INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Adaptadores para Facturação Electrónica

Rute Sofia Rodrigues Duarte

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Júri

Presidente: Pedro Manuel Moreira Vaz Antunes de Sousa, Departamento de Engenharia Informática (DEI)

Orientador: Diogo Manuel Ribeiro Ferreira, Departamento de Engenharia Informática (DEI)

Vogais: Miguel Leitão Bignolas Mira da Silva, Departamento de Engenharia Informática (DEI)

Setembro de 2007

Agradecimentos

Ao Prof. Diogo Ferreira pelo apoio prestado à realização deste trabalho, pelas ideias e esclarecimentos.
Ao Prof. Miguel Mira da Silva.

Ao João Martins pelo enorme apoio prestado longo deste trabalho, pela sua disponibilidade, pelas trocas de ideias, pelos esclarecimentos e motivação.
Aos antigos colegas da Create It pelo apoio e ajuda.

Aos meus pais e ao Júlio pela apoio incondicional, pelos seus esforços para poder frequentar e acabar o curso e concretizar os meus objectivos e ambições, e pelos conselhos e apoio que me tem dado ao longo da vida.
Ao meu namorado José pelo apoio, motivação, compreensão e ajuda nos momentos mais difíceis e de stress.

Ao amigo e colega Bruno pelo apoio, pela ajuda, pelo companheirismo e compreensão ao longo deste trabalho.

Aos amigos Maria João e Sérgio pela ajuda já no final do trabalho.

Resumo e Palavras-chave

A importância da integração de Sistemas de Informação tem crescido devido à necessidade de interligar as suas funcionalidades e informação que manipulam, sendo o maior desafio integrar os sistemas internos que resolvem problemas específicos da empresa, com os sistemas de informação externos dos seus parceiros empresariais, por forma a tirar partido do negócio electrónico.

A integração entre empresas é um aspecto importante na facturação electrónica pois baseia-se na troca de dados entre diferentes aplicações das organizações. Este trabalho pretende mostrar uma forma de integrar os sistemas de informação existentes no mercado, com a construção de um adaptador que consiga enviar e receber facturas electrónicas de forma tanto quanto possível transparente para os *Enterprise Resource Planning* (ERP's) das várias organizações intervenientes. O mecanismo estudado baseia-se na utilização de metadados para descrever repositórios aplicacionais distintos e específicos, de forma a facilitar a integração entre os diferentes formatos utilizados pelos intervenientes no sistema de facturação e integração de novos intervenientes. Essa camada de metadados irá ser utilizada para conversão dos dados num padrão conhecido para troca de dados com um *Broker* de Facturação.

Palavras-chave:

Sistemas Legados, Facturação Electrónica, Integração de Aplicações e Processos Empresariais, Adaptadores, Metadados.

Abstract

The development of Information Systems has been led by the necessity of interconnection between the functionalities and information they manipulate, the biggest challenge being the integration of internal systems, which solve company specific problems, with external information systems of their partners, leading to a successful e-business.

Integration among companies is an important aspect in electronic invoice since it is based in the exchange of data between the different company applications. This project consists in the integration of the Information Systems, available at the e-market, through the construction of an adapter capable of sending and receiving electronic invoices in a transparent way to the *Enterprise Resource Planning* (ERP's) of the involved companies. The studied mechanism will be the use of metadata, to describe distinct and specific application repositories to ease the integration between the different formats used by the invoice system users as well as new users. This metadata layer will be used as a data converter to a known pattern to allow data exchange with an invoice Broker.

Keywords: Legacy systems, Electronic Invoice, Business Process and Application Integration, Adapters, Metadata.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas

EAI – Enterprise Application Integration

B2B - Business to Business

ERP – Enterprise Resource Planning

XML – Extensible Markup Language

XSD – Xml Schema Definition

WCF – Windows Communication Foundation

SQL – Structured Query Language

API – Application Programming Interface

EDIFACT – EDI for Administration, Commerce and Transport

EBXML – Electronic Business Extensible Markup Language

EDI – Electronic Document Interchange

W3C – Wide Web Consortium

SOA –Service Oriented Architecture

PME – Pequenas e médias empresas

CRM – Customer Relationship Management

BDC – Business Data Catalog

MSMQ – Microsoft Message Queuing

1. Introdução

A integração de sistemas de informação (*Enterprise Application Integration* ou EAI) foi sempre uma parte importante no desenvolvimento de sistemas de informação. Recentemente essa importância cresceu devido à necessidade de integrar diversos sistemas de informação entre empresas, e ao surgimento das chamadas arquitecturas orientadas a serviços (*Service Oriented Architecture* ou SOA). O grande desafio actualmente é integrar os sistemas internos, que resolvem problemas específicos da organização, com os sistemas de informação externos dos seus parceiros, de forma a tirar partido do negócio electrónico

A integração entre organizações (conhecida como *Business to Business Integration* ou B2BI) para este trabalho torna-se importante porque a facturação electrónica é baseada na troca de dados entre diferentes aplicações das organizações. As diferenças residem na forma como é realizada a troca de dados, nos tipos de dados que são trocados, nas garantias oferecidas, etc.

Um cenário típico para a facturação electrónica passa pela aquisição para os *Enterprise Resource Planning* ou ERP's da organização de um módulo específico para a troca segura de documentos em formato electrónico. Com este trabalho pretende-se estudar uma forma de integrar os sistemas de informação existentes no mercado electrónico, com a implementação de um adaptador que permita enviar e receber facturas electrónicas de forma transparente para os ERP's das várias organizações intervenientes.

Como protótipo foi desenvolvido um adaptador capaz de efectuar a troca de mensagens entre um ERP no mercado nacional e um Broker de facturação, sem implicar alterações no ERP.

Um dos mecanismos a estudar é a utilização de metadados (descrição das regras, estruturas de dados e restrições aplicáveis - dados sobre as API's[1] das aplicações de negócio (ERP)) para descrever repositórios aplicacionais distintos e específicos, por forma a que a construção da camada de metadados ajude na integração entre os diferentes formatos utilizados pelos intervenientes no sistema de facturação, e integração de novos intervenientes. Para cada aplicação de negócio, os metadados definem as entidades de negócio com que a aplicação de negócio interage e os métodos disponíveis na aplicação de negócio. Define-se estes metadados usando XML, sendo armazenados num repositório de metadados.

Esta camada de metadados irá ser utilizada para a conversão dos dados para um esquema XML padronizado para a troca de dados com o Broker de Facturação. Será também estudada a tecnologia WCF para o desenvolvimento de Web Services para a comunicação com o Broker de Facturação. Finalmente de referir que este protótipo será desenvolvido em tecnologias *Microsoft .Net 3.0* e o ERP utilizado para integrar é o Primavera Express da *Primavera Business Solutions*.

[1] *API* – é um conjunto de funções de software usadas por um programa como meio de providenciar o acesso às capacidades do sistema para comunicações.

O restante documento está organizado da seguinte forma: o capítulo 2 contém o estado-da-arte onde é feito uma análise de temas e tecnologias existentes relacionados com o que foi desenvolvido. Aborda conceitos de integração de empresas, integração de sistemas legados dentro de uma empresa e com outras empresas, facturação electrónica, formatos de dados existentes e adaptadores utilizados no mercado. No capítulo 3 é descrito o sistema de informação de base utilizado neste trabalho, bem como aspectos relacionados com a correspondência dos campos de uma factura electrónica com os campos presentes no sistema. No capítulo 4 é descrito o repositório de metadados, descrevendo objectivos, princípios gerais, estrutura e funcionamento. No capítulo 5 é descrito a solução tecnológica de integração, onde é mostrada a arquitectura do adaptador, dividindo-a nas suas camadas principais demonstrando desta forma a sua estrutura. Em cada camada é descrito o processo de realização do trabalho, incluindo aspectos de funcionamento, tecnologias utilizadas e decisões tomadas. No capítulo 6 é descrito um caso de estudo, onde é dado um cenário de negócio e mostrado um exemplo do tipo de documentos envolvidos, é explicado o processo de configuração do repositório de metadados e finalmente mostrado esquemas e imagens do funcionamento do adaptador.

2. Estado-da-Arte

2.1. Negócio Electrónico

A disponibilização do *World Wide Web*[2], durante os anos 90, permitiu a grande parte do tecido empresarial reequacionar as estratégias de actuação no mercado, provocando alterações profundas no ambiente negocial tradicional, designadamente no modo de relacionamento entre clientes e fornecedores. Esta alteração deu origem a uma nova forma de vender e comprar – o negócio electrónico – que se tem convertido num factor fundamental de competitividade e num fortíssimo indutor de produtividade para a generalidade das empresas.

De uma forma mais generalista pode-se definir o negócio electrónico como sendo o processo de distribuição de bens e serviços por intermédio de meios electrónicos. É então, um serviço da *Internet* onde empresas exibem os seus produtos para os utilizadores da *web*, colocando-os assim à disposição para compra *on-line*. Quando a integração entre os vários intervenientes é mais profunda, com negociação de formatos e devida contratualizada, estas interacções são conhecidas como Negócio Electrónico.

