escolhidas e tem um papel essencial na implementação da solução implementada.

Um factor tido em conta é o acoplamento de aplicações. Aplicações integradas devem minimizar as suas dependências umas das outras de forma a que sua evolução não cause problemas. Quando as aplicações estão fortemente acopladas assumem várias condições nas outras aplicações e se quebradas essas condições a integração deixará de funcionar também. A implementação considerada para este trabalho depende essencialmente dos dados das entidades do Magento. Estes dados podem ser acedidos através do seu ORM, uma camada de abstracção mais resistente a alterações, sendo uma das componentes mais estáveis do sistema. Outra característica a ser minimizada é a intrusão. A integração das aplicações escolhidas é feita através da extensão das funcionalidades da plataforma e-commerce sem alterar a forma como as operações são conduzidas no Magento, nem os seus objectos. Para cada pedido ao sistema que implique a criação ou alteração de entidades da plataforma e-commerce, relevantes para as aplicações de facturação e contabilidade, é adicionado um passo extra no processo que mapeia e calcula os atributos a serem armazenados nas bases de dados das aplicações Projecto Colibri ou GnuCash.

Capítulo 4
Implementação

Este capítulo descreve os passos essenciais da implementação de dois módulos que estabeleçam uma ligação entre a plataforma e-commerce e as aplicações de facturação e contabilidade, Projecto Colibri e Magento.

A primeira secção deste capítulo irá abordar as instalações e configurações iniciais. Os passos essenciais da implementação de um módulo que permita exportar as entidades da plataforma e-commerce de forma a serem lidos por os demais componentes vão ser descritos mais detalhadamente na secção de Integração do Magento com o Projecto Colibri. Muitos desses passos são reproduzidos para a integração com a segunda componente, a de contabilidade, onde serão descritos os pontos em que as duas abordagens diferem.

O principal desafio do ponto de vista do desenvolvimento foi a adaptação ao *framework* Zend[1] sobre a qual está assente a plataforma Magento. Existe um certo consenso da comunidade Magento que este tem uma curva de aprendizagem íngreme. A documentação peca em explicar mais que do que alguns conceitos iniciais criando alguma incerteza dos passos a ser tomados a seguir, como descrito por alguns programadores:

> Classes are referenced dynamically, various aspects are contained in XML files and there is no clear flow that you can just debug through. The sheer volume of files and folders makes finding something unbelievably tedious.
>
> A final Cynical Note… Many people claim that the complexity of Magento is somewhat intentional. The profitability of Magento relys on consulting, technical support and installations. Making the codebase complex could mean that many developers will start out, get stuck and pay for help. If this is the intention of Varien then perhaps they have been very goal focused. (2009, pickledshark.com)

Num inquérito realizado pela equipa do *Zend Framework* (Zend Framework 2.0 Requirements , 2009) está patente que esta é uma opinião consensual da comunidade. As conclusões incluem questões como:

- Difficulty in the "first hour" with the framework.
- Uncertainty about the "next steps" following the quick start.
- Inconsistent APIs in the source code itself. One component may use "plugins," another "helpers," and yet another "filters."
- Uncertainty about where extension points exist, and how to program for them.
- Confusion over whether they can use Zend Framework only as an MVC stack or as individual components.

O *Zend Framework* aponta nos seus objectivos para a versão 2.0 a necessidade de suavizar a curva de aprendizagem, facilidade de extensão e melhor documentação.

Apesar de todos estes desafios a plataforma Magento é uma plataforma grátis, *open-source* e robusta com vasto conjunto de funcionalidades e por isso uma das melhores soluções do ponto de vista do cliente.

[1] http://framework.zend.com/

Do ponto de vista da implementação o desafio assenta na correspondência entre entidades sem falhas. De certa forma a implementação implicou algum trabalho de engenharia reversa uma vez que para aplicações de código fechado, como o software Projecto Colibri, é possível apenas observar o resultado final de uma acção. A ausência de um canal directo de comunicação com a aplicação não permite assim que essa mesma acção (e.g. criar novo cliente) possa ser pedida à aplicação. Foi por isso necessário estabelecer que alterações foram feitas e calcular como elas podem ser replicadas.

4.1. Instalação e configuração inicial

A utilização de uma solução integrada das várias componentes precisa à partida de reunir certas condições que serão semelhantes quer num ambiente de desenvolvimento quer num ambiente de produção. O software Projecto Colibri[1] e GnuCash[2] apresentam-se como softwares multi-plataforma que operam independentemente com o seu método próprio de armazenamento. No entanto no âmbito da integração entre os vários componentes estes dependerão de uma base de dados MySQL. Também a plataforma Magento tem a sua base em MySQL dependendo de um servidor Web que corra PHP. Esta secção aborda a instalação das aplicações e as tecnologias

4.1.1. Ambiente de desenvolvimento

O ambiente de desenvolvimento necessita essencialmente ser configurado para a plataforma Magento. XAMPP[3] é um pacote AMP (Apache, MySQL e PHP) independente de plataforma e permite instalar rapidamente as dependências da plataforma Magento. Apesar de o Magento ter passado a dar suporte a mais que um Web Server (Apache e Nginx) esta configuração é um padrão para o desenvolvimento em PHP (Glass, 2004). De notar que este pacote é destinado para um ambiente de desenvolvimento em que muitas das características de segurança estão desactivadas por padrão e não deverá ser implementando para um ambiente de produção para servir páginas Web. O XAMPP foi planeado de forma a que programadores e designers possam testar o seu trabalho sem necessitar de ligação à Internet e para facilitar o processo muitas das configurações de segurança estão inactivas por defeito.

O XAMPP está disponível na sua página[4] e a versão usada neste projecto foi a versão v1.8.3 (PHP 5.5.15) para Windows. Depois de instalado o XAMPP tem uma interface simples para gestão dos serviços que fazem parte do pacote (ver Figura 4.1). Os serviços que devem estar a correr são Apache e MySQL.

[1] projectocolibri.com
[2] http://www.gnucash.org/
[3] X (para qualquer dos diferentes sistemas operativos), Apache, MySQL, PHP, Perl
[4] https://www.apachefriends.org/download.html

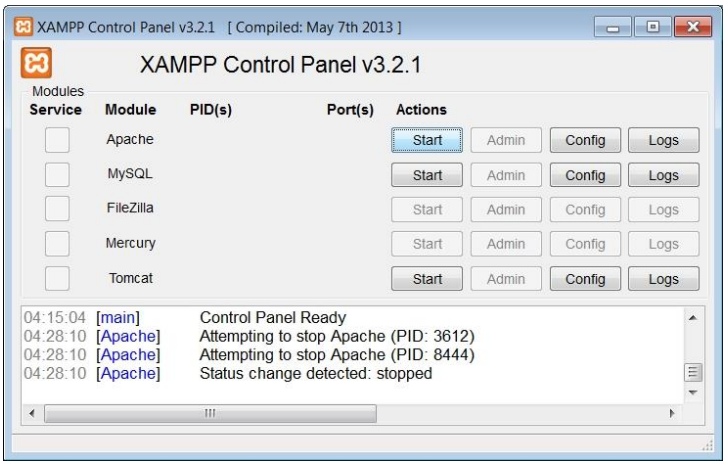

Figura 4.1. Painel de Controlo do XAMPP

Para testar se o servidor está a funcionar correctamente basta abrir um navegador de Internet e digitar `http://localhost/` ou `http://127.0.0.1/` na barra de endereços. Se estiver configurado correctamente a página apresentada será semelhante à da apresentada na Figura .

Figura 4.2. Splashscreen do XAMPP

A recomendação é que nas configurações o servidor seja 127.0.0.1 em vez de localhost pois nem sempre o último é reconhecido. Depois de instalado o XAMPP é preciso criar uma base de dados com o nome `'developdei'` através da ferramenta *phpmyadmin*, em `http://127.0.0.1/phpmyadmin`. A criação de novas bases de dados é feita através do separador 'Databases'. Existe uma conta com o nome de utilizador `root` e sem palavra-passe, criada por defeito. A criação de um novo utilizador é opcional mas aconselhado sendo que a palavra-passe será uma condição essencial quando configurado num ambiente de produção. No âmbito deste projecto foi criado um utilizador com as credenciais dev:estagio12345 e com permissões de dados (`SELECT, UPDATE, INSERT, DELETE, FILE`) e de estrutura (`CREATE, ALTER, INDEX, DROP, CREATE TEMPORARY TABLES, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, EXECUTE , CREATE VIEW, EVENT, TRIGGER`),

4.1.2. Magento

A edição Magento Community Edition está disponível no site oficial do Magento[1]. A versão usada na implementação deste projecto é Magento 1.9.0.1, que é a versão estável mais recente lançada a 16 de Maio de 2014.

Depois de feito o download a pasta deve ser movida para a pasta htdocs do XAMPP. Ao aceder à pasta através do endereço `127.0.0.1/magento/` é apresentado o assistente de instalação. É importante que a pasta tenha as permissões correctas para que a instalação possa prosseguir. As permissões definidas pelo Magento são 777 (chmod) caso seja implementado num sistema Unix.

O guia de instalação leva o utilizador a configurar as definições de zona e a ligação à base de dados. Os campos foram preenchidos de acordo com Tabela 4.1. O campo de prefixo pode ser deixado em branco pois o Magento assume que novas tabelas na mesma base de dados têm o mesmo prefixo, o que pode dificultar a integração com os outros componentes.

Região	Português (Portugal) / português (Portugal)
Fuso Horário	GMT Standard Time (Europe/London)
Moeda por omissão	Euro

Database Type	MySQL
Servidor	127.0.0.1
Nome da Base de Dados	developdei
Nome de Utilizador	dev
Palavra-passe do Utilizador	estagio12345
Prefixo das tabelas	

Tabela 4.1. Definições de zona e base de dados da instalação do Magento

De seguida o utilizador tem que configurar uma conta administrativa para a plataforma. O e-mail usado foi 'jclaro@student.dei.uc.pt' e a palavra-passe 'estagio12345'. Concluídos estes passos o sistema cria as tabelas na base de dados utilizadas pelo Magento. O utilizador pode depois aceder ao *frontend* ou *backend* da loja.

4.1.3. Projecto Colibri

O software de gestão comercial – Projecto Colibri – é também descarregado através da sua página oficial[2]. Para a implementação deste projecto foi usada a versão gratuita Projecto Colibri RCP 9.4.3 para o Windows 64 bits, lançada a 21 de Janeiro de 2014.

Após executar o instalador é apresentado o menu de autenticação (ver Figura 4.3). O software está inicialmente configurado para armazenar os dados numa base de dados H2, uma base de dados SQL em JAVA. Ao clicar em 'Novo Registo', no separador Base de Dados existem outros sistemas de gestão de base de dados tendo sido escolhido o MySQL (ver Tabela 4.2).

[1] http://www.magentocommerce.com/download
[2] http://www.projectocolibri.com/downloads-colibri

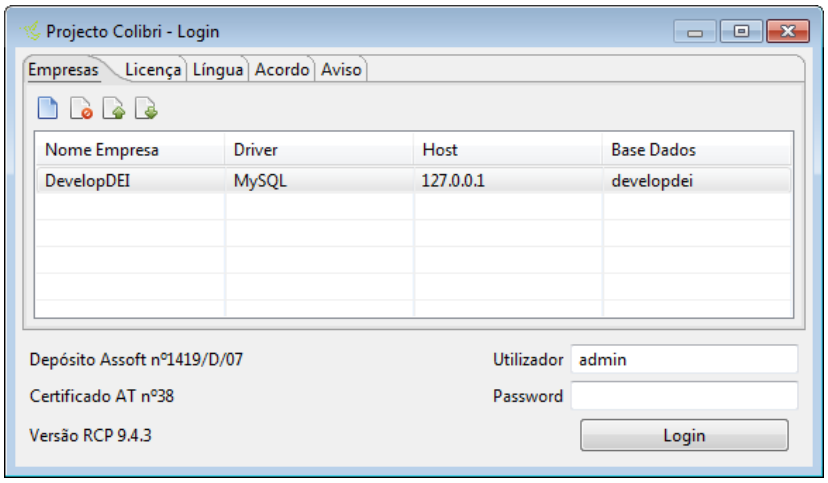

Figura 4.3. Interface de autenticação do Projecto Colibri

Driver	MySQL
Host	127.0.0.1
Base de dados	developdei
Utilizador	dev
Password	estagio12345

Tabela 4.2. Definições de base de dados da instalação do Projecto Colibri

A interface permite testar se a ligação à base de dados é feita com sucesso e após a confirmação da ligação foi gravado o novo registo. A autenticação por defeito do Projecto Colibri tem 'admin' como Utilizador e Password que podem ser alteradas posteriormente na Barra de Ferramentas > Ficheiro > Tabelas > Utilizadores.

4.1.4. GnuCash

A versão do GnuCash é 2.6.3 para Windows e pode ser descarregada através da página oficial[1] ou da página Sourceforge[2]. Depois de instalada e iniciada a aplicação o assistente de configuração permite escolher a moeda a utilizar e as categorias relevantes para o utilizador e que estão associadas a um conjunto de contas a ser criadas. Nesta fase é apenas relevante o método de armazenamento escolhido que é também o último passo apresentado pelo assistente. O formato por omissão é um ficheiro XML a ser mudado para a ligação com uma base de dados MySQL configurada com os mesmos parâmetros que os anteriores componentes (ver Tabela 4.3).

Data Format	mysql
Host	127.0.0.1
Database	developdei
Nome de Utilizador	dev
Password	estagio12345

Tabela 4.3. Definições de base de dados da instalação do GnuCash

[1] http://www.gnucash.org/
[2] http://sourceforge.net/projects/gnucash/

Criação de um novo módulo para Magento

Um dos primeiros passos para desenvolver para a plataforma Magento é desactivar a cache através do Painel de administração > System > Cache Management > Select All > Actions: Disable > Submit. Embora muito útil para o desempenho num ambiente de produção, a cache dificulta a depuração do sistema.

A força motriz do Magento pode ser encontrada em módulos individuais dentro da pasta `app/code`, que é dividida em três áreas: core, community e local.