2.1.1. Negócio entre Empresas

Esta é uma categoria bastante importante do negócio electrónico, pois é nas transacções electrónicas entre empresas que se gera o maior volume de negócios deste tipo de negócio. Esta forma de negócio incentiva a novas e inovadoras formas de cooperação entre empresas, de forma a enfrentar com sucesso os desafios lançados pela globalização, tornando assim a empresa mais competitiva.

Acompanhando o desenvolvimento da Internet, o negócio electrónico entre empresas tem vindo cada vez mais a ganhar expressão, à medida que vão surgindo mais tecnologias *web* que se propõem dar suporte às transacções electrónicas entre várias empresas. Este tipo de negócio electrónico, e tal como o próprio nome sugere, envolve negócios entre empresas, e pode ser definido como a substituição dos processos físicos que envolvem as transacções comerciais, por processos electrónicos.

A implementação de negócio electrónico entre empresas envolve diversas fases, como referido em [21] mostrado na *Figura 2.1*. As fases representadas, e as mais importantes para este trabalho, referem-se à formatação num padrão conhecido e interpretação dos dados. A fase de formatação dos dados num padrão conhecido, abordada com maior detalhe na *Secção 2.3, Secção 4 e Secção 5.4*, trata-se do mapeamento de um arquivo do ERP para um formato conhecido no mercado, que pode ser: XML, EDIFACT, RosettaNet, EBXML, etc. A integração dos dados com as aplicações é um dos aspectos mais importantes, onde os diversos tipos de ERP deverão ser contemplados, pois os

[2] *World Wide Web* ou WWW – Rede de computadores na *Internet* que fornece informação mundial.

projectos de integração são normalmente dispendiosos e de resultados não imediatos e complexos de atingir se não forem bem planeados.

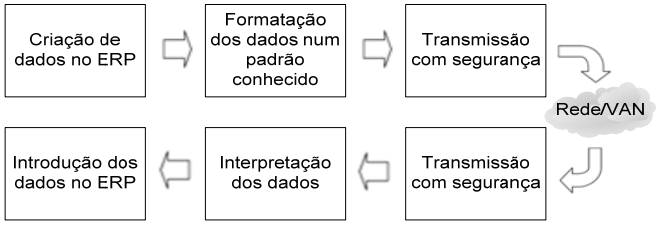

Figura 2.1 – Etapas no negócio electrónico entre empresas [21]

2.1.2. Factura Electrónica

A factura electrónica consiste num documento digital, que tem o mesmo valor legal da antiga factura em papel e vem substituir a mesma. A diferente entre uma factura em papel e uma factura electrónica e que a última é em formato digital, assim como as tecnologias utilizadas por ambas para processamento e gestão. Uma factura consiste na prova de pagamento respeitante ou a uma venda, fornecimento de serviços, ou de outras actividades. Trata-se de um documento de pagamento ou recibo, utilizado oficialmente para aspectos contabilísticos e fiscais. Esta factura é enviada e processada pela empresa que presta os serviços. [18] [19]

Estes documentos são transaccionados entre as partes intervenientes utilizando para isso a Internet, deixando assim de ser necessário o seu envio em papel. É esperado com isto simplificar e modernizar aspectos relacionados com o arquivo e transmissão do documento, bem como em matéria de aplicação do imposto (IVA).

Como nos diz [36], "a adopção da factura electrónica é um projecto de modernização tecnológica com custos relativamente baixos, e com impactos potenciais extremamente interessantes em áreas tão fundamentais como as da produtividade e competitividade da economia nacional, a redução de custos administrativos no Estado, e a promoção da inovação e base tecnológica com o consequente desenvolvimento de novas áreas de negócio."

Para a sua utilização é necessário respeitar algumas regras relativas à sua transmissão e arquivo electronicamente, como explicado em detalhe na *Secção 2.1.3.* Como nos diz [37], os objectivos principais da utilização da factura electrónica são a supressão da construção, circulação e arquivo das facturas em suporte papel, bem como os atrasos e trabalho inerente com a circulação. As vantagens para os emissores, descritas neste documento, são: Elimina custos desnecessários, deixando de ser necessário o uso de papel para impressões de factura. Outro aspecto também importante é a redução dos custos inerentes ao volume da correspondência. Ainda relacionado com o processo de correspondência, através da utilização da factura electrónica é possível reduzir o tempo de tratamento e do próprio processo de correspondência. Ao contrário do sucedido na factura normal, na factura electrónica é enviada a confirmação de recepção pelos destinatários. Desta forma é

possível enviar facturas sem atrasos, sem hipóteses de extravio e de forma cómoda. Por outro tem as seguintes vantagens para os receptores: Com a utilização de facturas electrónicas (equivalente a um documento electrónico), torna-se mais fácil o seu arquivamento e pesquisa a qualquer momento. Deixa de ser necessário o arquivamento físico destes documentos, sendo alocados no sistema imediatamente a sua recepção. Para pesquisa basta utilizar o sistema utilizado pelas empresas para processo de facturação.

De acordo com o Decreto-Lei nº 196/2007 [32] e o documento [18], existem dois tipos possíveis de facturas electrónicas, que garantem a sua autenticação:

- Documentos electrónicos onde é utilizada um assinatura electrónica avançada ou digital para garantir a sua integridade e autenticação;
- Documentos electrónicos trocados no contexto do EDI (*Electronic Data Interchange* - utilização de um contrato preestabelecido entre as partes participantes).

Um aspecto bastante importante da utilização da factura electrónica consiste em aspectos de segurança e confirmação de recepção que é possível alcançar com a utilização da factura electrónica. Através dos dois tipos de facturas anteriormente referido, pode ser também assegurada a integridade do documento, garantindo que o mesmo chega ao receptor sem qualquer alteração ou modificação, em comparação com o documento enviado pelo emissor. [19]

Como o documento [19] [32] nos diz, uma factura electrónica deve obedecer aos seguintes critérios e características:

- Garantir a autenticidade do emissor;
- Garantir a integridade das facturas;
- Garantir a integridade da sequência das facturas;
- Assegurar o não repúdio em ambas as partes participantes na transacção (emissor e receptor).

Como mais à frente na *Secção 5.2.1* será explicado e também mencionado na *Secção 2.1.3*, a autenticação consiste na identificação, sem dúvidas, da parte envolvida na transacção.

A certificação necessária para o uso das facturas electrónicas, como indicado pelo documento [37], consiste na utilização de uma assinatura digital ou assinatura electrónica avançada, através de certificados digitais. A assinatura digital consiste numa assinatura electrónica avançada ou digital que garante a autenticidade do remetente (protecção contra utilização não permitida da identificação de um interlocutor) e integridade do conteúdo do documento (protecção contra alterações não detectadas nos dados). Garante também o não repúdio de recepção (protecção contra a negação da recepção dos dados) e o não repúdio de origem (protecção contra a negação da autoria dos dados) numa transacção electrónica. Esta assinatura corresponde a uma chave de codificação, feita através de um certificado digital. Um certificado digital serve essencialmente para identificação dos intervenientes, e garantir a confidencialidade da informação trocada entre ambos. Estes certificados são obtidos através de uma entidade certificadora, que consiste numa entidade externa e

independente de confiança, a nível de certificação. Desta forma, ficará validade a integridade e a autenticidade do conteúdo das facturas. Alguma modificação ou alteração nos dados resulta numa assinatura diferente do que a produzida pelos dados iniciais.
Resumidamente, e de acordo com [35], os principais benefícios inerentes à utilização da factura electrónica consiste na:

- Redução de custos;
- Redução do tempo e dos recursos humanos necessários nas tarefas de processamento e envio de uma factura em suporte papel;
- Redução do tempo e dos recursos humanos necessários nas tarefas de processamento e introdução dos dados de um documento recebido;
- Integridade do conteúdo do documento e segurança da identidade do seu emissor.

2.1.3. Legislação

Como o documento [19] nos indica, e no âmbito da internacionalização da utilização da factura electrónica entre empresas, tornou-se essencial a criação de um quadro comum de suporte legal à facturação electrónica que permitisse uma uniformização das regras aplicadas a esta forma de troca de documentos electrónicos. A factura tem um grande poder contabilístico e fiscal, sendo essencial para a aplicação do Imposto de Valor Acrescentado (IVA) sobre qualquer transacção financeira de uma empresa.

Como indicado em [29], com aprovação do Decreto-Lei nº 375/99 [33], de 18 de Setembro de 1999, foi estabelecida a equiparação entre um factura emitida em suporte papel e uma factura electrónica.
Com a adopção da Directiva nº 2001/115/CE, Decreto-Lei nº 256/2003 [30] e do Decreto-Lei 196/2007 [32], as altas entidades pretendem simplificar e modernizar as condições aplicáveis à facturação em relação ao IVA. Destaque para o estabelecimento de uma lista de elementos obrigatórios a constar nas facturas emitidas, bem como regras de elaboração, arquivo e conservação por meios electrónicos. Apenas é permitido o arquivo em suporte electrónico das facturas ou documentos electrónicos emitidos via electrónica, desde que se encontrem disponíveis para acesso em qualquer momento e circunstância, e com garantia de integridade da origem e do seu conteúdo.
São considerados documentos equivalentes à factura os documentos que contêm os requisitos exigidos para a factura, e visem alterar a factura original. As facturas ou documentos equivalentes podem ser emitidos pela entidade que adquiriu um bem ou serviço ou por terceiros em nome e por conta do sujeito passivo.

Como referido anteriormente na *Secção 2.1.2* e em [17], é exigido garantia de autenticidade da origem e integridade do seu conteúdo na emissão de facturas via electrónica, através da utilização de uma assinatura electrónica avançada ou através da utilização de intercâmbio electrónico de dados. Para tal é necessário um acordo prévio entre os intervenientes da transacção, bem como o não repúdio por ambas as partes.

Também a necessidade de numeração em uma ou mais séries as facturas, documentos equivalentes e guias ou notas de devolução, devendo ser guardado na respectiva ordem os seus duplicados e todos os exemplares que tenham sido anulados ou inutilizados, com identificação daqueles pelos quais foram substituídos, no caso.

Com a Resolução do Conselho de Ministros n.º 137/2005 [31], foram decididos vários aspectos, tais como:

- Emissão obrigatória das facturas via electrónica, excepto se o destinatário indicar vontade de recepção da factura ou documento equivalente em suporte papel. Mas deverá ser sempre emitida e enviada para o destinatário a factura ou documento equivalente via electrónica em conjunto com a factura em suporte papel;
- Recebimento de facturas ou documentos equivalentes por via electrónica preferencialmente com uma antecedência mínima de três meses relativamente ao seu início.

2.2. Integração de Sistemas de Informação

A integração dos sistemas de informação muitas vezes é um obstáculo para muitas empresas, não podendo na maioria das vezes comunicar com os parceiros empresariais, devido à incompatibilidade dos seus sistemas. Estes sistemas autónomos e em muitos casos heterogéneos não foram projectados para colaborarem com outras aplicações, tendo a tendência de ser desenvolvidos independentemente e sem qualquer coordenação. Recentemente, esta situação agravou-se ainda mais devido à necessidade de integrar sistemas de informação quer dentro de uma empresa, quer entre empresas. [23]

Para tirar partido do negócio electrónico é necessário integrar os vários sistemas de informação dentro da própria organização. A solução adoptada por muitas das empresas Portuguesas passou pela integração dos sistemas existentes através de sistemas ERP. Foi assumido por muitos que o problema poderia ser resolvido substituindo todos os sistemas de informação por um único ERP.

Como referido em [14], para implementar um sistema ERP, primeiro é preciso reimplementar os processos de negócio da empresa de forma a adoptar o padrão dos processos de negócio do ERP. Esta reimplementação pode resultar num benefício para empresas que necessitem de reestruturar os seus processos de negócio ou pretendam deixar os seus sistemas legados. Para outras empresas, no entanto, o facto de ser preciso reimplementar os seus processos de negócio para implementar ERP pode resultar em problemas graves para a empresa. O ERP serve também como uma aproximação ascendente desde que a sua implementação parta dos processos elementares do negócio. Os indivíduos dentro da organização não podem seleccionar os seus próprios (internos e originais) processos de negócio para uso no novo sistema, em vez disso têm de aceitar o padrão proposto pelo ERP para os processos de negócio. Isto pode levar à resistência dos colaboradores da organização em o utilizar.

Os sistemas ERP internos da empresa são frequentemente implementados de acordo com o padrão dos dados, processos de negócio e lógica do negócio reimplementada. O insucesso e insuficiência

para alcançar os seus objectivos, pode ser atribuído ao facto dos sistemas ERP não cobrirem todos os requisitos das tecnologias de informação, e não irem ao encontro dos processos de negócio. [14]

Os benefícios de adoptar um sistema ERP deve-se ao facto que requer que uma empresa seja orientada ao processo e que todas as unidades de negócio internas utilizem o mesmo preciso processo. Como resultado, oferece uma infra-estrutura construída em torno de actividades que adicionam eficiência e eficácia à organização.

Os sistemas ERP suportam processos genéricos e as melhores práticas de organizações, possibilitando frequentemente a adaptação e configuração dos mesmos de forma a melhor suportar os processos e estratégias de negócio das organizações que a eles recorrem. Mas a personalização é uma tarefa difícil, que causa sérios problemas de integração, porque os pacotes ERP são complexos, pouco flexíveis e muitas vezes não concebidos para colaborar com outras aplicações. À medida que as organizações foram instalando sistemas de informação incompatíveis entre si, tornou-se mais complicada a integração entre organizações. [22] A grande preocupação das organizações actualmente é integrar os seus sistemas de informação internos com os sistemas de informação das outras organizações. Esta integração entre organizações é também designada por *B2B Integration*. [4]

Embora existam diferentes formas de integrar os sistemas de informação – ao nível da interface, dos métodos, dos dados, etc. – todas se baseiam na forma como os dados são trocados entre as organizações. As diferenças residem na forma como a troca de dados é efectuada, no tipo de dados utilizados, nas garantias oferecidas para a troca de dados, etc. [12] É neste contexto que uma aproximação para obter integração foi estudada e está em utilização por muitas grandes empresas, o EAI, que irá ser abordado em pormenor na *Secção 2.2*. O modelo de integração do EAI encontra-se representado na *Figura 2.2*.

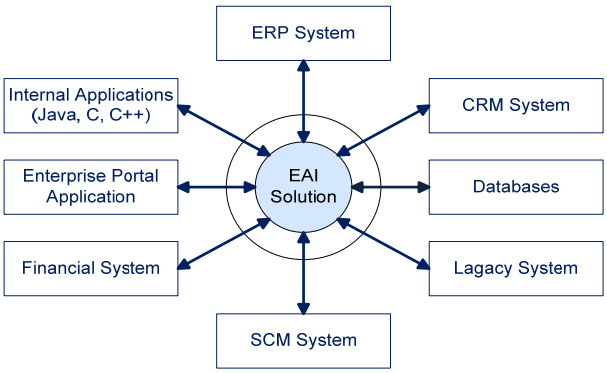

Figura 2.2 - Modelo de Integração de Aplicações Empresariais (EAI) [15]

2.2.1. Sistemas Legados

A grande discussão sobre os sistemas e dados legados centra-se frequentemente em torno da manutenção do código antigo, e conseguir que um sistema existente sobreviva a um melhoramento

de hardware, sistema operativo, ou base de dados. Na maioria das vezes, os sistemas legados foram desenvolvidos por uma geração anterior de programadores, dai o termo legado. O código é normalmente volumoso e complicado de modificar. Mas actualmente, os sistemas legados devem ser projectados para serem capazes de integrar com outras aplicações noutras empresas. [15]

Substituir os sistemas legados é uma estratégia de negócio arriscada pois raramente existe uma especificação completa do sistema legado, que adicionalmente foi tendo evoluções incrementais ao longo dos anos. Consequentemente, não existe uma forma directa de especificar um novo sistema, que seja funcionalmente idêntico ao sistema que está a ser usado; os processos de negócio e a forma como os sistemas legados funcionam são na maioria das vezes inextricáveis. Se o sistema é substituído, estes processos terão também que mudar, com custos e consequências potencialmente imprevisíveis; importantes regras de negócio podem ser integradas no *software* e no entanto sem serem documentadas; desenvolver *software* novo é arriscado porque podem existir problemas inesperados com o novo sistema e a integridade do sistema legado não pode ser comprometida de maneira nenhuma na implementação do novo sistema. [15]

O desafio de integrar sistemas legados é compreender a funcionalidade, o projecto, as operações e o desempenho do sistema e antecipar que tipo de alterações são necessárias no processo de integração. Para isso, existem duas aproximações para reutilização dos sistemas legados: integração e reimplementação.

Reimplementar ou reestruturar o código dos sistemas legados requer que o sistema e o código estejam bem documentados e/ou possam ser automaticamente analisados e transformados por um processo automático. Reimplementar um sistema é lento e dispendioso. Já a integração de uma sistema legado é mais rápida e mais barata que reimplementar. Para integrar um sistema legado, deve-se definir o papel de cada subsistema, definir interfaces para cada subsistema, e construir um objecto invólucro (*wrappers*) para cada subsistema.