- **core.** A pasta `app/code/core` contém as funcionalidade de gestão de produtos, categorias, clientes, pagamentos, e não deve ser modificado. O Magento é estruturado de a forma que seja possível alterar a funcionalidade destes serviços sem modificar directamente o seu core, o que garante que o sistema se mantém actualizável.
- **community.** Como o nome sugere, `app/code/community` é o lugar onde estarão módulos fornecidos por terceiros. Vários módulos estão disponíveis através do Magento Connect e podem ser instalados através do Gestor de Módulos do Magento.
- **local.** Em app/code/local podemos adicionar módulos para a nossa instalação do Magento em particular de modo a servir as nossas necessidades. É portanto esta a pasta onde foram desenvolvidos os módulos.

Dentro da pasta local foi criada uma nova pasta com o nome 'DevelopDei'. A primeira pasta tem a função de ser o *namespace* dos módulos, normalmente com o nome da empresa ou autor dos mesmos. Dentro dessa pasta foram criadas duas pastas para cada módulo a implementar, 'Colibri' e 'GnuCash' (ver Figura 4.4)

```
app
   code
      local
         DevelopDei
            Colibri
            Gnucash
```

Figura 4.4. Estrutura de pastas de um módulo para o Magento

O passo seguinte foi a configuração do módulo. O ficheiro de configuração deve ser criado em `<nome do módulo>/etc/config.xml` com configurações iniciais que não vão muito mais à frente que a definição da versão do módulo. O passo final visa informar o sistema que existe um novo módulo criando um ficheiro xml na pasta `app/etc/modules` (ver Figura 4.5)

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
   <modules>
      <DevelopDei_Colibri>
         <active>true</active>
         <codePool>local</codePool>
      </DevelopDei_Colibri>
    </modules>
</config>
```

Figura 4.5. Configuração de um novo módulo para Magento

4.2. Integração Magento e Projecto Colibri

Nesta secção irão ser descritos os passos essenciais da implementação de um módulo que permite exportar as entidades da plataforma e-commerce, relevantes no âmbito da facturação, de forma a serem lidos pelo Projecto Colibri. Com a abordagem tomada consegue-se reduzir o nível de intrusão naquele que é o processo natural da plataforma e-commerce minimizando comprometer o seu desempenho ou o sucesso da operação.

4.2.1. Modelo de Dados

O modelo de dados é um dos pontos-chave da implementação. A sua compreensão permite construir a correspondência entre os dois modelos, permitindo assim os dados possam ser exportados num formato compreendido pelo software de facturação. No Diagrama de Entidade Relacionamento da Figura 4.6 podemos ver os atributos associados à entidade produto do software de facturação Projecto Colibri (com a notação 'artigos'). O armazenamento de uma nova instância na base de dados implica o preenchimento destes atributos, mesmo que parcialmente, desde que reunidas determinadas condições (tipo de dados esperados, se estes são de carácter obrigatório, se cumprem as regras impostas para chaves estrangeiras e se fazem sentido no contexto da aplicação). Esta representação difere daquela implementada pela plataforma Magento que usa o sistema Entity Attribute Value Model (EAV). Este modelo generaliza o registo de uma entidade guardando todos os seus atributos em uma ou várias tabelas que armazenam os dados em pares atributo-valor (ver Figura 4.7). O modelo EAV torna-se especialmente útil quando os atributos das entidades variam significativamente em termos de tipos de dados e/ou quando existe um vasto número de atributos e o número de atributos associados à entidade é variável. Por exemplo, mesmo numa loja de produtos electrónicos enquanto um telemóvel estará associado a atributos como resolução de ecrã, sistema operativo e horas de autonomia, uma máquina de café terá atributos como potência, capacidade e sistema de filtragem de água.

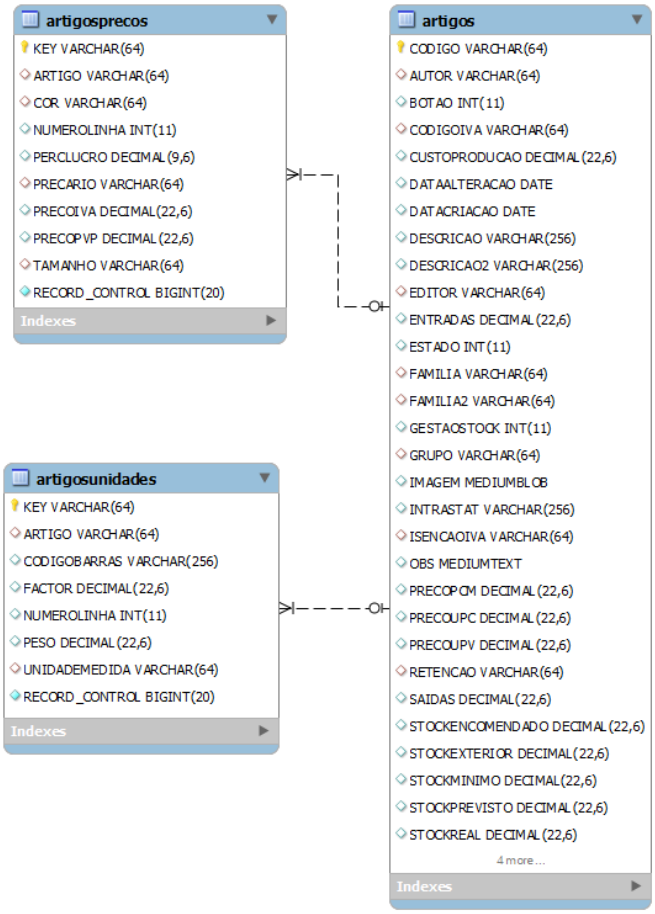

Figura 4.6. Diagrama Entidade-Relacionamento para Produtos

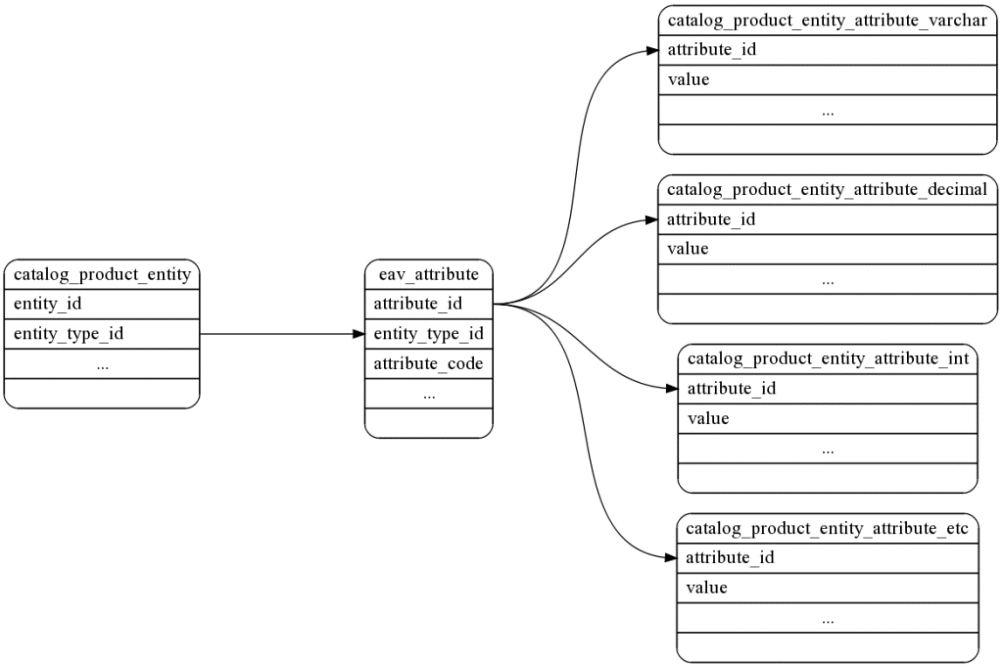

Figura 4.7. Modelo Entidade-Atributo-Valor do Magento

O modelo EAV dá assim flexibilidade ao sistema com a vantagem prática de evitar que a estrutura das tabelas tenha que ser alterada para adicionar novos produtos, que no modelo tradicional seria equivalente a uma nova coluna. No entanto existe um custo associado ao modelo: é impossível fazer o carregamento dos dados de um produto com uma chamada simples à base de dados. Apesar de o sistema EAV do Magento ser uma das características mais controversas (Ishenko, 2008) e com uma maior curva de aprendizagem esta complexidade não transparece no entanto da mesma forma para o ORM (Object Relational Mapping) da plataforma. A camada de abstração permite aceder aos atributos através de funções como `getPrice()`, `getComments()` ou `getTaxClassId()`. Os registos devolvidos por estas funções podem ser encontrados em diferentes tabelas com diferentes relações entre entidades mas a complexidade da chamada à base de dados é escondida do programador.

Nas figuras 4.8 e 4.9 estão representados os Modelos Entidade-Relacionamento de outras duas entidades relevantes para esta implementação: Cliente e Encomenda.

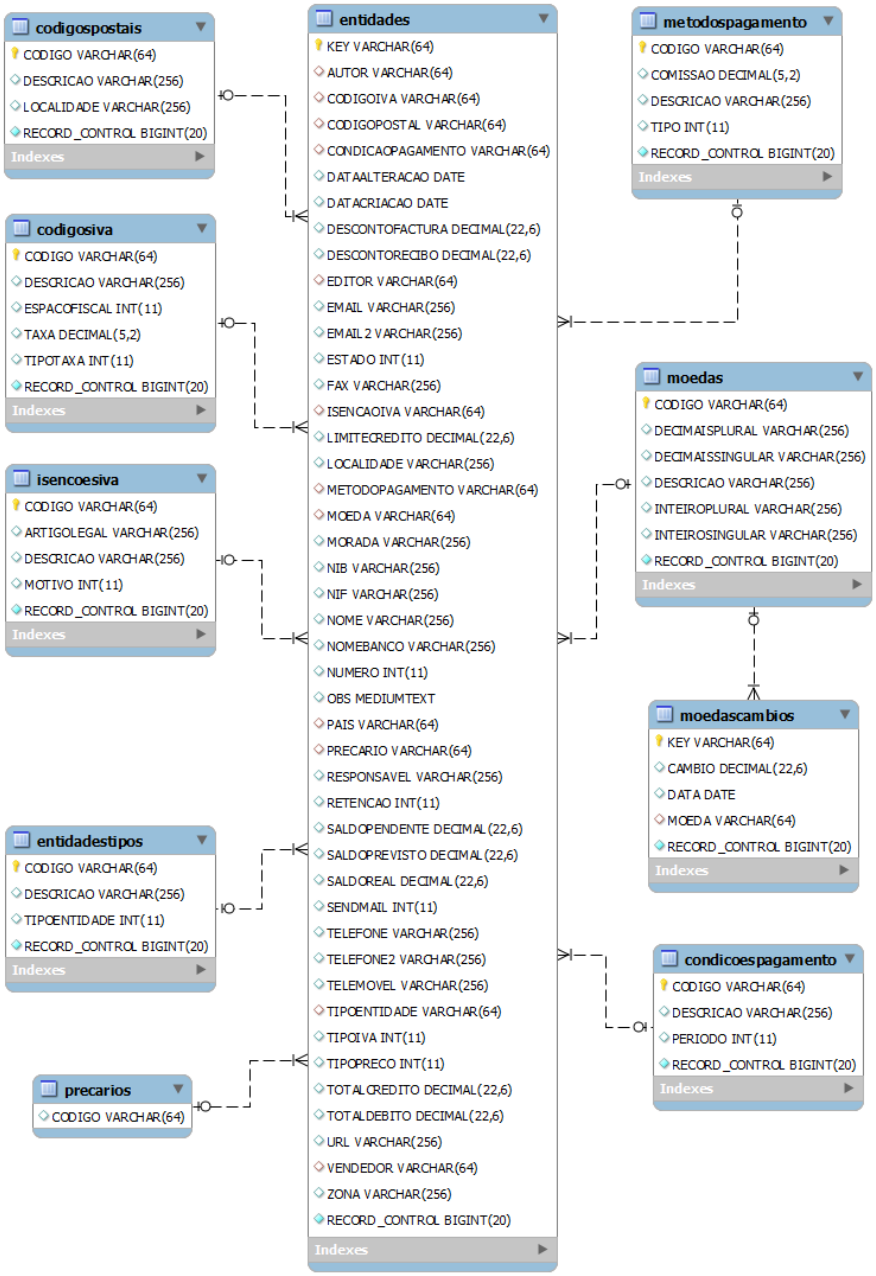

Figura 4.8. Diagrama Entidade-Relacionamento para Clientes

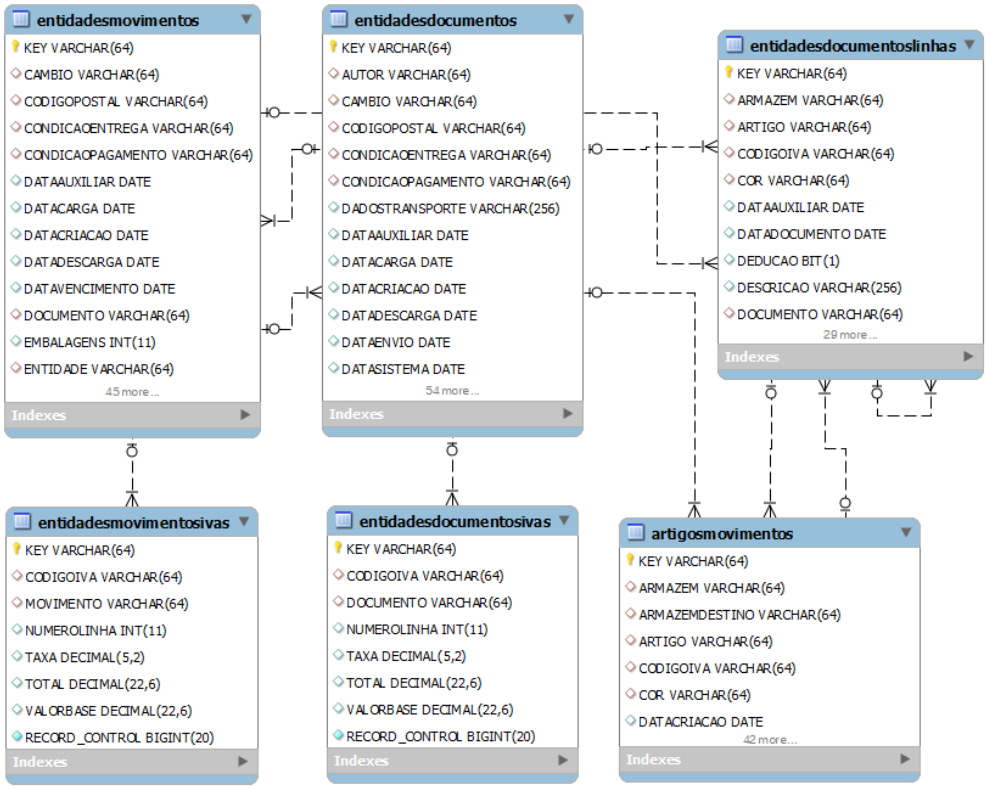

Figura 4.9. Diagrama Entidade-Relacionamento para Encomendas

É visível nos diferentes modelos de entidade-relacionamento que a operação de exportar dados de uma aplicação para outra implica a definição de diversos atributos em diferentes tabelas. O desafio é portanto compreender como estes devem ser preenchidos e que papel têm os dados das instâncias da plataforma e-commerce nesta operação.

4.2.2. Correspondência entre Entidades

Do ponto de vista da facturação de uma encomenda a discriminação das várias características do produto não é relevante. O Código do Imposto sobre o Valor Acrescentado dita que as facturas devem ser apresentadas, relativamente ao produto/serviço vendido, com os dados[1]:

- quantidade
- denominação usual dos bens transmitidos ou dos serviços prestados
- o preço, líquido de imposto, e os outros elementos incluídos no valor tributável
- as taxas aplicáveis
- o montante de imposto devido;

Os atributos apontados encontram-se em ambos os componentes, vendas e facturação, pela necessidade e importância que estes têm para o cliente antes e após a compra. Apesar de existir um mapeamento quase directo, mudando apenas a notação, de alguns atributos existem vários desafios na correspondência entre a entidade Produto, genericamente

[1] Código do IVA. Artigo 36º. Aditado pelo D.L. nº 197/2012, de 24 de Agosto, com entrada em vigor em 1 de Janeiro de 2013

falando, do Magento e Projecto Colibri. Para melhor compreensão os atributos do Projecto Colibri foram divididos em quatro categorias:

- **Atributos de correspondência.** Este atributo, ou atributos, são únicos no seu conjunto, o que significa que podem servir como chave de cada registo na base de dados e o ponto chave para fazer a correspondência entre os diferentes modelos na medida em que o distingue dos outros.
- **Atributos imediatos.** Estes atributos existem no modelo em consideração na plataforma Magento e são exportados a partir da instância do modelo.
- **Atributos calculáveis.** Os atributos não existem *per se* no modelo em consideração mas podem ser calculados ou transformados de forma preencher as necessidades do Projecto Colibri.
- **Atributos não presentes.** Os atributos são específicos do Projecto Colibri e não têm correspondência na plataforma e-commerce.

As entidades têm por norma um atributo único, a chave primária. Esta seria a solução candidata a fazer a correspondência entre, por exemplo, o produto com o `id = 104` (Magento) e o artigo com o `CODIGO = 104` (Projecto Colibri). No entanto foi identificado um problema claro com esta abordagem. O *id* de um registo é por norma um número inteiro auto-incrementado de forma a garantir que cada linha da coluna tem um *id* exclusivo (que não se vai repetir na coluna). Por cada inserção de um novo produto no Magento é gerado automaticamente um novo *id*. Este tipo de correspondência é muito sensível à ordem pela qual os registos foram adicionados e pode rapidamente levar a inconsistência na relação entre as entidades nas diferentes componentes. Além disso, na eventualidade do utilizador já ter introduzido anteriormente vários produtos no software de facturação teria que alterar manualmente cada produto de forma a fazer corresponder ao *id*. A solução encontrada foi o atributo SKU (Stock Keeping Unit) um valor indentificador único de um produto ou serviço. Este valor preserva a necessidade de ser distinto, sendo a sua introdução obrigatória em ambas as componentes, com a vantagem de não ser dependente do momento de inserção. A utilização do SKU é também uma prática importante para gestão do inventário.

De forma análoga para a entidade Cliente foi escolhido o atributo e-mail (a plataforma Magento não permite o registo de clientes com o mesmo e-mail). Um atributo alternativo para a correspondência do cliente é o Número de Identificação Fiscal, tendo em conta que o Magento não obriga a sua introdução e nesse caso clientes sem NIF não seriam exportados.

Outras soluções possíveis para correspondência:

- Chaves compostas. Dois clientes podem ter o mesmo nome, ou a mesma morada, mas pode ser considerado que não há dois clientes iguais com o mesmo nome e morada. Mesmo que seja idealizada uma chave composta única a existência de um só atributo único será preferível.
- Tabela de Mapeamento. As chaves que identificam as entidades podem não ser necessariamente as mesmas (e.g. e-mail na plataforma Magento e NIF no Projecto Colibri) ou o mesmo valor. Uma solução seria então a criação de uma tabela que fizesse a correspondência entre os atributos (Ver Figura 4.10).

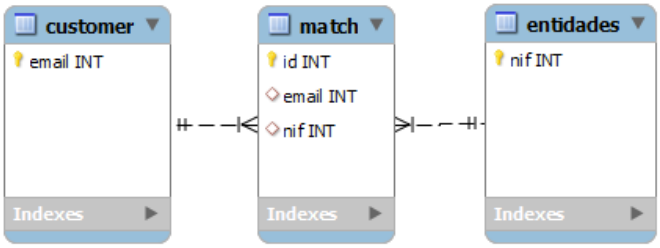

Figura 4.10. Tabela de Mapeamento entre Entidades com chaves diferentes

Esta solução implica no entanto a pesquisa em duas tabelas para obter a correspondência assim como a manutenção da tabela de mapeamento para novas entidades. Seria no entanto a opção que deveria ser tomada na ausência de chaves simples ou compostas.

4.2.3. Implementação dos Modelos

A criação das tabelas na base de dados é apenas o primeiro passo para a integração entre a plataforma e-commerce Magento e o Projecto Colibri. O ORM (Object Relational Mapping) do Magento tem um papel vital na forma como este interage com a base de dados sendo que todos os modelos que interagem com a base de dados herdam a classe Mage_Core_Model_Abstract, que por sua vez herda a classe Varien_Object. O carregamento ou armazenamento de dados passa a ser feito portanto através de objectos (ver Figura 4.11).