Uma estratégia de integração pode ser intrusiva ou não intrusiva. Uma integração intrusiva requer conhecimento das relações internas dos sistemas legados, enquanto que uma integração não intrusiva requer conhecimento das relações externas dos sistemas legados. Se uma ligação a um sistema legado é efectuada sem necessidade de se alterar o mesmo, ou com alterações mínimas, essa ligação é considerada não intrusiva. Se o código da aplicação de origem é modificado, a conexão é considerada intrusiva. Existem duas aproximações para a integração dos sistemas legados e conseguir interoperabilidade: integração das aplicações e integração dos dados. [13] [6]

2.2.1.1. Integração das Aplicações

Nesta aproximação as aplicações contêm a lógica de negócio da empresa, e a solução passa por preservar essa lógica de negócio estendendo a interface das aplicações para interoperarem entre elas, ou com novas aplicações. Algumas das anteriores soluções de integração das aplicações utilizadas são:

- *Modernização da interface do utilizador:* Esta é a parte mais visível do sistema. Ao modernizar a interface do utilizador melhora-se a forma de a utilizar, o que pode ser extremamente

apreciado pelos utilizadores finais. Uma técnica comum para modernizar a interface do utilizador é o *screen scraping* que consiste em envolver a velha interface com novas relações gráficas. [24]

A verdade, no entanto, é que os utilizadores geralmente detestam a mudança, seja ela qual for.

- *Integração Ponto-a-Ponto:* Nesta solução, os canais de comunicação são desenvolvidos entre pares de aplicações. Esta solução é cara, pois resulta a prazo num *esparguete* de aplicações, que aumenta a complexidade da solução de integração assim como o número de aplicações ligadas, e tende a crescer exponencialmente. Mesmo o impacto de pequenas alterações na comunicação ou adição de uma nova aplicação é significativo. A manutenção desta solução é um problema devido ao elevado número de nós que pode existir. [23]
- *Routers de mensagens:* O número de interfaces pode ser reduzido significativamente através do uso de um *middleware*[3] – baseado em mensagens ou baseado numa arquitectura que permita a interoperabilidade entre aplicações em ambientes distribuídos heterogéneos (CORBA – *Common Object Request Broker Architecture*). Esta solução requer a utilização de adaptadores para ligação de cada aplicação ao *broker*. Cada aplicação tem somente uma ligação a esse *broker*, e comunicam publicando mensagens nesse *broker* que a entrega apenas aos subscritores interessados nela. O *middleware* pode também garantir uma entrega, que essa entrega seja certificada, a transformação da mensagem (utilizando o *broker*), entre outros aspectos. [2][24]
- *Integração CGI (Common Gateway Interface):* é um padrão para conectar aplicações externas com servidores, tais como http ou servidores *web*. A integração de sistemas legados através do CGI é utilizada para fornecer acessos rápidos à *web* aos recursos existentes, incluindo monitores de transacções e *mainframes*. [24]

Embora existam todas estas soluções, a solução mais adoptada actualmente, e com perspectivas de crescimento, é a utilização de arquitecturas orientadas a serviços (SOA – *Service Oriented Architecture*), tirando partido de tecnologias de integração tais como os *Web Services*. Com é referido em [27], o SOA é uma metodologia nova que permite às empresas aproveitar as aplicações já existentes, inclusive em sistemas legados, para integração com os novos processos de negócio por meio de utilização de *Web Services*. O SOA disponibiliza as aplicações de uma empresa através de uma arquitectura orientada aos serviços, onde esses serviços se abstraem dos processos de negócio da empresa. Isto permite a reutilização de funcionalidades já existentes na empresa, sem causar muitos transtornos à empresa. Isto torna-se importante para o desenvolvimento e crescimento de uma empresa, onde os processos de negócio podem ser melhorados e aumentados e incorporados sem grandes dificuldades.

[3] *Middleware* é a designação genérica utilizada para referir os sistemas de software que se executam entre as aplicações e os sistemas operativos. O objectivo do *middleware* é facilitar o desenvolvimento de aplicações, tipicamente aplicações distribuídas, assim como facilitar a integração de sistemas legados ou desenvolvidos de forma não integrada.

Como é referenciado em [7], os *Web Services* oferecem um conjunto de normas baseadas na Internet, bastante simples e populares, para integração de aplicações e sistemas de informação. Disponibilizam uma forma standard de comunicação, independente da plataforma e linguagem de programação utilizada, ajudando no processo de integração entre empresas parceiras. Para tal utilizam a linguagem WSDL, que utiliza o XML para descrever a interface dos serviços disponibilizados aos clientes, abstraindo de aspectos de implementação dos mesmos. É através do WSDL que as empresas clientes têm acesso aos serviços da empresa com quem pretendem comunicar.

2.2.1.2. Integração dos Dados

Nesta aproximação o importante é os dados, e são eles que são trocados entre organizações. A lógica de negócio implicada nos dados e metadados pode ser facilmente e directamente manipulada pelas aplicações na nova arquitectura da empresa. Algumas das soluções de integração de dados são:

- *Integração XML:* O padrão XML é um formato que foi amplamente adoptado para estruturar os documentos e dados na *web*. O XML é um formato de texto simples e flexível derivado da metalinguagem SGML (*Standard Generalized Markup Language*) e desenvolvido pelo W3C. O XML, com o passar do tempo está a transformar-se numa solução para integração dos dados, como referido posteriormente na *Secção 2.4.2*. O XML tem a desvantagem de só resolver o problema sintáctico na integração dos dados. Actualmente o grande problema centra-se em aspectos semânticos, estando a ser desenvolvidas tecnologias como *Semantic Web Services* [8] e linguagens de definição de ontologias, tal como OWL [9], de forma a solucionarem este problema.

  Esta foi a solução adoptada para este trabalho como explicado na *Secção 4*. [2][24]
- *Replicação dos Dados:* É o processo de copiar e manter os objectos da base de dados em múltiplas bases de dados, que compõem os sistemas de bases de dados distribuídos. A replicação dos dados fornece acesso rápido e local aos dados partilhados aos utilizadores e grande disponibilidade para as aplicações, pois existem outros modos de acesso aos dados. Mesmo que um local esteja indisponível, os utilizadores podem continuar a efectuar operações sobre bases de dados noutro local. A replicação das bases de dados é usada frequentemente para acesso descentralizado a dados legados armazenados em *mainframes* ou outros sistemas. Uma variante deste mecanismo consiste em ter bases de dados locais em todos os postos de trabalho dos colaboradores de uma organização, que são sincronizados regularmente com uma base de dados central. [2]

2.2.2. Integração das Aplicações Empresariais

A integração do negócio tornou-se um aspecto chave para muitas empresas para expandir o seu mercado, integrando os processos tanto internamente como com os parceiros.

Para alcançar este aspecto, o mercado emergiu para uma solução que os pode ajudar a alcançar uma melhor integração do negócio o qual é referida como EAI.

Inicialmente o EAI apenas focava a integração de sistemas ERP com outras aplicações dentro da empresa mas agora é geralmente usado para cobrir todos os outros aspectos da integração do negócio.

Em conjunto com o crescimento explosivo da Internet, expansão complexa do negócio, pressões competitivas, novos modelos de negócio, e a necessidade de "aproximar" os processos de negócio tanto internamente como com os parceiros e fornecedores, tornou a integração do negócio um aspecto importante para muitas empresas.

A maioria das empresas investiu em pacotes e aplicações personalizadas que executam funções de negócio específicas. Por outro lado, o objectivo principal da integração do negócio é ter aplicações dentro e fora da empresa a evoluir independentemente, mas que consigam utilizar as funcionalidades umas das outras. O maior desafio na integração do negócio é a interacção, que pode ser definida como consistindo na interoperabilidade e integração com as aplicações internas ou externas à empresa. [10]

As estratégias mais difundidas para conseguir interoperabilidade são através de integração dos dados ou disponibilizando uma interface da aplicação (estas estratégias são abordadas em pormenor na *Secção 2.2.1.1* e *Secção 2.2.1.2*). [6]

Como as aplicações empresariais são compostas por componentes autónomos, heterogéneos, e distribuídos oferecem por sua vez um desafio, devido a aspectos como a escalabilidade, volatilidade, autonomia, heterogeneidade, e sistemas legados. A integração do negócio e o EAI também necessitam de conversão da variedade de representações dos dados entre sistemas intervenientes e de ligar as fontes dos dados proprietários/legados, ERP, aplicações, processos e sistemas dos parceiros. [10]

Uma solução baseada na integração das aplicações envolve o transporte e transformação da informação entre uma ou mais aplicações. Suporta também regras de sincronismo temporal quando ocorre a transformação e transporte, assim como integridade, que determina o sucesso ou falha da integração. Numa perspectiva mais técnica, o artigo [23] propõe que o EAI é alcançado em 3 camadas de integração: a camada de transporte, a camada de transformação, e a camada de automatização dos processos. A camada de transporte transfere a informação da aplicação origem para a infra-estrutura de integração, e mais tarde dessa mesma infra-estrutura para a aplicação de destino (correspondente à Camada de Comunicação abordada na *Secção 5.2*). A camada de transformação traduz a informação do formato da aplicação de origem para a estrutura utilizada pela aplicação de destino. A camada de automatização dos processos integra os processos de negócio e controla o mecanismo de integração. As camadas de transformação e automatização correspondem à Camada de Comunicação com a Base de Dados e Conversão de Dados deste trabalho, abordada na *Secção 5.4*.

Muitas empresas utilizam o EAI como software de integração que permite uma integração mais eficaz tanto dentro das organizações como com organizações externas. Ao fazer isso, podemos agora incorporar funcionalidades de diversas aplicações. A Integração entre aplicações combina tecnologias de integração tradicionais (ex. *middleware* orientado às bases de dados) com tecnologias EAI (ex. adaptadores, *Message brokers*[4]) e abordagens SOA de forma a suportar eficientemente a interligação dos diversos sistemas de informação. Desta forma, a integração das aplicações resulta na interligação de dados, objectos e processos, bem como integração de aplicações cliente, sistemas e soluções de negócio electrónico. As principais aproximações EAI variam do EDI (Figura 2.3), aos *Web Services*[5], com integração dos processos através do XML (*Secção 2.3.2 - XML*). [6][10][25]

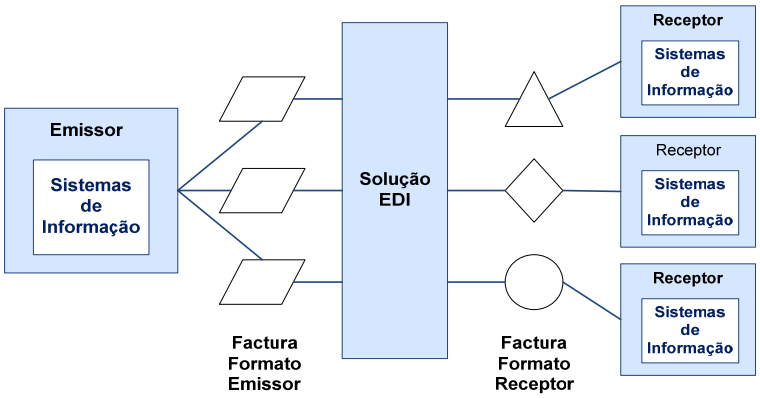

Figura 2.3 - Exemplo da utilização de tecnologias EDI para compatibilidade do formato das facturas entre diferentes sistemas de informação.

Mas cada vez mais se assiste a uma grande vontade dos utilizadores de disporem de arquitecturas flexíveis e capazes de endereçar um largo conjunto de problemas de negócio de uma forma mais rápida do que as tecnologias no âmbito do EAI. Foi neste contexto que o modelo EAI é, cada vez mais, substituído pelo SOA. A utilização deste modelo de arquitectura orientada aos serviços e dos *Web Services* são o principal motivo. À medida que os *Web Services* se tornam mais inteligentes, mudam também as decisões de negócio que são tomadas e implementadas. O SOA está estrategicamente centrado numa intersecção entre o negócio e a tecnologia, pelo que permite às empresas adaptarem-se rapidamente às mudanças de mercado.

Esta arquitectura permite aos departamentos de tecnologias de informação acompanharem os imperativos de negócio à medida que se automatizam os processos de forma totalmente separada das tecnologias. Desta forma, acaba por criar flexibilidade ao nível do próprio negócio permitindo um maior controlo na forma como as empresas efectuam negócios com os parceiros. [26]

---

[4] *Message Broker* é uma tecnologia que serve de intermediário de integração entre várias aplicações ou sistemas de informação.

[5] *Web Services* – Componente que permite às aplicações enviar e receber dados em formato XML, na comunicação entre aplicações diferentes.

2.3. Formato dos Documentos

A importância de interligação de sistemas de informação referida anteriormente, no contexto das transacções realizadas na facturação electrónica, torna a estruturação dos dados muito importante.
Um dos requisitos do negócio electrónico – no caso deste trabalho, mais especificamente na parte de facturação electrónica – é que os dados sejam estruturados como forma de facilitar as transacções.
Inicialmente o mercado electrónico utilizava tecnologias como o EDI (intercâmbio electrónico de dados), EDIFACT ou envio de faxes para ligação entre empresas. Mas o custo de implementação e manutenção da utilização das tecnologias EDI e EDIFACT revelou-se bastante elevado, havendo por exemplo a necessidade de estabelecer linhas dedicadas para o seu uso. Este aspecto dificultou a integração da maioria das empresas, as PME, visto não poderem suportar os custos inerentes a estas tecnologias.
A utilização de tecnologias como o EDI, XML, ebXML, entre outros, na troca de mensagens estruturadas entre empresas de forma normalizada, dá um contributo importante para a aceitação e facilitação de práticas de facturação electrónica. A omnipresença da Internet dá outro contributo importante a esta prática.

2.3.1. EDI

O EDI significa a transferência de mensagens normalizadas entre sistemas informáticos de parceiros comerciais, utilizado com um mínimo de intervenção humana. Utiliza um formato de dados estruturado e padronizado que permite que os dados sejam transformados para outros formatos sem serem reintroduzidos.
O seu objectivo fundamental é possibilitar que dois parceiros de negócio com aplicações de gestão e respectivos formatos dos dados diferentes, possam trocar mensagens de negócio padrão, que permitam a sua automática conversão para formatos internos e consequente integração aplicacional, como referido em [1]. Os documentos em EDI estão normalmente comprimidos e dependem de computadores para a sua interpretação. Os documentos EDI não podem ser considerados informação mas apenas dados estruturados, podendo ser facilmente interpretados de forma eficaz, sem ambiguidades, pelas aplicações informáticas [1].
A utilização principal do EDI é transferir transacções de negócio repetitivas tais como: encomendas, facturas e notificações de envio. O EDI não implica comunicação em tempo real.
Assente no princípio de relação contratual prévia, como referido em [16], o EDI utiliza uma série de mensagens e elementos padrão que são trocadas, e ligações [1] ponto-a-ponto, redes privadas, *intranet* ou VANS (*Value Added Networks*) que resultam em custos elevados. Para além desse facto, por cada novo parceiro de negócio e formato diferente, o EDI exige um esforço de mapeamento da informação. Estas redes privadas ou VANS acabam por resultar em custos muitos elevados de

manutenção dos servidores e da rede prestadora, o que dificulta a adopção por parte das PME. As PME não são capazes de suportar o custo que isso implica, dificultando a integração, e consequentemente a troca de mensagens entre empresas.

Nas relações tradicionais, construídas em cima de ligações de longa duração com um número limitado de actores, o EDI provou ser uma solução de sucesso. Mas quando se tentou adequar estas soluções para ligações das empresas através da *Internet*, verificou-se um dos problemas da utilização do EDI, muitas vezes relacionado com a flexibilidade limitada, o que torna estas soluções inviáveis.

As relações baseadas em EDI sofrem de fraca adaptação a novas tendências e desenvolvimentos. Isto é uma consequência que advém do facto de o EDIFACT se ter proposto como a norma para todas as áreas de negócio.

Uma consequência de falta de flexibilidade é que as soluções de EDI tradicionais são susceptíveis de manter os processos de negócio de uma empresa. Mudanças nos processos de negócio implicam sempre mudanças na comunicação EDI. Devido ao elevado custo de implementação e aposta em relações de longo prazo, levam a que os padrões de comunicação EDI existentes condicionem, até certo ponto, o desenvolvimento dos processos de negócio. Por isso os padrões EDI limitam o desenvolvimento dos processos existentes.

Os dois padrões principais mais antigos, considerados hoje como tradicionais, sendo também os mais utilizados na Europa e Estados Unidos, são o X12 e o EDIFACT.

2.3.1.1. EDIFACT [1]

O EDIFACT define as regras para troca electrónica de informação para administração, transporte e negócio. A comunicação geralmente é realizada com o emissor gerando a mensagem, convertendo-a para uma linguagem intermédia e enviando-a para um receptor, que a converterá finalmente para o formato específico da sua aplicação. As mensagens não são traduzidas por humanos e sim por sistemas.

O esforço do EDIFACT por fazer uma norma válida para todos tipos de segmento de actividade fez com que a norma viesse a ter muitas variantes / subconjuntos. Há uma grande redundância e a implementação da comunicação electrónica tem que ser muitas vezes seguida de ajustes ao padrão. Isto criou uma situação em que a norma se tornou demasiado generalista para responder às exigências dos utilizadores. Outro problema existe quando se actualiza a versão para uma nova (geralmente são anuais), pois obriga-se as empresas a mudar o formato dos dados que enviam e recebem.

O EDIFACT pode ser dividido em catálogos e regras de sintaxe. A sintaxe pode ser descrita como uma estrutura gramatical enquanto que o catálogo descreve as diferentes partes das mensagens e segmentos padrão e elementos.