```
$product = Mage::getModel('catalog/product')->load(1);
$product->setPrice(100)->save();
```

Figura 4.11. Exemplo de operações sobre objectos pelo ORM do Magento

Os objectos do Magento definidos neste modelo suportam as funcionalidades básicas CRUD (Create, Read, Update e Delete) e têm os respectivos `getters` e `setters` para manipulação dos seus atributos.

- `$product = Mage:getModel('catalog/product')` – O método `getModel()` retorna uma instância da classe `DevelopDei_Colibri_Model_Product`.
- `$product->load($id)` – O método `load()` preenche a instância com os atributos armazenados na base de dados através da sua chave primária, ou de outro campo definido no segundo parâmetro (e.g. `$product->load(9.99, 'price')`).
- `$product->save()`– Insere um registo do novo modelo na base de dados ou actualiza se já existir.
- `$product->delete()`– Remove o registo do modelo na base de dados.

- `$product->getData()` – Devolve um dicionário (par chave/valor) dos campos do objecto. Pode também devolver apenas um campo específico (`e.g $product->getData('price')`).

- `$product->getOrigData()` – Devolve os dados do modelo como estava no momento em que o modelo foi inicialmente carregado.

- `$product->setData('price', 9.99)` – Define ou altera o valor do campo escolhido.

- `$product->addData($array)` – Preenche a instância com os valores definidos no array sobrepondo-se aos já definidos.

Para dar a conhecer então as entidades do Projecto Colibri foi necessário:

1. Criar classes/modelos para cada entidade.
2. Associar cada classe à respectiva tabela.

As classes são criadas na pasta Model do módulo. O Magento tem três classes básicas – *Model*, *Resource* e *Collection*. A classe *Model* é mais frequentemente usada e representa dados de um certo tipo, como Encomenda, de uma maneira agnóstica em relação à base de dados. A classe para a entidade encomendas (DevelopDei/Colibri/Model/Orders.php) foi inicialmente implementada com o código apresentado na Figura 4.12.

```
class DevelopDei_Colibri_Model_Order extends Mage_Core_Model_Abstract
{
   protected function _construct()
   {
      $this->_init('colibri/order');
   }
}
```

Figura 4.12. Criação de um novo modelo para Magento

Os modelos não contêm nenhum código que remeta para a conexão à base de dados. Em vez disso é usada outra classe: `modelResource` (ver Figura 4.13). A equipa do Magento explica esta abordagem como um objectivo em desacoplar a parte lógica do modelo do código. Assim a comunicação com a base de dados permite configurar diferentes recursos para comunicar com diferentes esquemas de base de dados e outras plataformas sem que os modelos sejam alterados. Cada modelo tem por isso por isso um recurso associado que é utilizado para fazer o carregamento de uma instância do modelo da base de dados. A classe recurso para a encomenda (DevelopDei/Colibri/Model/Resource/Order.php) identifica o modelo, o campo da base de dados que identifica de forma única cada registo. Além disso, no caso particular das entidades do Projecto Colibri, foi indicado também que a chave não é um valor auto-incrementável.

```
Class DevelopDei_Colibri_Model_Resource_Order
     extends Mage_Core_Model_Resource_Db_Abstract
{
    protected function _construct()
    {
        $this->_init('colibri/order', 'NUMEROAUXILIAR');
        $this->_isPkAutoIncrement = false;
    }
}
```

Figura 4.13. Criação de um novo recurso para Magento

Existem situações em que é necessário retornar mais do que um registo da base de dados e não apenas uma instância em específico. Cada tipo de modelo tem uma colecção associada que permite retornar múltiplos registos (ver Figura 4.14). As colecções implementam as interfaces `IteratorAggregate` e `Countable` do PHP o que significa que permitem que os resultados sejam percorridos e que se saiba quantos registos foram encontrados. As colecções encontram-se dentro da mesma pasta:

(DevelopDei/Colibri/Model/Resource/Order/Collection.php).

```
class DevelopDei_Colibri_Model_Resource_Order_Collection
     extends Mage_Core_Model_Resource_Db_Collection_Abstract
{
    protected function _construct()
    {
       $this->_init('colibri/order');
    }
}
```

Figura 4.14. Criação de uma nova colecção de modelos para Magento

O último passo é a configuração do ficheiro `configs.xml` onde foram declarados os modelos e as tabelas a que estão associados (ver Figura 4.15).

```
<globals>
   <models>
      <colibri>
         <class>DevelopDei_Colibri_Model</class>
         <resourceModel>colibri_resource</resourceModel>
      </colibri>
      <entities>
         <client>
            <table>entidades</table>
         </client>
         <seller>
            <table>vendedores</table>
         </seller>
         <postcode>
            <table>codigospostais</table>
         </postcode>
         …
      </entities>
   </models>
</globals>
```

Figura 4.15. Declaração dos novos modelos e respectivas tabelas no Magento

Com esta configuração o Magento assume que as tabelas se encontram na mesma base de dados que a definida inicialmente para a plataforma e-commerce. Ter as tabelas todas na mesma base de dados é uma abordagem que pode ser feita na instalação dos diferentes componentes e foi a tomada numa fase inicial do desenvolvimento desta integração. No entanto no sentido de estender as capacidades de integração e caminhar para uma solução mais próximo de um ambiente de produção foram definidas diferentes bases de dados para cada módulo através da configuração de um novo `resource` (ver Figura 4.16). Desta forma não se força uma ligação estrita entre as várias aplicações e a mesma base de dados. Cada base de dados pode ser configurada sem as alterações terem impacto nas demais e abre caminho a que possam estar em locais diferentes não restringindo o uso da aplicações a um só computador.

```
<globals>
   <resources>
         <colibri_database>
            <connection>
               <host><![CDATA[127.0.0.1]]></host>
               <username><![CDATA[dev]]></username>
               <password><![CDATA[estagio12345]]></password>
               <dbname><![CDATA[colibri]]></dbname>
               <initStatements><![CDATA[SET NAMES utf8]]></initStatements>
               <model><![CDATA[mysql4]]></model>
               <type><![CDATA[pdo_mysql]]></type>
               <pdoType><![CDATA[]]></pdoType>
               <active>1</active>
            </connection>
         </colibri_database>
         <colibri_write>
            <connection>
               <use>colibri_database</use>
            </connection>
         </colibri_write>
         <colibri_read>
            <connection>
               <use>colibri_database</use>
            </connection>
         </colibri_read>
         <colibri_setup>
            <connection>
               <use>colibri_database</use>
            </connection>
         </colibri_setup>
      </resources>
</globals>
```

Figura 4.16. Configuração de recursos no Magento para diferentes bases de dados

Os modelos que gerem clientes, produtos e encomendas foram desenvolvidos de forma a terem uma interface composta por três métodos essenciais para o processo de exportação:

- `setObject()`: o modelo recebe uma instância do modelo da plataforma Magento.
- `import()`: os atributos relevantes são copiados da instância recebida sendo os atributos específicos do software Projecto Colibri preenchidos com valores padrão. Os valores padrão foram configurados de acordo com o que foi observado ao adicionar novos clientes, artigos e encomendas no Projecto Colibri.
- `export()`: a instância e as suas dependências são armazenadas na base de dados.

Foi feita uma separação clara entre a inicialização/população do modelo (import) e o seu armazenamento (export) de forma a permitir que o objecto possa ser manipulado num estado já preenchido. Neste estado podemos por exemplo definir ou alterar atributos de acordo com outros parâmetros enviados com o pedido (e.g. a submissão de um formulário).

O diagrama da Figura 4.17 representa um pedido do utilizador que resulta na criação/ alteração de um dos modelos sob consideração.

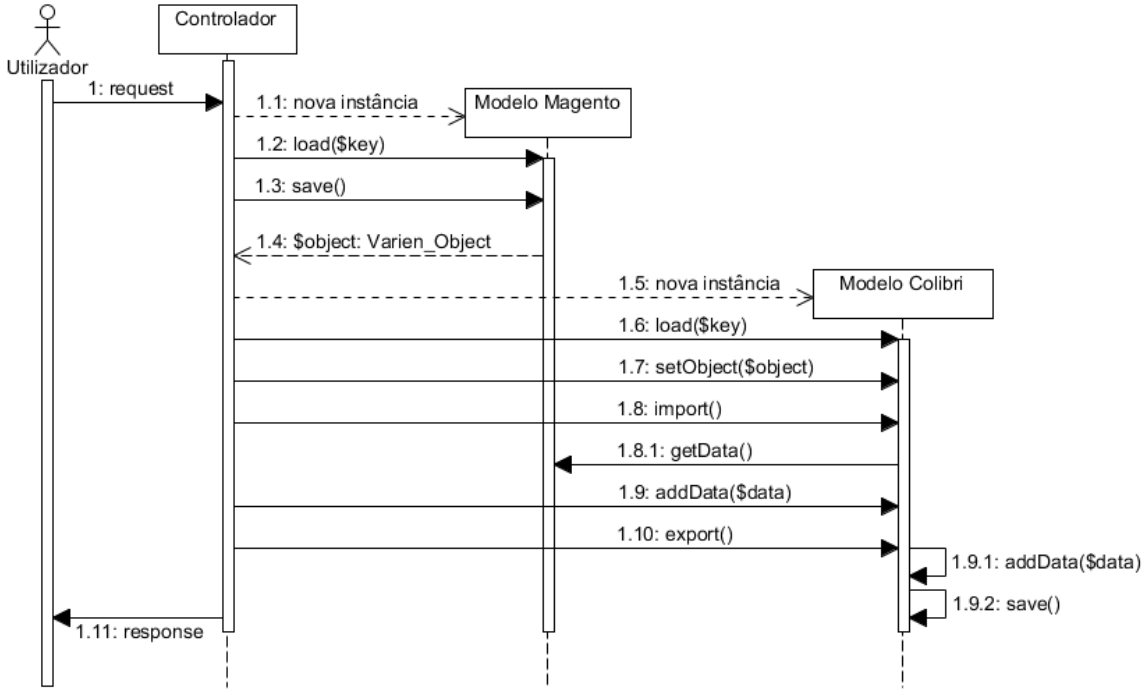

Figura 4.17. Diagrama de Sequência da sincronização entre Entidades

O sistema começa por criar ou alterar a instância proprietária da plataforma e-commerce. É de seguida carregada a instância correspondente do Projecto Colibri, se existente, através da chave que faz a correspondência entre as entidades. A instância do Projecto Colibri recebe a instância proprietária assim como dados que possam ter sido passados ao controlador através de um POST no pedido (ver secção 4.2.9. Interface). Antes de ser guardada a instância percorre a instância proprietária recolhendo atributos comuns ou calculando-os se necessário e na ausência de dados vitais para o software de facturação são adicionados os valores padrão observados na interacção com o software de facturação.

4.2.4. Validação e integridade dos dados

A exportação dos dados é uma operação que implica mais do que uma chamada à base de dados. Não só os dados específicos de um produto, cliente ou encomenda não se encontram apenas numa tabela como existem chaves estrangeiras com restrições que têm que ser verificados. Tomando o 'Produto' como exemplo, existem outros dois modelos que lhe estão intrinsecamente ligados (ver Figura 4.6) e que armazenam características como unidades de medida, quantidade e peso (artigosunidades) e preço, preço com IVA e margem de lucro (artigosprecos). O registo de um novo produto faz apenas sentido para o software de facturação se o registo for efectuado nos três modelos. Nesse sentido em vez de gravar iterativamente cada instância de 'artigos', 'artigosunidades' e 'artigosprecos' através da função `save()` foi definido que todos os registos devem ser inseridos numa só transacção. Daqui resulta que o produto é inserido na sua totalidade ou não é inserido de todo, garantindo a integridade dos dados. Isto é conseguido através do Modelo Transaction (resource_transaction) do Magento (ver Figura 4.18).

```
$units  = $this->getUnits();
$prices = $this->getPrices();

try{
   Mage::getModel('core/resource_transaction')
                  ->addObject($this)
                  ->addObject($units)
                  ->addObject($prices)
                  ->save();
   Mage::log('[accounting/product] Saved successfully (single transaction)' ,
             Zend_Log::DEBUG);

} catch(Exception $e) {

   Mage::log('[accounting/product] Failed saving. Rolledback. '
           . $e->getMessage() , Zend_Log::DEBUG);
  throw $e;
}
```

Figura 4.18. Transacção única para múltiplas operações de armazenamento

Este modelo percorre os objectos e ao gravar define que os objectos na sua posse serão gravados numa transacção. Na ocorrência de algum erro as alterações já feitas serão revertidas, como se pode ver na função `save()` deste modelo (ver Figura 4.19)

```
public function save()
{
   $this->_startTransaction();
   $commit = true;
   $errors = array();

   foreach ($this->_objects as $object) {
      try {
         $object->save();
      }
      catch (Exception $e) {
         $commit = false;
         $errors[] = $e->getMessage();
      }
   }

   if ($commit) {
      $this->_commitTransaction();
   }
   else {
      $this->_rollbackTransaction();
      Mage::throwException(join("\n", $errors));
   }

   return $this;
}
```

Figura 4.19. Implementação de transacções no Magento

Os impostos aplicados sobre um produto não encontram também a mesma representação entre as duas componentes. Se o produto for definido na plataforma e-commerce com uma outra taxa que não as reconhecidas de origem pelo software de facturação o seu valor é verificado e é inserido um novo registo no modelo respeitante à tabela 'codigosiva' do Projecto Colibri. Este passo garante que a chave estrangeira do nosso modelo produto referencia correctamente o registo.

Outra consideração importante diz respeito aos atributos que ficam por preencher pois não estão presentes na instância importada do Magento ou que não foram preenchidos, como no caso de um novo cliente que ainda não inseriu a sua morada e número de identificação fiscal no sistema. Quando observado no Projecto Colibri o registo de um novo cliente tem já valores padrão para estes atributos, como o NIF `999999990`, um NIF válido para o sistema e que está associado ao consumidor final (ver Figura 4.20).

```
protected function _getAddressData() {

   $address = array('CODIGOPOSTAL' => '1000-000',
                    'LOCALIDADE'    => 'Indefinido',
                    'NIF'           => '999999990');
   if($this->_address) {

      $address['CODIGOPOSTAL'] = $this->_address->getPostcode();
      $address['LOCALIDADE'] = $this->_address->getCity();
      $address['MORADA'] = $this->_address->getStreetFull();
      $address['TELEFONE'] = $this->_address->getTelephone();
      $address['TELEMOVEL'] = $this->_address->getTelephone();

      $vat = $this->_address->getVatId();
      if(!empty($vat)) {
         $address['NIF'] = $vat;
      }
   }

   return $address;
}
```

Figura 4.20. Atributos por omissão na entidade Cliente do Colibri

É criado à partida um *array* com os atributos por defeito, aceites pelo Projecto Colibri. Se existir uma morada anteriormente importada para o nosso modelo então esses atributos são sobrepostos. De notar que existindo uma morada definida para um cliente no Magento os atributos Localidade e Código Postal estão necessariamente preenchidos. No entanto o Número de Identificação Fiscal pode não ter sido preenchido. Para evitar que um valor em branco se sobreponha é feita uma verificação prévia.

Existem também atributos que precisam ser calculados ou transformados. Uma encomenda está associada a um ou mais produtos e respectivas quantidades , uma relação 1:N sendo N o número de produtos encomendados. O Projecto Colibri segue o mesmo modelo mas guarda também numa tabela os subtotais associados à encomenda por tipo de taxa aplicada. Pode ser visto um exemplo desta relação na Tabelas 4.4 e 4.5.

Encomenda	Produto	Quantidade	Preço total C/IVA (€)	IVA (%)
5	Produto 1	3	10,42	23
5	Produto 2	1	5,68	6
5	Produto 3	1	1,19	13
5	Produto 4	1	2,39	23

Tabela 4.4. Exemplo de lista de produtos de uma encomenda

Encomenda	IVA (%)	Total (€)
5	6	5,68
5	13	1,19
5	23	12,81

Tabela 4.5. Exemplo da Tabela de IVA associada à encomenda

A abordagem tomada para agrupar estes valores foi através de uma query SQL. Uma vez preenchida a tabela com os produtos da encomenda em consideração, é efectuado um pedido de todos os registos associados à encomenda agregados pela taxa que lhes é aplicada (ver Figura 4.21). A opção tomada em comparação com o cálculo do somatório no próprio objecto PHP é justificada não só pela simplicidade mas sobretudo pela garantia de que a agregação de valores corresponde exactamente aos dados que já se encontram efectivamente inseridos na base de dados.

```
$taxes = Mage::getModel('accounting/orderitem')->getCollection();
$taxes->getSelect()
        ->columns('CODIGOIVA')
        ->columns('SUM(PRECO) AS base')
        ->columns('SUM(PRECOIVA) AS total')
        ->where("DOCUMENTOENTIDADES = ? ", $this->_order_key)
        ->group('CODIGOIVA');

// SELECT CODIGOIVA, SUM(PRECO) AS base, SUM(PRECOIVA) AS total
// FROM artigosmovimento
// WHERE DOCUMENTOENTIDADES = { chave da encomenda }
// GROUP BY CODIGOIVA
```

Figura 4.21. Cálculo do valor total da encomenda para a aplicação de diferentes impostos

Existe ainda a preocupação de que os dados façam sentido não apenas do ponto de vista do modelo relacional e correcta leitura por parte do Projecto Colibri mas também que se mantenham válidos no contexto das regras de facturação em vigor. Uma dessas exigências é a assinatura digital do documento. A assinatura digital do documento é efectuada através de um processo matemático que calcula a hash da mensagem (através de atributos como data do documento, código e série do documento, valor total da encomenda e a hash do documento anterior) e a cifra com a chave privada do emissor. Qualquer receptor (normalmente o software que permite visualizar uma mensagem/documento assinado digitalmente) pode recalcular o hash da mensagem, decifrar a assinatura com a chave pública do emissor e, se as duas hash forem iguais, o receptor tem a garantia que foi cifrada com a chave privada correspondente e que o documento não foi alterado. A chave privada é

portanto do conhecimento exclusivo do produtor de sofware e, em consequência, o atributo hash não pode ser calculado.

Outra exigência da Autoridade Tributária é a associação de uma série à factura. Todos os tipos de documentos deverão ser emitidos cronologicamente em uma ou mais séries devendo ser datados e numerados de forma progressiva e contínua, dentro de cada série[1]. A solução encontrada foi a criação de uma nova série de documentos para encomendas. Esta série (`'CEN:MAG'`) segue o mesmo template do documento da série original (`'CEN:2014'`), implicando que quer os campos a serem preenchidos quer o documento emitido permanecem iguais. Esta série serve como uma série auxiliar e não é legítima para a emissão de documentos. No entanto, o registo de uma encomenda associada a esta série mantém uma propriedade transversal a todos os registos que é a sua duplicação. Ao duplicarmos o documento gerado pela plataforma e-commerce, sem *hash*, poderemos então seleccionar uma série válida e uma *hash* será gerada para o duplicado. Esta série é adicionada automaticamente no Projecto Colibri na instalação do módulo através de um *script* na pasta do módulo DevelopDei\Colibri\data\colibri_setup.

4.2.5. Eventos relevantes para o processo

O Magento é uma framework orientada a objectos permitindo assim a sua customização através de subclasses (herança) dos seus modelos e blocos. Além disso a sua arquitectura suporta também programação orientada a eventos o que possibilita uma mais clara separação do código customizado do código *core* da plataforma.

Os eventos são *triggers* que informam o sistema que deve olhar para os seus subscritores (observadores) e notificá-los automaticamente das alterações no seu estado. Na perspectiva dos processos de negócio os eventos alertam os subscritores acerca de um problema ou uma oportunidade no fluxo de processo em questão. Os serviços à escuta são entidades independentes dos objectos que geram esses eventos e que podem ser adicionados ou removidos à lista de subscritores sem afectar os objectos.

No Magento um evento é accionado especificando um código descritivo para o evento, e.g `catalog_product_save`, sendo opcionalmente enviados dados juntamente com o evento (ver Figura 4.22)

```
Mage::dispatchEvent('event_code', array('myObject' => $myObject))

Mage::dispatchEvent('admin_session_user_login_success',
        array('user'=>$user));
```

Figura 4.22. Exemplo de sinalização de eventos do Magento

O sistema acciona implicitamente vários eventos através da classe base de todos os seus objectos Mage_Core_Model_Abstract. Para o caso em que uma nova instância é guardada, são invocados dois métodos, `_beforeSave` e `_afterSave` . Estes dois métodos accionam um

[1] Despacho n.º 8632/2014, publicado no DR II.ª série, n.º 126, de 3 de julho

evento genérico e um específico cujo código descriptivo é construído de acordo com o objecto que está a ser guardado (ver Figura 4.23)

```
protected function _beforeSave()
{
       Mage::dispatchEvent('model_save_before', array('object'=>$this));
       Mage::dispatchEvent($this->_eventPrefix.'_save_before', array($this-
>_eventObject=>$this));
         return $this;
}
```

Figura 4.23. Sinalização de eventos do Magento

O Magento oferece assim uma série de eventos padrão que cobrem os processos de leitura, escrita e remoção de modelos. Os nomes desses eventos terminam com:

```
_load_before
_load_after
_save_before
_save_after
_save_commit_after
_delete_before
_delete_after
_delete_commit_after
```

Os eventos `_commit_after` são accionados apenas após finalizada a transacção no sistema de gestão de base de dados.

Cada classe tem definidas as propriedades `_eventPrefix`, que identifica a entidade associada ao evento e que gera assim uma descrição única para o evento, e `_eventObject` que ajuda o observador a fazer a chamada da instância do objecto incluída no evento. No caso da classe `Mage_Catalog_Model_Product` veríamos como apresentado na Figura 4.24.