O mais importante de uma mensagem EDIFACT é o elemento “dados”. Os elementos “dados” são precisamente a informação que vai ser enviada ou recebida. Os elementos são divididos em três tipos de categorias: obrigatórias, opcionais e condicionais. Os elementos de “dados” não são

habitualmente descritos em texto simples mas numa espécie de código. Por causa do tratamento automático isto melhora muitas vezes a eficiência das transacções.

2.3.1.2. ANSI X12 [1]

O ANSI X12 foi desenvolvido no final dos anos setenta. Assim como o EDIFACT, o objectivo do X12 era desenvolver uma norma uniforme para EDI. Esta norma é semelhante ao EDIFACT em muitos aspectos. A diferença reside na estrutura dos elementos de dados: no ANSI X12 não existem elementos de dados compostos, o que permite uma estruturação diferente dos elementos de dados e na parte final das mensagens EDI.

Esta norma define como os dados devem ser estruturado, quais os documentos que podem ser transmitidos, quais os dados que devem ser incluídos em cada documento e os identificadores de cada documento.

Para cada documento existe um código chamado *transaction set* a ser utilizado por esse documento.

2.3.2. XML

Desde a sua especificação em 1998 pelo Consorcio W3C, o XML rapidamente adquiriu grande popularidade, tendo já sido firmemente integrado no negócio electrónico. O XML tornou-se na sintaxe preferida para troca de dados em ambientes de integração de aplicações empresariais devido ao seu formato universal para partilha de dados entre aplicações, que podem estar escritos em diferentes linguagens de programação ou correrem em diferentes ambientes, como é explicado em [11], e pela sua grande legibilidade.

O XML facilita declarações precisas, não só dos conteúdos de um documento, como também dos elementos convenientes à estruturação desses conteúdos. Ao permitir ao autor do documento definir os seus próprios elementos, criando e usando as etiquetas[6] mais adequadas à descrição e à estruturação dos dados, serve perfeitamente para estruturar os dados e para descrevê-los em forma de texto, como se pode verificar através de [11].

As etiquetas podem ser utilizadas para associar a descrição do formato aos dados de um documento, assim como podemos descrever o conteúdo e fixar a estrutura lógica desse mesmo conteúdo. Novas etiquetas podem ser adicionadas por necessidade de extensão. A interpretação das mensagens depende das etiquetas e não da posição do campo na mensagem.

Num documento em formato XML os dados são descritos em texto (*Unicode*). Contudo, um formato de texto não é a solução mais económica para armazenar dados, mas muitas vezes é a mais eficaz pois é mais fácil trabalhar com um texto, do que trabalhar com os formatos binários típicos de muitas bases de dados e mesmo do formato EDI. Outra das vantagens de toda a descrição ser textual é que

[6] Etiquetas ou *tags* são estruturas de linguagem de marcação, que consistem em breves instruções, tendo uma marca de início e outra de fim. São actualmente utilizadas como delimitadores de estilo e/ou conteúdo, tanto em HTML como em XML.

resolve muito dos problemas de portabilidade, tornando-o independente das plataformas e ainda o não ser bloqueado por *firewalls*[7]. Os documentos são reutilizáveis visto que o mesmo ficheiro pode proporcionar diferente informação dependendo das necessidades. [1]

Como referido em [7], o XML é assumido por muitos como a linguagem para formatar mensagens. Através da utilização do XML, uma empresa pode criar novos formatos de mensagens para encomendas, facturas ou qualquer outro tipo de documentos. Esta flexibilidade permite às empresas adequar o formato das mensagens de acordo com as suas particularidades de negócio. Em contrapartida obriga a utilização de ferramentas e/ou conversores para introdução destas mensagens no sistema de informação empresarial e/ou obtenção das mesmas no sistema de informação empresarial. Mas o XML é flexível e menos dispendioso de implementar pois existem numerosos programas gratuitos para interpretação dos dados, e o seu modelo da informação é suportada por imensas ferramentas, podendo desta forma transportar variados tipos de dados e mantê-los estruturalmente coesos.

Para a interpretação de documentos XML existem duas *API's* que podem ser utilizadas: o DOM (*Document Object Model*) e o SAX (*Simple API for XML*). O DOM permite construir uma representação em árvore dos dados, e o SAX permite que os dados sejam processados como uma *stream* de eventos, com processamento sequencial, o que é especialmente útil no processamento de mensagens de grandes dimensões.

Existe um elevado número de *frameworks*[8] de padrões XML para utilização no negócio electrónico, como o ebXML, RosettaNet, HL7, BizTalk Framework, entre outros. Esta variedade de *framewoks* revela um novo problema para as aplicações e para as empresas, já que não existe um padrão único para os documentos trocados entre empresas. As aplicações precisam de converter os dados dos documentos de outros padrões para o padrão XML. Para isso, utiliza-se geralmente a linguagem XSL-T[9] que permite controlar o modo como os dados são representados e a conversão dos dados para documentos XML. [1]

Devido à flexibilidade e suporte do formato XML para as transformações automáticas dos dados, assim como pela sua capacidade de passar pelas *firewalls*, o XML é bastante utilizado na integração de aplicações empresariais, assim como para trocas entre empresas no contexto do negócio electrónico.

Como referido em [3], enquanto que o EDI define um formato fixo para os documentos nas comunicações, o XML está associado a um formato flexível. Embora o XML seja inerentemente menos eficiente, em conjunto com a adição de etiquetas e delimitadores, o interesse na flexibilidade e

---

[7] *Firewalls* são mecanismos que lidam com os riscos inerentes à ligação de redes privadas de organizações a outras redes não controladas, nomeadamente a *Internet*.

[8] *Frameworks* são conjuntos de metodologias de programação e código base testado e reutilizável para implementação de projectos.

[9] XSL-T – Extensible Stylesheet Language Transformations

tempo de vida da entrega dos dados numa comunicação entre empresas que necessitam de integração, “esconde” os custos adicionais.
Estes factores levaram à escolha do XML como o formato para a construção de um repositório de metadados, explicado melhor na *Secção 4.3*, bem como o formato normalizado utilizado para troca de mensagens entre as organizações, explicado na *Secção 5.2.1* e *Secção 5.3*.

2.3.2.1. EBXML

Este projecto foi desenvolvido no ano 1999 com o objectivo inicial de construção de um modelo de especificação que permitisse às empresas independentemente da sua dimensão e localização, fazer negócio através da *Internet*.
É uma infra-estrutura aberta, assente em XML, que permite a troca de mensagens, constituição de relações, comunicação de informação e define e regista processos de negócio entre parceiros. O ebXML permite acelerar e facilitar os processos de integração com os sistemas de informação, ao permitir criar etiquetas para definição, transmissão, validação e interpretação entre aplicações e entre empresas.
O ebXML é um conjunto de especificações, que juntas oferecem uma *framework* modular para processos de negócio electrónico. A arquitectura do ebXML é baseada em padrões abertos, desenhados através de processos colaborativos. A arquitectura permite [1]:

- Estabelecer nível de transporte de mensagens uniforme: infra-estrutura que garanta interoperabilidade de comunicação de dados, através de mecanismos de transporte de mensagens padrão, com interfaces bem definidas, regras agregadas e modelos de segurança;
- Forma para registar e descobrir sequências de processos de negócio com troca de mensagens relacionadas. *Framework* de semânticas para garantir interoperabilidade comercial, definir processos de negócio e modelo de informação e conjunto de componentes reutilizáveis e vocabulário XML através de Meta-modelos;
- Definir perfis de empresas e acordos entre parceiros de negócio: Mecanismo para permitir organizações encontrar outras e estabelecer acordos de relacionamento de negócio electrónico através de partilha de um repositório de dados com perfis de empresas e modelos de processos de negócio e processo para definir e formalizar CPA (Collaboration Protocol Agreement).

2.3.2.2. RosettaNet

A RosettaNet é uma organização não lucrativa que pretende implementar um padrão para transacções IT (*Information Technology* – tecnologias de informação) na *Internet*. É um conjunto de mecanismos padrão (processos) que permitem às companhias acordar no processamento de transacções de negócio (neste caso o XML foi adicionado para tomar o lugar do EDI como padrão de

troca de dados). É o padrão mais sofisticado disponível pois tem em conta, ao mesmo tempo, os processos e os dados, definindo processos comuns e pontos de integração numa industrial vertical.
É fornecido um dicionário mestre que define propriedades para os produtos, parceiros e transacções de negócio, o qual, acoplado a uma *framework* de implementação, pode ser usado para suportar o diálogo de integração das aplicações no negócio electrónico entre empresas (B2B).
Como referido em [5], os dois dicionários existentes no RosettaNet dizem-nos o significado das etiquetas XML e estabelecem uma troca comum com XML e DTD's[10]. O dicionário de propriedades técnicas, que providencia especificações técnicas para todas as categorias de produtos, e o dicionário de propriedades de negócio que descreve os atributos usados para definir as propriedades da cadeia de valor das companhias e respectivas transacções de negócio.
Também referido em [5], os aspectos mais importantes no RosettaNet são os processos de interface entre parceiros (PIP) comuns, que fornecem modelos de dados empresariais comuns e documentos, permitindo o desenvolvimento de sistemas que implementam interfaces de negócio electrónico RosettaNet. O PIP torna possíveis os processos de negócio electrónico, isto é, a ferramenta pela qual um processo empresarial electrónico é transmitido. São na verdade contratos ou acordos que cada organização adere.
Uma aproximação como a RosettaNet diminui o risco de um padrão disperso, mas por outro lado, há perigo que as companhias envolvidas fiquem reféns da sua própria evolução nos processos predefinidos. A principal vantagem deste padrão é que não há confusão sobre que processos fazem o quê e onde.

2.4. Adaptadores

Adaptadores ou conectores são pedaços de software que são usados na integração de aplicações e servem como mediadores para acesso a aplicações que não foram pensadas para integração, tais como sistemas legados.
Durante anos foram construídos adaptadores para *Message Brokers*, mas sempre de acordo com os interesses das empresas que os desenvolviam, não existindo um modelo padrão. Estes adaptadores são camadas entre os *Message Brokers* e as aplicações de origem ou de destino. Segundo [2], "Os adaptadores podem ser, por exemplo, um conjunto de "bibliotecas" que mapeiam as diferenças existentes entre duas interfaces distintas – escondendo a complexidade dessas interfaces para os utilizadores ou mesmo para os programadores da aplicação de integração entre as empresas que usam o *Message Broker*".
Actualmente, e no contexto de integração entre as empresas, os adaptadores utilizados estão a tornar-se cada vez mais "inteligentes", facilitando assim a captura de eventos na execução dos sistemas intermediários. Estes adaptadores inteligentes são designados para tratar muitas das

[10] DTD ou Document Type Definition - Definição de Tipo de Documento

tarefas dinâmicas necessárias na integração das aplicações, usualmente tratadas tanto no *broker* de integração ou pelo programador, tal como actualizar o código do adaptador quando uma aplicação na empresa é actualizada.

2.4.1. Tipos de Adaptadores

Como referido em [2] existem dois tipos de adaptadores no contexto dos *Message Brokers*: adaptadores finos e adaptadores densos; estes adaptadores têm ainda dois tipos de comportamento, em que podem ser dinâmicos ou estáticos. A descrição dos tipos de adaptadores e seus comportamentos encontra-se nas secções seguintes.

2.4.1.1. Adaptadores Finos (Thin Adapters) [2]

Estes adaptadores são na maioria dos casos API's que mapeiam a interface do sistema de origem ou destino para uma interface comum, suportada pelo *Message Broker*, como se pode verificar na Figura 2.4. Por outras palavras, este tipo de adaptadores não disponibiliza uma camada entre os sistemas, sendo apenas uma simples abstracção.

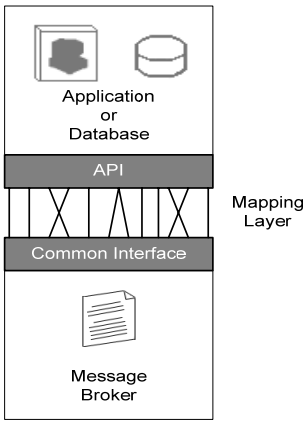

Figura 2.4 - Os adaptadores finos são apenas uma simples abstracção no topo da API existente. [2]

Embora estes adaptadores sejam simples de implementar, têm a desvantagem de causar um grande impacto no desempenho mas sem acrescentar funcionalidades. Isto deve-se ao facto de apenas ser negociado o software de interacção entre os sistemas intervenientes, necessitando ainda de programação. Outro aspecto que complicado é o facto das API's mapeadas serem na grande maioria proprietárias.

2.4.1.2. Adaptadores Densos (Thick Adapters) [2]

Estes adaptadores disponibilizam software e funcionalidades entre a infra-estrutura do *Message Broker* e a aplicação de origem ou destino. Tornam, também, a gestão da movimentação de informação ou invocação dos processos menos complexo e demorado, devido à sua camada de

abstracção, representada na Figura 2.5. Essa camada de abstracção, em conjunto com o gestor responsável pela negociação das diferenças entre as aplicações a integrar, facilita a tarefa de integração, reduzindo a necessidade de alteração e implementação de código.

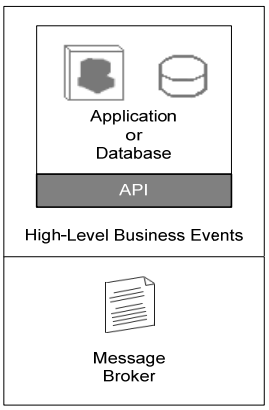

Figura 2.5 - Representação da utilização de uma camada de abstracção no topo da interface da aplicação. [2]

A camada de abstracção permite que a complexidade das aplicações intervenientes seja transparente para o *Message Broker*. Desta forma, o utilizador apenas vê uma representação dos processos e informação dos metadados como sendo controlado pela camada de abstracção e o adaptador. O utilizador liga os sistemas entre si através dessa camada de abstracção e da interface do utilizador.
Um dos aspectos mais importantes no cenário destes adaptadores são os repositórios, pois permitem absorver grande parte da informação sobre as aplicações intervenientes, utilizando essa informação como mecanismo de interacção com essas aplicações em nome do *Message Broker*.
Embora as muitas vantagens referidas anteriormente e conveniência da utilização deste tipo de adaptadores, o seu progresso é lento pois requer muito tempo para desenvolvimento, chegando a ser seis vezes superior ao desenvolvimento dos adaptadores finos.
Com a crescente necessidade de integração das aplicações entre empresas, este tipo de adaptadores mais sofisticados torna-se uma necessidade para as empresas pois representam uma solução de fácil implementação e reduzida porção de código, disponibilizando um método fácil para integração das empresas.

2.4.1.3. Adaptadores Estáticos

Estes adaptadores têm de ser implementados manualmente de acordo com os requisitos e especificidades dos sistemas que os utilizam. Não têm mecanismos para compreensão do esquema relacional das bases de dados, o que resulta numa configuração manual para receber informação do esquema da origem dos dados. Qualquer alteração ou modificação no esquema da base de dados a que está ligada não é reconhecido por estes adaptadores, pois não têm mecanismos para suportar estas actualizações. Este tipo de adaptadores apenas liga uma aplicação a outra, ou a um *broker* de integração, numa ligação ponto-a-ponto, impossibilitando a transformação de dados incompatíveis

entre diferentes aplicações. Contudo o programador não tem que trabalhar com todas as diferentes relações da aplicação, pois este tipo de adaptadores trata apenas da tradução básica. [2]

2.4.1.4. Adaptadores Dinâmicos

Estes adaptadores, também referidos como adaptadores inteligentes, são designados para tratar muitas das tarefas dinâmicas necessárias na integração das aplicações. Têm capacidade de aprendizagem sobre os sistemas com que interagem através da leitura de informação adquirida na ligação inicial à aplicação ou base de dados. É feita uma leitura da informação do esquema da base de dados no repositório ou código da fonte para determinar a estrutura, índice e semântica das aplicações dos sistemas ligados. Para além da aprendizagem sobre os sistemas legados, podem também reaprender, ao longo do tempo, quando ocorrem alterações nos sistemas ligados. [2]

2.4.2. Adaptadores Existentes no Mercado

2.4.2.1. iWay Software Universal Adapter Suite [55]

A *iWay Software* é uma empresa líder mundial no desenvolvimento de adaptadores, acelerando a integração do negócio, ao fornecer ferramentas que tornam os processos de integração fáceis de implementar. Tem-se destacado por desenvolver *middleware's* para a integração de sistemas, com o desenvolvimento de adaptadores inteligentes para ligar mais de 300 fontes de dados, entre elas bases de dados, aplicações, sistemas de transacções de dados, negócio electrónico, entre outras. Os adaptadores iWay simplificam o acesso a sistemas de ERP e gestão da relação com o cliente (CRM), encaminhamento electrónico, sistemas legados, protocolos de negócio electrónico como o ebXML, entre outros. Adicionalmente, a transformação de mensagens e integração de dados faz dos adaptadores da *iWay* uma escolha para a integração – exclusivamente ou com outro *middleware*.

Estes adaptadores vão ao encontro das exigências de soluções rentáveis e de fácil implementação, integrando aplicações díspares, transacções e dados através de múltiplas plataformas. Com os extensos conjuntos de adaptadores *iWay*, as empresas não necessitam de abdicar da sua arquitectura existente em prol de tecnologias actualizadas para ter uma solução integrada, porque a *iWay* suporta um grande leque de plataformas e padrões.

Estes adaptadores permitem uma arquitectura orientada a serviços (SOA) escalável e reutilizável, como se pode verificar na Figura 2.6, que acelera o desenvolvimento de múltiplas aplicações de negócio electrónico.