```
protected $_eventPrefix      = 'catalog_product';
protected $_eventObject      = 'product';
```

Figura 4.24. Prefixos de eventos no Magento

Esta invocação genérica permite então escutar eventos específicos como "antes de criar uma encomenda" ou "depois de guardar os dados de um cliente". No contexto da integração com o Projecto Colibri é indispensável estar consciente da criação de novos clientes, produtos e encomendas, assim como qualquer de alteração que possa ser feita, de modo a replicar a mesma acção no software de facturação. Os eventos subscritos são por isso:

```
catalog_product_save_commit_after
catalog_product_delete_commit_after
customer_save_commit_after
customer_address_save_commit_after
~~sales_order_place_after~~
sales_order_status_change
```

Uma plataforma e-commerce pode receber por dia várias encomendas sendo que a operação de guardar cada encomenda é complexa e consome mais recursos do que as restantes operações em consideração. A acção de finalizar uma encomenda é também um ponto vital do negócio e deve ser executada rapidamente e resistente a falhas. No sentido de proteger essas condições a replicação da encomenda foi transferida para a área de administração. A encomenda é exportada para o Projecto Colibri apenas mediante interacção do gestor da loja, através da mudança de estado da encomenda para `PROCESSING`.

Desta forma, o Projecto Colibri recebe apenas dados de encomendas já validadas para que se possa proceder à respectiva facturação, sendo minorados os casos em que esta precisará ser alterada. De forma a exportar os dados apenas na alteração de estado foi introduzido um novo evento através da herança da classe de encomendas `Mage_Sales_Model_Order` e sobreposição da função responsável por alterar o estado (ver Figura 4.25). A abordagem é por isso diferente das outras entidades que exportam ao gravar. Neste caso o evento a captado inicialmente, `sales_order_place_after`, foi substituído pelo novo evento `sales_order_status_change`.

```
class DevelopDei_Colibri_Model_Salesorder extends Mage_Sales_Model_Order
{
   protected function _setState($state, $status = false, $comment = '',
                                $isCustomerNotified = null,
                                $shouldProtectState = false){

   Mage::dispatchEvent('sales_order_status_change',
                        array('order' => $this,
                        'state' => $state,
                        'status' => $status,
                        'comment' => $comment,
                        'isCustomerNotified' => $isCustomerNotified));

   return parent::_setState($state, $status, $comment,
                            $isCustomerNotified, $shouldProtectState);
   }
}
```

Figura 4.25. Implementação de um novo evento para alteração do estado de encomendas no Magento

4.2.6. Ligar uma função a um evento

Os observadores são registados no ficheiro de configuração, `config.xml`, com a seguinte estrutura apresentada na Figura 4.26.

```
<globals>
   <events>
      <catalog_product_save_commit_after>
            <observers>
               <product_save_commit_after_observer>
                  <type>singleton</type>
                  <class>colibri/observer</class>
                  <method>catalog_product_save_commit_after</method>
               </product_save_commit_after_observer>
            </observers>
         </catalog_product_save_commit_after>
   </events>
</globals>
```

Figura 4.26. Configuração de um observador no Magento

O sistema capta a lista de observadores através do nó `<events>`. Este nó pode estar inserido em `<global>`, `<frontend>` ou `<adminhtml>` dependendo da área que queremos restringir para a execução do observador (e.g. para a exportação de encomendas queremos apenas que o evento seja accionado na área de administração).

O nome do evento a captar é registado dentro do nó `<events>` e o nome que identifica o observador deve ser definido dentro do nó `<observers>`.

O nó `<class>` e `<method>` contêm, respectivamente, uma referência para a classe do observador (`Colibri/Model/Observer.php`) e o nome da função a ser invocada – `catalog_product_save_commit_after()`. Apesar de não estar em conformidade com a notação do PHP, o nome da função está definido de acordo com o nome do evento para facilitar a associação entre ambos.

O nó `<type>` define o âmbito do observador. Este nó é opcional e, não sendo especificado, instanciará o observador como `singleton`, significando que as propriedades do observador se mantêm enquanto o processamento do pedido não terminar. De outra forma para instanciar vários observadores deverá ser definido como `model`.

4.2.7. Implementação do Observador

As funções definidas no observador recebem um parâmetro `$observer` que contem os objectos passados para o evento. É possível a partir daqui ler e manipular as instâncias de clientes, produtos ou encomendas sendo que no contexto de integração com o Projecto Colibri existe apenas interesse em ler informação garantindo que o comportamento indesejado do módulo não terá influência na integridade dos dados do sistema e-commerce.

Na Figura 4.27 pode ver-se implementada a função que irá guardar os dados do cliente para o software de facturação.

```
class DevelopDei_Colibri_Model_Observer {

   public function catalog_product_save_commit_after($observer) {

      $this->_log($observer);
      try {

         $product    = $observer->getProduct();
         $sku        = ($product->getOrigData('sku') ?
                         $product->getOrigData('sku') : $product->getSku());

         $post       = Mage::app()->getRequest()->getPost();

         $item       = Mage::getModel('colibri/product')
                           ->load($sku)
                           ->setProduct($product)
                           ->import()
                           ->addData($post)
                           ->export();

      }catch(Exception $e) {
         …
      }

   return $this;
}
```

Figura 4.27. Implementação de um observador para Produtos no Magento

A função `_log($observer)` é uma função de *debug* e regista no `system.log` o nome do evento capturado. Esta função foi implementada com vista a identificar se o evento foi accionado e se o observador foi devidamente configurado (ver Figura 4.28).

```
protected function _log($observer) {

   Mage::log('[colibri/observer] ' . $observer->getEvent()->getName()
                                        . ' called.');
}
```

Figura 4.28. Função de *debug* de eventos accionados

As restantes instrucções são incluídas num bloco `try/catch` para que o comportamento imprevisto da implementação não afecte o fluxo do pedido, i.e. caso o módulo não tenha sucesso em exportar os dados este não afectará o restante comportamento esperado da plataforma e-commerce.

No caso descrito o objecto que accionou o evento foi um produto. O observador permite aceder ao objecto através da função genérica `get<_eventObject >`.

O campo que permite fazer a correspondência com um produto já exportado para o Projecto Colibri é o SKU (Stock Keeping Unit). Apesar de o SKU ser um valor único definido para cada produto este pode ter sido um dos campos alterados no pedido que causou a alteração do objecto o que levanta um problema de correspondência.

É invocado por isso o método `getOrigData() que` retorna dados do objecto no momento em que ele foi inicializado. Se o método `getOrigData('sku')` retornar um valor então o objecto já tinha sido exportado para o Projecto Colibri e o SKU a corresponder é aquele que garantidamente não foi alterado após inicialização.

```
$sku        = ($product->getOrigData('sku') ?
                $product->getOrigData('sku') : $product->getSku());
```

Figura 4.29. Correspondência entre entidades

Se o método `getOrigData('sku')` não retornar um valor significa que é um novo produto criado no Magento (quando foi inicializado não tinha qualquer valor) e o SKU a corresponder será o actual.

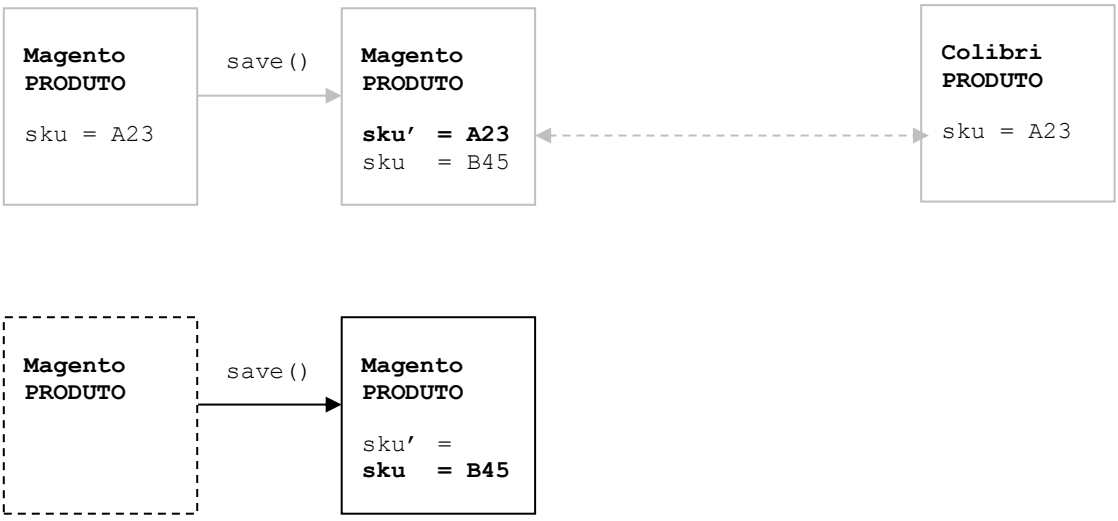

Figura 4.30. Correspondência entre entidades usando o valor original

Recolhido o valor do SKU é instanciado o modelo para posterior armazenamento no Projecto Colibri.

```
$item       = Mage::getModel('colibri/product')->load($sku);
```

Figura 4.31. Carregamento de um produto pelo atributo SKU

O objecto do tipo produto do Magento é passado para o modelo através da função `setProduct( )` para que os dados relevantes para o Projecto Colibri possam ser copiados através da função `import()`. Antes de a instância ser inserida ou actualizada na base de dados através da função `export()` esta recebe ainda os dados dos parâmetros que podem estar incluídos no pedido. Estes dados resultam da submissão do formulário da zona de administração onde têm origem as operações que produzem alterações nos produtos como adição de novos produtos, gestão de inventário e preços Estes parâmetros irão substituir os definidos por padrão para as novas instâncias.

Os restantes observadores seguem a mesma configuração e estrutura variando nalguns detalhes:

`catalog_product_delete_commit_after`

O observador que capta a remoção de um produto irá apenas alterar o seu estado para 'Inactivo' no Projecto Colibri.

```
$product = $observer->getProduct();
$sku = $product->getOrigData('sku');

Mage::getModel('accounting/product')
        ->load($sku)
        ->disable()
        ->save();
```

Figura 4.32. Implementação do observador para produtos apagados no Magento

`customer_save_commit_after`

```
public function customer_save_commit_after($observer) {

   try {
      $this->_log($observer);

      $customer  = $observer->getCustomer();
      $email     = $customer->getOrigData('email') ?
                   $customer->getOrigData('email') : $customer->getEmail();

      $post = Mage::app()->getRequest()->getPost();

      $client    = Mage::getModel('accounting/client')
                         ->load($email,'EMAIL')
                         ->setCustomer($customer)
                         ->import()
                         ->addData($post)
                         ->save();
   } catch(Exception $e){
      …
   }

   return $this;
}
```

Figura 4.33. Implementação do observador para alteração de clientes no Magento

`customer_address_save_commit_after`

Enquanto na plataforma Magento um cliente pode ter mais do que uma morada associada, podendo esta estar definida como morada de entrega ou facturação (ou ambas), para o Projecto Colibri é dada relevância à morada de facturação. Isto significa que enquanto no primeiro caso existem dois modelos distintos, no segundo a morada faz parte dos dados do modelo único de cliente. Como o sistema alerta primeiro para o armazenamento da morada é preciso garantir que já existe um cliente a quem associar a informação.

```
public function customer_address_save_commit_after($observer) {

   try {

      this->_log($observer);

      $address   = $observer->getCustomerAddress();
      $email     = $address->getCustomer()->getEmail();
      $client    = Mage::getModel('colibri/client')->load($email,'EMAIL');

      if(!$client->isObjectNew()) {

         $client->setAddress($address)
                 ->export()
      }
   }catch(Exception $e){
      …
   }

   return $this;
}
```

Figura 4.34. Implementação do observador para alteração de moradas no Magento

`sales_order_status_change`

```
public function sales_order_status_change($observer) {

   $this->_log($observer);
   $state = $observer->getEvent()->getState();

   if($state == Mage_Sales_Model_Order::STATE_PROCESSING) {

      $sales_order = $observer->getEvent()->getOrder();
      $post = Mage::app()->getRequest()->getPost();

      try {
         $order = Mage::getModel('accounting/order')
                         ->load($sales_order->getRealOrderId())
                         ->setOrder($sales_order)
                         ->import()
                         ->addData($post)
                         ->export();
   }catch(Exception $e){
         …
      }
   }
}
```

Figura 4.35. Implementação do observador para encomendas a ser processadas no Magento

Na situação particular das encomendas não existe interesse em aceder ao valor original do atributo que faz a correspondência pois este não pode ser alterado na instância do Magento.

4.2.8. *Debug* e Registo de Eventos

Durante a fase de desenvolvimento pode ser necessário verificar que eventos foram accionados durante um determinado pedido ao sistema. Não só é mais fácil compreender se o evento sob consideração foi accionado como permite compreender que outros eventos do Magento existem e como podem ser relevantes para as nossas necessidades. Para ter acesso à lista de eventos será necessário alterar o ficheiro `app/Mage.php` e adicionar a seguinte linha de código ao início da função `dispatchEvent()`, responsável por accionar os eventos.

```
public static function dispatchEvent($name, array $data = array())
{
   Mage::log($name, null, 'events.log', true);
   Varien_Profiler::start('DISPATCH EVENT:'.$name);
   $result = self::app()->dispatchEvent($name, $data);
   Varien_Profiler::stop('DISPATCH EVENT:'.$name);
   return $result;
}
```

Figura 4.36. Debug de eventos accionados pelo Magento

O resultado esperado ao abrir o ficheiro `var/log/events.log` será o seguinte:

```
2014-06-15T19:59:45+00:00 DEBUG (7): customer_address_save_commit_after
2014-06-15T19:59:45+00:00 DEBUG (7): model_load_before
2014-06-15T19:59:45+00:00 DEBUG (7): core_abstract_load_before
2014-06-15T19:59:45+00:00 DEBUG (7): resource_get_tablename
2014-06-15T19:59:45+00:00 DEBUG (7): model_load_after
2014-06-15T19:59:45+00:00 DEBUG (7): core_abstract_load_after
2014-06-15T19:59:45+00:00 DEBUG (7): model_save_commit_after
2014-06-15T19:59:45+00:00 DEBUG (7): customer_save_commit_after
```

4.2.9. Interface

A integração foi implementada tendo em vista a automatização dos processos que sincronizam os objectos entre as componentes de vendas e facturação minimizando o custo acrescido para o utilizador. Apesar da integração na plataforma Magento a interacção do utilizador com os objectos será feita através da interface da plataforma e-commerce ou através da interface do software de facturação, dependendo de haver ou não necessidade de alterar os valores por omissão ao exportar a instância. No contexto da integração é importante dar alguma liberdade ao utilizador para alterar alguns passos implícitos no processo de exportação, tornando a integração configurável, de uma maneira que seja simples para o utilizador.

Nesse sentido foram criados formulários para cada um dos modelos: clientes, produtos e encomendas. Estes formulários foram incorporados no painel de administração da plataforma Magento na página de edição de cada modelo. Para simplificar a experiência do utilizador com a interface, os campos foram dispostos de forma semelhante àquela encontrada no Projecto Colibri (Figura Figura 4.37 e Figura 4.38).

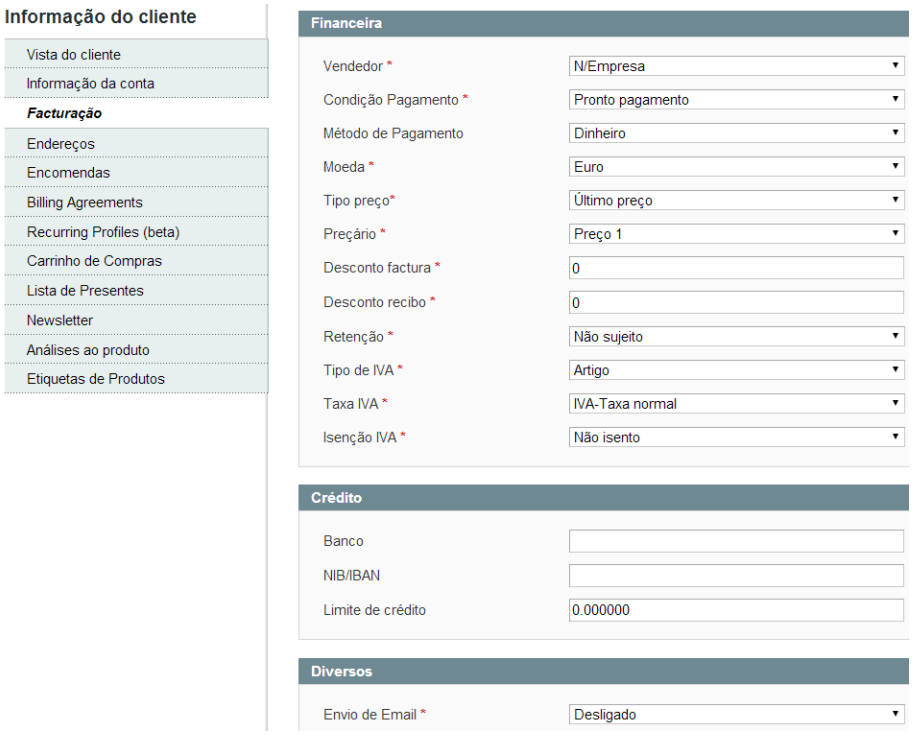

Figura 4.37. Formulário de Edição de Cliente na plataforma Magento

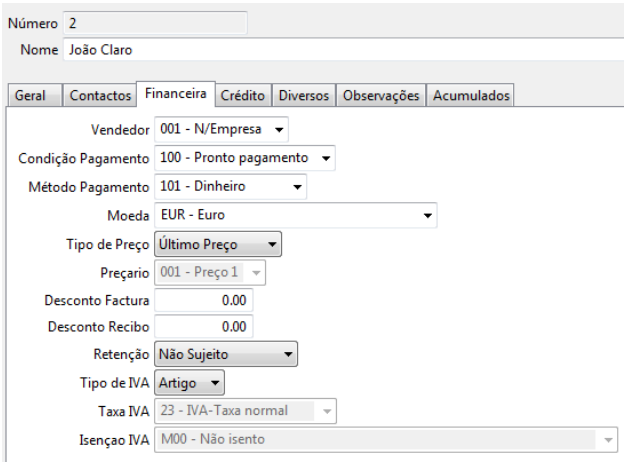

Figura 4.38. Formulário de Edição de Cliente no Projecto Colibri

Antes da criação dos formulários foi preciso indicar ao sistema que há uma alteração nos blocos dos separadores (*tab*) da página de edição de cada modelo (ver Figura 4.39). Esta alteração passa por acrescentar mais uma tab ficando o conteúdo do separador por definir. O sistema é então direccionado para carregar o bloco gerado pela classe criada com a customização desejada. Esta classe herda a classe do bloco que normalmente seria apresentado mas adiciona-lhe um novo separador (ver Figura 4.40). A este novo separador são adicionados vários campos de *input* de diferentes tipos e carregados dinamicamente com os valores definidos e valores possíveis (ver Figura 4.41).

```
<global>
   <blocks>
      <accounting>
         <class>DevelopDei_Accounting_Block</class>
      </accounting>
      <adminhtml>
         <rewrite>
            <catalog_product_edit_tabs>
               DevelopDei_Accounting_Block_Adminhtml_Catalog_Product_Edit_Tabs
            </catalog_product_edit_tabs>
            <customer_edit_tabs>
               DevelopDei_Accounting_Block_Adminhtml_Customer_Edit_Tabs
            </customer_edit_tabs>
         </rewrite>
      </adminhtml>
   </blocks>
</global>
```

Figura 4.39. Sobreposição de conteúdo na área de administração do Magento

```
class DevelopDei_Colibri_Block_Adminhtml_Catalog_Product_Edit_Tabs
      extends Mage_Adminhtml_Block_Catalog_Product_Edit_Tabs
{
   private $parent;

   protected function _prepareLayout()
   {
      $this->parent = parent::_prepareLayout();
      $this->addTab('accounting', array(
            'label'     => Mage::helper('catalog')->__('Facturação'),
            'content'   => $this->getLayout()
            ->createBlock('accounting/adminhtml_catalog_product_accounting',
                'category.product.grid')->toHtml(),
      ));
      return $this->parent;
   }
}
```

Figura 4.40. Adição de um novo separador para o formulário de produtos

```
class Developdei_Colibri_Block_Adminhtml_Customer_Accounting
     extends Mage_Adminhtml_Block_Widget_Form
{
   protected function _prepareForm()
   {
       $form = new Varien_Data_Form();
       $this->setForm($form);

       $this->_addFields();

       return parent::_prepareForm();
   }

   protected function _addFields()
   {
      $form = $this->getForm();
      $customer = Mage::registry('current_customer');
      $client = Mage::getModel("accounting/client")
                     ->load($customer->getEmail(), "EMAIL");

      $fieldset = $form->addFieldset('colibri',
           array('legend'=>Mage::helper('customer')->__('Contactos'))
      );

      $fieldset->addField('TELEFONE', 'text',array(
         'label'     => $this->__('Telefone'),
         'required'  => false,
         'name'      => 'TELEFONE',
         'value'    => $client->getData("TELEFONE"),
      ));

      $fieldset->addField('TELEFONE2', 'text',array(
         'label'     => $this->__('Telefone 2'),
         'required'  => false,
         'name'      => 'TELEFONE2',
         'value'    => $client->getData("TELEFONE2"),
      ));
```

Figura 4.41. Exemplo da implementação dos campos a apresentar no formulário

4.3. Integração Magento e GnuCash

A integração com o software GnuCash passou também ela pela criação de um novo módulo a ser adicionado à plataforma e-commerce, Magento, pelo utilizador. A abordagem tomada para este módulo segue o mesmo processo de captação de eventos específicos aquando alteração dos modelos do Magento na qual estamos interessados. A descrição deste módulo passa pela descrição de casos específicos de implementação.

4.3.1. Modelo de Dados

O modelo de dados do GnuCash é razoavelmente simples, o que faz sentido tendo em consideração que foi desenvolvido para ser uma solução mais genérica com poucas imposições ou funcionalidades demasiado específicas e que pode ser alterado de forma a preencher necessidades particulares. Uma característica do GnuCash é a ausência de uma ligação definida entre as entidades, como se pode ver no Diagrama de Entidade-Relacionamento.

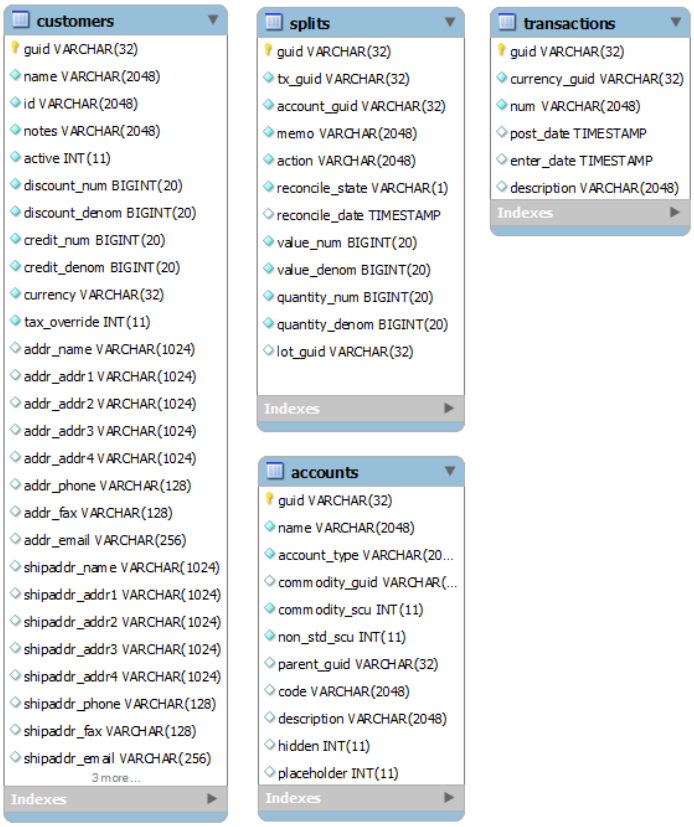

Figura 4.42. Modelo de Entidade Relacionamento do GnuCash

Uma característica distintiva nestes modelos é a geração da chave-primária `guid`. O termo GUID refere-se normalmente às várias implementações do padrão UUID (GUID structure (Windows)), sendo que todos os GUIDs são assumidos como únicos. No entanto, não se pode dizer que o são porque não há nenhum mecanismo para fazer cumprir essa singularidade. GUIDs são normalmente armazenados como valores de 128 bits e apresentados como 32 dígitos hexadecimais. O número total de GUIDs únicos é 2^{122}. Este número é tão grande que a probabilidade de o mesmo número a ser gerado aleatoriamente por duas vezes é negligenciável.

No PHP podemos gerar identificadores únicos através da função `uniqid()`. Esta função retorna um identificador único prefixado baseado no tempo actual em milionésimos de segundo. O prefixo pode ser usado para casos em que são gerados identificadores em vários servidores em simultâneo podendo nesse caso ocorrer que sejam gerados identificadores no mesmo milionésimo de segundo. Neste caso o identificador pode ser igual e levar a problemas de colisão de chaves.

Para a chave primária do modelo foi gerado um identificador único com um prefixo aleatório, à qual foi aplicada ainda o algoritmo de hash `md5`. A utilização do algoritmo md5 não foi escolhida para servir algum tipo de propósito de segurança sendo aplicado unicamente pelo facto de a sua hash ser representada por uma sequência de 32 caracteres hexadecimais.

```
uniqid()
// retorna um identificador de 13 caracteres.
>> 4f829833a963b

uniqid('foo_')
>> foo_53fcf4104dd2d
// retorna um identificador de 17 caracteres. O número de caracteres depende
do tamanho do prefixo (4+13 neste caso).

uniqid(mt_rand())
>> 116594346353fcf4b0ae626
// retorna um identificador de 23 caracteres. A função mt_rand() gera um
número aleatório através do algoritmo Mersenne Twister que produz um valor
"mais aleatório" e é 4x mais rápido que a função rand().


md5(uniqid(mt_rand()))
>> fe20c17f232c55d76bd69bd5e916f462
// retorna um identificador de 32 caracteres.
```

Figura 4.43. Descrição por passos da geração de identificadores únicos para o GnuCash

4.3.2. Correspondência entre entidades

Este módulo tem como objectivo cumprir duas operações:

- exportar informações sobre os clientes da plataforma e-commerce para o GnuCash.
- inserir uma nova transacção no GnuCash quando uma encomenda tiver sido dada como concluída.

À semelhança da implementação do módulo de integração com o software de facturação endereço de e-mail mantém-se o atributo de referência para fazer corresponder o cliente do Magento ao GnuCash. De igual modo o `id` da encomenda tem um papel na correspondência com a respectiva transacção. Este valor é inserido no campo `num` do registo de uma transacção, que segundo a documentação GnuCash serve exactamente o propósito de inserir opcionalmente um número que identifique ou esteja associado à transacção (e.g. número do cheque).

4.3.3. Implementação dos Modelos

A propriedade de dupla-entrada do software de contabilidade implica que todas as transacções comerciais envolvem pelo menos duas contas. Os contabilistas utilizam os termos "débito" e "crédito" para descrever se o dinheiro está a ser transferido para ou de uma conta. O montante é registado na coluna do débito, que é a coluna da esquerda, quando está a ser transferido para uma conta. O montante é registado na coluna do crédito, que é a coluna da direita, quando está a ser transferido de uma conta. Para qualquer transacção, o total dos débitos (entradas na coluna da esquerda) tem de ser igual ao total dos créditos (entradas na coluna da direita). Os registos do GnuCash usam por defeito títulos de colunas "comuns" tais como "depósito" e "levantamento".

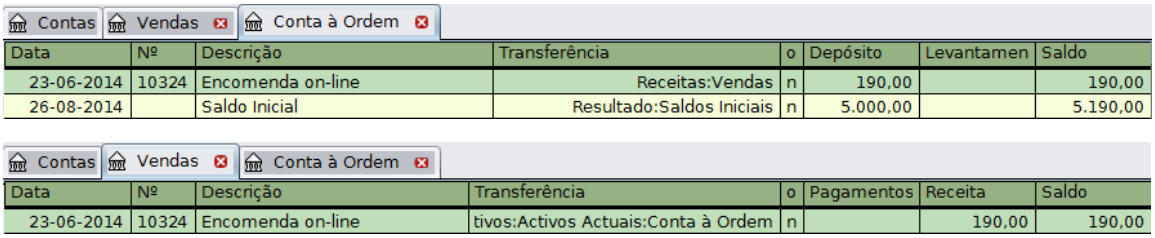

Figura 4.44. Sistema de dupla-entrada no GnuCash

Por exemplo, quando uma empresa pede dinheiro ao banco a sua conta 'Conta Bancária' irá aumentar e a sua conta de obrigações 'Empréstimos' irá diminuir. De igual modo se uma empresa pagar 1000€ por um anúncio publicitário, a sua 'Conta Bancária' registará o crédito desse valor e a sua conta de despesas em 'Publicidade' registará o débito desse mesmo montante.

Para o caso particular, as encomendas serão registadas na conta de receitas 'Vendas' e na conta de activos 'Conta à Ordem'. Cada transacção estará associada a duas entradas na tabela de dupla-entrada (tabela `splits`) e, à semelhança da implementação do módulo anterior, o registo de uma transacção e suas entradas deverão acontecer numa única transacção, ou falhar. A aplicação deste módulo automatiza assim os registos de vendas da plataforma e-commerce Magento ficando ainda por desenvolver uma interface que permita ao utilizador escolher as contas de origem e destino assim como o estado da encomenda que faz desencadear a exportação dos dados.

Capítulo 5
Aplicação Real

Este capítulo descreve a configuração de uma loja on-line onde foi aplicada a solução de gestão empresarial implementada. Esta loja é gerida por um empresário em nome individual que pretende vender peças de bijuteria para vários países da Europa. Está inicialmente disponível para os clientes em três línguas diferentes: português, inglês e francês. O pagamento pode ser feito por transferência bancária, *Paypal* ou à cobrança.

A loja tem domicílio fiscal em Portugal Continental e distribui a partir de França. O envio é feito a partir de França para outros países através da rede de distribuição *La Poste*[1]. Os custos de transporte são pagos pelo cliente ou oferecidos, se o número de produtos tiver um custo superior a 100 euros.

5.1. Alterar Idioma

A plataforma Magento vem apenas instalada com a língua inglesa. Foram descarregados os pacotes de português e francês na página de traduções[2] da plataforma Magento e extraídos para a pasta do Magento. O pacote de tradução pode não estar completo e algumas palavras aparecem ainda em inglês.

5.1.1. Alterar o idioma da zona de administração

Ao entrar na zona de administração é possível encontrar ao fundo da página uma caixa de selecção com o idioma actualmente definido "English (United States) / English (United States)" (ver Figura 5.1). O idioma foi alterado para "Português (Portugal) / português (Portugal)".

Figura 5.1. Selecção de idioma na área de administração do Magento

A partir desta secção as interacções com a área de administração da plataforma Magento serão descritas em português, de acordo com os valores apresentados pela tradução.

5.1.2. Adicionar um novo idioma para a área de cliente

No painel de administração, em `Sistema > Gerir Lojas`, foi criada uma nova vista de loja. As vistas de loja permitem ver a mesma loja sob configurações diferentes. Neste caso foram criadas mais duas vistas além da vista padrão (em inglês): "Português (PT)" e "Français (FR)".

No painel de administração, em `Sistema > Configuração > Geral` foi escolhido o "Espaço Actual de Configuração" para cada vista criada e em "Opções de Idioma" foi definido o

[1] http://www.laposte.fr/
[2] http://www.magentocommerce.com/translations/

idioma "Português (Portugal)" e "Francês (França)" para as respectiva vistas. O cliente pode seleccionar a vista pretendida através da caixa de selecção na barra superior da plataforma e-commerce (ver Figura 5.2).

Figura 5.2. Caixa de selecção de idioma para clientes no Magento

5.2. Configurar IVA

A plataforma Magento foi configurada de forma a aplicar os impostos definidos na Legislatura Portuguesa a aplicar na compra de produtos ou serviços: o IVA.

O IVA corresponde ao "Imposto sobre o Valor Acrescentado", uma taxa aplicada ao consumo, sendo tributado quando ocorre transmissão do bem ou do serviço, e se procede à emissão da factura, pela empresa que fornece esse produto ou serviço.

A taxa normal é aplicada na maioria dos serviços que são prestados aos clientes, no entanto existem algumas categorias de produtos ou alguns tipos de actividade como a hotelaria, construção civil ou transporte de passageiros em que se aplicam as taxas de IVA intermédia ou reduzida. A tabela 5.1 apresenta os diferentes valores do IVA para Portugal Continental, Madeira e Açores.

Taxa	**Portugal Continental**	**Madeira**	**Açores**
Normal	23%	22%	18%
Intermédia	13%	12%	10%
Reduzida	6%	5%	5%

Tabela 5.1. Taxas de IVA em Portugal

5.2.1. Adicionar um novo imposto

As novas regras de imposto podem ser adicionadas pelo painel de administração em `Vendas > Imposto > Gerir zonas de imposto e taxas`. Na página de gestão de taxas, clicar em "Adicionar Nova Taxa de Imposto" no canto superior direito (ver figura 5.3).

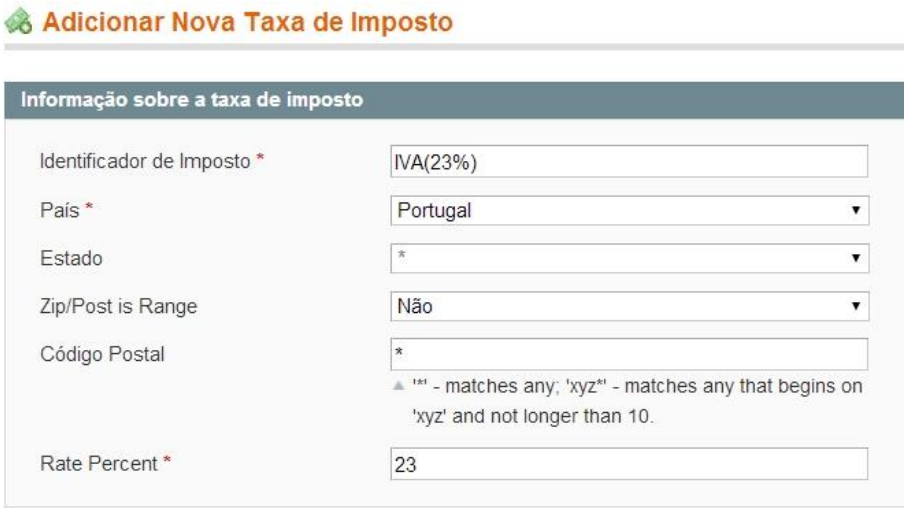

Figura 5.3. Criação de um novo imposto no Magento

A taxa normal de IVA foi configurada como apresentado na Tabela 5.2.

Opção	**Valor**	**Descrição**
Identificador de Imposto	IVA(23%)	Nome identificativo.
País	Portugal	País para o qual se aplica o imposto específico. Campo obrigatório.
Estado	*	Não aplicável para Portugal
Colecção de Código Postal	Não	Se a regra é limitada a um conjunto de códigos postais.
Código Postal	*	O caractere "*" significa "todos os existentes"
Taxa	23.00	Valor em percentagem da taxa a aplicar. 22 (Madeira) ou 18 (Açores)

Tabela 5.2. Configuração do IVA Normal no Magento

5.2.2. Adicionar uma nova regra de imposto

Foi necessário definir as regras para a qual se aplica esta taxa. Estas regras podem ser configuradas em `Vendas > Imposto > Gerir regras de imposto`.

Para adicionar uma regra basta clicar em "Adicionar Nova Regra de Imposto" no canto superior direito da página de gestão de taxas. A configuração foi feita de forma a aplicar o imposto sobre produtos na venda a retalho.

Opção	Valor	Descrição
Nome	IVA(23%)	Nome identificativo.
Classe de imposto do cliente	Retail Customer	A retalho
Classe de imposto do produto	Taxable Goods	Produtos
Taxa de Imposto	IVA(23%)	Taxa definida anteriormente
Prioridade	0	Prioridade deste imposto quando se aplica mais do que um imposto. Números mais baixos têm maior prioridade. Se duas regras fiscais têm a mesma prioridade então os impostos são aplicados juntos.
Calcular subtotal apenas	Não	
Ordem apresentação	0	Ordem em que as regras fiscais são exibidas na página "Gerir regras de imposto". Números mais baixos têm maior ordem de classificação na lista.

Tabela 5.3. Criação de uma nova regra de imposto para o IVA Normal no Magento

5.2.3. Cálculo e visualização de imposto

Na área de administração, em `Sistema > Configuração > Imposto`, foi definido que os preços inseridos no catálogo já incluem imposto. O método de cálculo de imposto definido foi "Total Linha". O imposto é por isso calculado sobre o valor total de uma linha de produtos. Quanto à visualização, foi definido que os preços são apresentados no catálogo incluindo impostos e são apresentados no carrinho de compras e encomendas com e sem impostos.

5.3. Adicionar produtos

A plataforma Magento permite a personalização dos produtos de diferentes formas, sendo uma delas a definição dos atributos associados ao produto. No Magento os "Atributos" são aspectos quantificáveis ou descritivos de um produto, como cor, fabricante e número de SKU. Existem dois tipos de atributos no Magento: "Atributos de Sistema" e "Atributos Personalizados". Por padrão, o Magento inclui todos os atributos de sistema necessários. Estes não podem ser excluídos e cada produto deve ter este tipo de produtos, como nome, preço e SKU. Por outro lado, os atributos personalizados são criados pelo responsável da loja e são específicos de certos produtos.

5.3.1. Gerir atributos

Para esta loja é preciso adicionar um atributo personalizado para as peças vendidas: "Tipo de Colar". Este atributo pode ter um de três valores: "Fio preto ajustável", "Corrente de Metal 45cm" e "Corrente de Metal 70cm". Adicionar um novo atributo está acessível através de

`Catálogo > Atributos > Gerir atributos`. No separador "Propriedades" foram configurados os seguintes campos:

Opção	**Valor**	**Descrição**
Código de Atributo	necklace_type	Para uso interno apenas. Deve ser único e sem espaços.
Espaço	Global	A configuração é aplicada a todas as vistas de loja.
Tipo de entrada	Selecção Múltipla	O atributo será escolhido através de uma caixa de selecção na página do produto.
Valor único	Sim	Só pode ser escolhida uma das opções.
Valores necessários	0	É obrigatório escolher uma das opções
Validação	Nenhuma	O sistema pode validar se o valor introduzido é um e-mail, número decimal, url, letras, etc…
Aplicar a	Todos os Tipos de Produtos	Permite seleccionar só alguns tipos de produtos.

Tabela 5.4. Novo atributo para tipo de colar no Magento

No separador "Gerir Rótulo / Opções" foi definido o título do atributo a apresentar na zona de administração, na vista padrão da loja e outras vistas configuradas:

Opção	**Valor**
Admin	Necklace Type
Default Store View	Necklace Type
Português (PT)	Tipo de Colar
Français (FR)	Type de collier

Tabela 5.5. Tradução do atributo para tipo de colar para as diferentes vistas no Magento

Também de acordo com as diferentes vistas foram definidos as opções de escolha possíveis para o atributo:

Admin e **Default Store View**	**Português (PT)**	**Français (FR)**
Black adjustable wire	Fio preto ajustável	Cordon Noir Ajustable
Metal chain 45 cm	Corrente de metal 45 cm	Chaîne en métal 45 cm

Metal chain 70 cm	Corrente de metal 70cm	Chaîne en métal 70 cm

Tabela 5.6. Opções de escolha para o atributo tipo de colar no Magento

5.3.2. Gerir grupos de atributos

De forma a associar as peças de bijuteria, mais especificamente colares, a este novo atributo foi necessário criar um novo grupo que una este atributo aos atributos do sistema. Em `Catálogo > Atributos > Gerir Grupos de Atributos` foi criado um novo grupo com o nome "Colares" (uso interno apenas). O atributo "necklace_type" foi adicionado ao grupo, assim como, implicitamente, os restantes atributos de sistema.

5.3.3. Gerir produtos

A área de produtos está acessível através do painel de administração em Catálogo > Vendas. Na página "Gerir Produtos" estarão listados os produtos já adicionados. O botão "Adicionar novo produto" permite definir inicialmente o tipo de produto. O tipo de produto pode ser:

- Produto Simples. O tipo de produto mais comum, com uma configuração única.
- Grupo de produtos. Permite juntar dois ou mais produtos e vendê-los como um só.
- Produto Configurável. Permite aos clientes seleccionar variantes do produto (e.g. diferentes tamanhos e cores para *t-shirts*)
- Produto Virtual. Os produtos virtuais não existem nem em formato físico nem digital e não podem ser enviados. Este tipo de produto adequa-se mais à venda de serviços.
- Pacote de Produtos. Este tipo de produto é adequado para situações em que o cliente tem que escolher um várias opções de configuração

No total foram adicionadas 21 peças com as seguintes opções do produto, variando nalguns atributos consoante a peça:

Opção	**Valor**
Grupo de Atributos	Colares
Tipo de Produto	Produto Simples

Opção	**Valor**
Nome	CHOCOLATE
Descrição	Teobromina: o alimento dos deuses (theo: Deus, broma: alimento), agora ao alcance dos amantes do chocolate numa jóia original. Esta molécula, teobromina, encontra-se no

	chocolate e no cacau, mas pode também ser encontrada no chá, na noz de cola e nas bagas de guaraná.
Pequena descrição	Teobromina
SKU	Mol1
Peso (Kg)	0.09
Estado	Activo
Visibilidade	Catálogo, Pesquisa
Preço (inclui imposto)	14
Classe de Imposto	IVA(23%)
Imagem	
Quantidade	50
Disponibilidade Inventário	Em inventário
Produtos relacionados	Foram seleccionadas todas as peças semelhantes.

Tabela 5.7. Criação de um novo produto no Magento

De notar que para o produto aparecer na área de cliente deve estar activo, visível no catálogo e em inventário. Outras configurações podem ser feitos na área de gestão de produto como categorias, metadados, promoções ou até mesmo configurações facturação se o módulo para o Projecto Colibri se encontrar instalado.

5.4. Definir método de envio

A plataforma Magento permite seleccionar vários métodos de envio configurando diferentes taxas a aplicar ao transporte de encomendas. Os métodos de envio podem ser configurados na área de administração em `Sistema > Configuração > Vendas > Métodos de Envio`. Foi importado um ficheiro em formato CSV[1] com as taxas a aplicar para cada país de destino, consoante o número de peças encomendadas. O ficheiro está representado na Tabela 5.8 sendo que o ficheiro tem efectivamente um registo para cada país europeu.

País	Região/Estado	Código Postal	# de itens (e superiores)	Preço de envio
FRA	*	*	0	1,65

[1] Comma-Separated Value

Todos os outros países europeus	*	*	0	2,55

Tabela 5.8. Configuração das taxas de envio para a europa no Magento

Foi activado o método "Envio Gratuito" para quantidades superiores a 100. Na área de administração em `Sistema > Configuração > Vendas > Métodos de Envio` foi definido França como o país de origem.

Figura 5.4 Simulação de compra de produtos no Magento

Existe também a opção de permitir a escolha de uma transportadora na altura de encomenda, sendo as taxas configuradas automaticamente. Existem algumas transportadoras já disponíveis na plataforma Magento como a UPS, DHL e FedEX, em que as taxas são calculadas através de uma ligação aos serviços das transportadoras.

5.5. Definir método de pagamento

Os métodos de pagamento foram definidos em `Sistema > Configuração > Vendas > Métodos de Pagamento`. O método de pagamento "Cartão de Crédito" foi desactivado e o método de pagamento "Transferência Bancária" e "Contra-reembolso" foram activados. A conta *Paypal* foi configurada através da conta de *Paypal* do responsável da loja.

5.6. Emitir factura de encomenda

As encomendas podem ser visualizadas na plataforma Magento em `Vendas > Encomendas`. Se a encomenda estiver pronta para ser processada basta seleccionar a encomenda a facturar e clicar no botão "Factura", no canto superior direito, e de seguida "Submeter factura".

As encomendas processadas encontram-se registadas no software de facturação, Projecto Colibri, no separador "Movimentos de Clientes". Após seleccionada a encomenda a facturar esta deve ser duplicada através do ícone "Duplicar Registo" (ver Figura 5.5) e guardada com a série 2014. O ícone "Imprimir relatório" emitirá o documento de factura em papel ou formato digital.

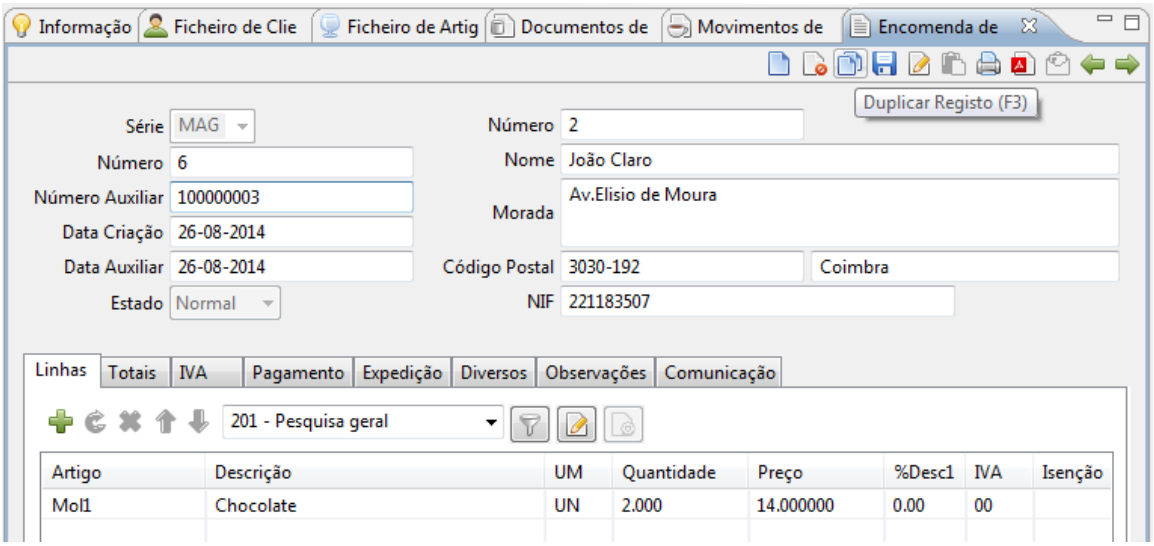

Figura 5.5 Duplicação do documento Factura no GnuCash

5.7. Emitir relatório de Balanço Patrimonial

O Balanço é um instrumento contabilístico que reflecte a situação económico financeira da empresa. O Balanço representa a situação patrimonial da empresa (activos, dívida e capital) num determinado momento de tempo. O relatório de Balanço pode ser gerado no software de contabilidade GnuCash através da Barra de Ferramentas > Relatórios > Activos e Passivos > Balanço.

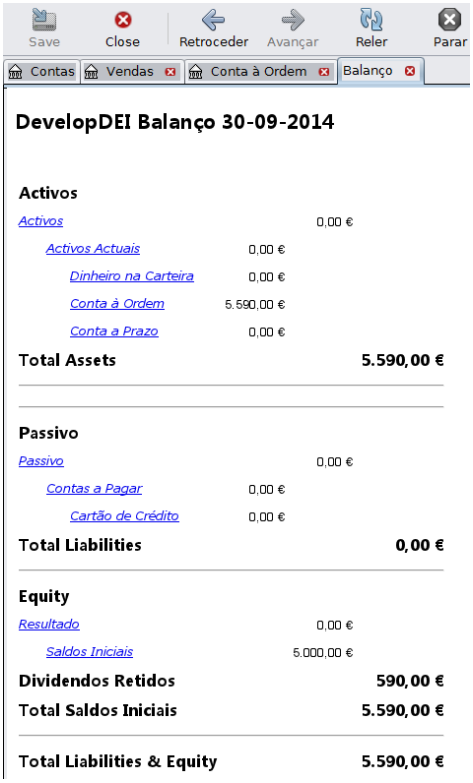

Figura 5.6. Relatório Exemplo de Balanço do 3º Trimestre do GnuCash

Capítulo 6
Conclusões

Este trabalho tem o propósito de apresentar uma solução integrada de gestão empresarial, sem custos de aquisição, com foco em software *open-source*, uma área emergente e considerada vantajosa para as empresas. As aplicações foram seleccionadas com base nas suas funcionalidades, capacidade criação de novas funcionalidades e integração com outras aplicações, popularidade e adopção pela comunidade. O cumprimento da legislação portuguesa no que respeita à utilização de software certificado para facturação levou à escolha de uma aplicação não *open-source* de modo a que fosse cumprido outro requisito importante no resultado de final, o de apresentar uma solução grátis. A utilização de software *open-source* para o software de certificação abriria espaço a outras abordagens como o desenvolvimento de uma interface no Projecto Colibri para criação de novos produtos, clientes e encomendas (*push*) ou importação de dados apenas a pedido da aplicação (*pull*). A solução apresentada é apenas uma prova de conceito e propõe-se apenas a responder às necessidades básicas de uma empresa que venda os seus produtos on-line. A solução final foi aplicada num caso concreto de um empresário em nome individual e com necessidades particulares para o seu negócio. A solução apresentada não teve problemas em adaptar-se aos requisitos e mostra-se suficientemente genérica para venda de produtos ou serviços on-line, deixando no entanto por explorar o real valor na área de contabilidade.

Para um trabalho futuro é importante que o desenvolvimento do sistema passe somente pela utilização de software *open-source*, de modo a que possam retiradas outras vantagens do software *open-source* que não apenas o seu custo inicial. Pelo seu carácter particular nacional o software de facturação apresenta um maior desafio na adaptação frequente à nova legislação, assim como a sua certificação junto da Autoridade Tributária e Aduaneira. Também pelo seu carácter particular e por ser uma necessidade transversal a qualquer negócio de venda de produtos ou serviços a alternativa *open-source* tem bastante procura, evidente em fóruns e outros espaços de discussão on-line. O trabalho futuro passa também pelo desenvolvimento de novas funcionalidades para uma solução de gestão empresarial mais completa com funcionalidades como CRM[1], HR[2] e BI[3]. Importante também para a validação e desenvolvimento contínuo de um ERP FOS ajustado às necessidades dos empresários em Portugal é a criação de um espaço próprio on-line para distribuição do software que atraia atenção da comunidade.

De um modo geral, a utilização de software *open-source* integrado apresenta-se como uma solução de grande potencial para as empresas. A utilização de aplicações já existentes traz algum conforto na maturidade do software e envolvimento das respectivas comunidades. Esta dependência implica que a capacidade da solução final se manter actualizada está dependente das novas versões lançadas pelas aplicações utilizadas. Esta questão pode ser um obstáculo para empresas que não incluam pessoas com competências de programação, estando assim dependentes da existência de uma comunidade envolvida e motivada.

[1] Customer Relationship Management, em inglês, ou Gestão do Relacionamento com o Cliente, em português.
[2] Human-Resources, em inglês, ou Recursos Humanos, em português.
[3] Business Intelligence, em inglês, ou Inteligência de Negócios, em português.

Referências

Adam, R., Kotzé, P., & Merwe, A. v. (2011). *Acceptance of enterprise resource planning systems by small manufacturing enterprises.*

Anderegg, T. (2007). *MRP / MRP II / ERP / ERM - Confusing Terms and Definitions for a Murky Alphabet Soup.* Obtido em 13 de 07 de 2014, de waikato linux users group: http://wiki.wlug.org.nz/EnterpriseSpeak

Carvalho, R., & Campos, R. (2009). Uma análise de aspectos relacionados ao desenvolvimento e adoção de Enterprise Resources Planning livre de código aberto. *Gest.Prod.*, 667-678.

Deep, A., Guttridge, P., & Burns, S. &. (2008). Investigating factors affecting ERP selection in made-to-order SME sector. *Journal of Manufacturing Technology Management*.

Ecommerce Europe. (2014). *European B2C E-commerce Reports 2014.*

Fowler, M. (2003). *Solving Integration Problems Using Patterns.* Obtido em 29 de 06 de 2014, de eaipatterns: http://www.eaipatterns.com/Chapter1.html

Glass, M. K. (2004). *Beginning PHP, Apache, MySQL Web Development.* John Wiley & Sons.

GNU General Public License, version 2. (s.d.). Obtido em 28 de 07 de 2014, de gnu.org: http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

GUID structure (Windows). (n.d.). Retrieved 07 03, 2014, from Msdn.microsoft.com: http://msdn.microsoft.com/en-us/library/aa373931(VS.85).aspx

Hohpe, G., & Woolf, B. (2004). *Enterprise integration patterns: Designing, building, and deploying messaging solutions.* Addison-Wesley Professional.

Huq, N. (2010). *Why Selecting an Open Source ERP over Proprietary ERP? A focus on SMEs and Suppliers perspective.*

Instituto Nacional de Estatística. (2013). *Estatísticas do Emprego. 1º Trimestre.*

Ishenko, O. (01 de 2008). *Magento EAV System.* Obtido de Solving Magento: http://www.solvingmagento.com/magento-eav-system/

Kim, H., & Boldyreff, C. (2005). *Open Source ERP for SME.*

Kissinger, K. (11 de 2008). *Can open source ERP software make an impact on the market?* Obtido de techtarget: http://searchmanufacturingerp.techtarget.com/news/article/0,289142,sid193_gci1339673,00.html

LeClaire, J. (12 de 2006). *Open Source, BI and ERP: The Perfect Match?* Obtido de Linux Insider: http://www.linuxinsider.com/story/54938.html

Manual de Utilização do OpenK POS. (12 de 2013).

Norton, J. (2012). *Open for Business.*

Portal das Finanças. (s.d.). *NOVAS REGRAS SOBRE UTILIZAÇÃO DE PROGRAMAS INFORMÁTICOS DE FACTURAÇÃO.* Obtido em 2014 de 07 de 05, de Portal

das Finanças: http://info.portaldasfinancas.gov.pt/NR/rdonlyres/92506C39-7068-44FC-8365-8CFACE749DC2/0/Cert_Prog_Fat.pdf

Rosenberg, J. (21 de 12 de 2009). *The meaning of open*. Obtido de Google Official Blog: http://googleblog.blogspot.pt/2009/12/meaning-of-open.html

Shaikh, M., & Cornford , T. (2011). *Total Cost of Ownership of Open Source Software.*

Smets-Solanes, J. P., & Carvalho, R. A. (2003). ERP5: A next-generation, open-source ERP architecture. *IEEE IT Professional*, 38-44.

softwarecertificado.com. (s.d.). *O que muda em 2012?* Obtido em 07 de 05 de 2014, de Software Certificado: http://www.softwarecertificado.com/Default.aspx?action=ArticleViewer&target=407

Webnet Hosting. (15 de 01 de 2013). *Most Popular Open source Ecommerce shopping carts – Top 20*. Obtido de Slideshare: http://pt.slideshare.net/boneyp/most-popular-open-source-ecommerce-shopping-carts-top-20

Zend Framework 2.0 Requirements . (2009). Obtido em 4 de 2014, de github.com/zendframework/: https://github.com/zendframework/zf2/wiki/Zend-Framework-2.0-Requirements