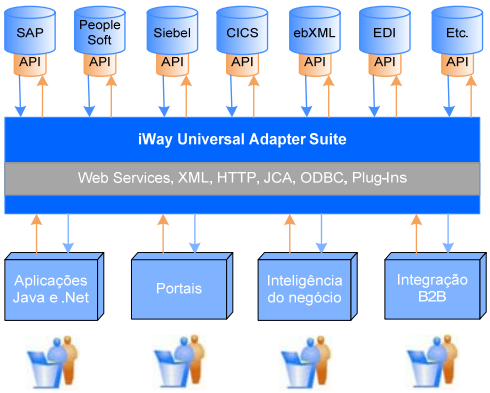

Figura 2.6 - Arquitectura orientada a serviços do adaptador *iWay Universal Adapter Suite* para integração. [55]

É possível também reduzir significativamente os custos de integração no negócio electrónico através de adaptadores prontos a utilizar, que integram rapidamente os dados de múltiplas plataformas e sistemas operacionais.

2.4.2.2. Attunity Connect [56]

A *Attunity Connect* disponibiliza adaptadores que permitem integração com muitas aplicações legadas e fontes de dados de uma forma fácil, nas diferentes plataformas de computação. Fornece uma forma mais intuitiva e fácil de integrar diferentes aplicações e dados, usando padrões como J2EE CA[11] e XML, além de interfaces de acesso a dados, como JDBC[12], ODBC[13] e SQL, como se pode verificar na Figura 2.7.

Fornece integração universal com suporte a integração tanto orientada aos dados, baseado no SQL, como orientado aos serviços, baseado no XML e *Web Services*. Ao fornecerem integração orientada aos serviços e eventos, permitem que aplicações, em qualquer ambiente computacional, interajam bi-direccionalmente com os adaptadores suportados. Este aspecto em conjunto com a utilização de adaptadores certificados, permite interoperabilidade com os programas legados e permite empacotar operações sobre dados como serviços reutilizáveis, usando XML, API's padrão e *Web Services,* acelerando a integração de aplicações empresariais e produtos de integração das aplicações.

[11] J2EE CA ou *J2EE Connector Architecture* – Solução tecnológica para conectar um servidor aplicacional a um sistema de informação da empresa. Considerado como um componente de integração de aplicações da empresa.

[12] JDBC ou *Java Database Connectivity* – É uma especificação do J2EE utilizado no acesso a bases de dados

[13] ODBC ou *Open Data Base Connectivity* – Tecnologia padrão de programação para acesso a bases de dados por meio de uma biblioteca de funções previamente definidas, criada pelo *SQL Access Group*

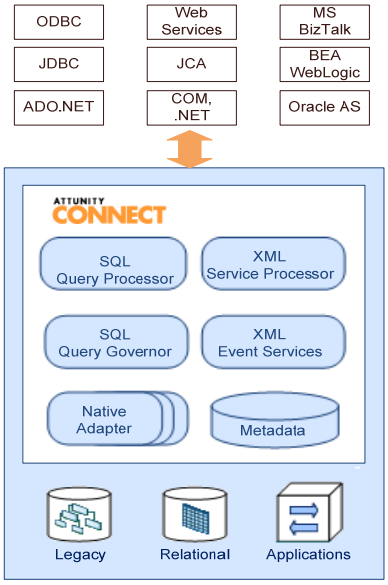

Figura 2.7 - Arquitectura funcional do *Attunity Connect* .*[56]*

Estes adaptadores geram eventos provenientes de cada aplicação legada, transformando-os automaticamente em mensagens XML que podem ser entregues às aplicações das empresas.
Não é necessário programação e os utilizadores utilizam uma interface para configurar e testar rapidamente os adaptadores.

2.4.2.3. LiveConnector da DataMirror

O *LiveConnector* é uma solução de integração de aplicações empresariais que oferece vantagens significativas em termos de custos de posse, rápida implementação e retorno do investimento. Esta solução suporta múltiplas plataformas de bases de dados (tais como Oracle, Sybase e Microsoft SQL Server) e permite a integração da base de dados com XML, para maximizar a flexibilidade e interoperabilidade.
O *LiveConnector* inclui *middleware* e adaptadores já construídos para as aplicações. Importa e exporta virtualmente transacções de negócio de e para qualquer pacote de aplicação, não obstante a plataforma, base de dados, estrutura das aplicações ou localização geográfica.
Uma vez executado, o *LiveConnector* fornece uma solução inteiramente automatizada que continuamente flúi os dados sem requerer nenhuma intervenção manual.
A solução *LiveConnector* não está limitada apenas à integração de ERP's e gestão do relacionamento com o cliente (CRM), mas também pode ser usada para complementar outros projectos de integração, tais como armazenamento de dados e negócio electrónico.

2.4.2.4. ICAN Suite eWay Intelligent Adapter da Sun SeeBeyond

O *ICAN Suite* é uma plataforma escalável e aberta para desenvolver, executar e gerir processos de negócio integrados como aplicações compostas. Integra nele negócio electrónico entre empresas, BPM[14] juntamente com *Web Services*, SOA, servidores aplicacionais J2EE, ETL[15], BAM[16], Gestor da Qualidade dos dados na empresa, portais, entre outros.

O componente *eWay Intelligent Adapters* do *ICAN Suite* acelera a integração com os *Web Services* disponíveis. A *Sun SeeBeyond* disponibiliza mais de 80 adaptadores *eWay Intelligent* que utilizam JCA[17] e *Web Services* para integração do negócio e aplicações.

Os produtos da *ICAN* utilizam estes adaptadores inteligentes com o intuito de cobrir um vasto conjunto de diferentes sistemas baseados em arquitecturas e protocolos distribuídos tais como J2EE, .NET, tecnologias legadas tais como CICS, CORBA e DCOM, e pacotes aplicacionais tais como Siebel, SAP, etc.

Estes adaptadores aceleram a integração e minimizam os custos de desenvolvimento e manutenção através de ferramentas orientadas ao utilizador, de configuração e distribuição. Cada adaptador *eWay* inclui lógica de negócio específico e validação de dados, construído dentro de mensagens electrónicas.

2.5. Considerações Finais

Este capítulo apresentou algumas das soluções desenvolvidas nos últimos anos para resolver os problemas de integração de sistemas de informação. Mostrou também que todas as soluções apresentam vários pontos fortes e algumas dificuldades e problemas, em que nenhum consegue realmente resolver o problema de integração que se pretende com este trabalho.

Desde o facto de ser necessário a utilização de inúmeros adaptadores para cada um dos sistemas com que se pretende comunicar, como o facto de acrescentar custos e ser necessário despender muito tempo para a sua implementação, estes e outros factores levaram a uma investigação da melhor forma de solucionar os problemas subjacentes com a integração de sistemas de informação legados e bastante diferentes.

Foram também estudos os diferentes formatos de dados existentes no mercado, e que se adequam ao comércio electrónico. De entre todos os formatos estudados, o XML foi o escolhido para troca de dados em ambientes de integração de aplicações empresariais devido ao seu formato universal para partilha de dados entre aplicações.

---

[14] BPM ou *Business Process Management* – Tecnologia que faz a gestão dos processos de negócio

[15] ETL ou *Extract, Transform and Load* – Processo de extrair dados de um sistema (base de dados), transformá-los de alguma forma e inseri-los em outra base de dados especial.

[16] BAM ou *Business Activity Monitoring* – Tecnologia que monitoriza as actividades de negócio em Tempo Real

[17] JCA ou *Java Connector Architecture* – Especificação do J2EE que é uma API que padroniza a ligação a aplicações legadas.

A solução que irá ser explicada ao longo deste trabalho é, de todos aqueles que foram referidos e descritos neste capítulo, aquele que melhor se enquadra no contexto de utilização num cenário de mediação electrónica. Embora existam algumas tecnologias e ferramentas algo semelhantes, ficou decidido criar um sistema personalizado mas à sua semelhança e partindo da sua implementação.

No capítulo seguinte irá ser feita uma abordagem ao Sistema ERP escolhido para o âmbito deste trabalho, o Primavera Express. Serão referidos aspectos de funcionamento, bem como algumas partes do seu modelo relacional utilizados neste trabalho. É também mostrado a relação entre o conteúdo de uma factura electrónica e o conteúdo da base de dados do Primavera Express.

3. Sistemas de Informação de Base

Nesta secção irá ser abordado o sistema de informação de base para a implementação e teste do adaptador. O sistema escolhido foi o Primavera Express, uma das soluções da Primavera BSS. A decisão por este sistema deveu-se ao facto da sua disponibilização ser grátis e ser relativamente fácil de utilizar, não deixando por outro lado de conter todas as funções importantes e necessárias para um Sistema ERP.

No final será mostrada a estrutura das tabelas utilizadas do sistema Primavera Express, bem como o seu conteúdo e respectiva correspondência com uma factura, neste caso a correspondência com a estrutura do documento electrónico trocado.

3.1. Visão Geral

Com as Resoluções e Decretos-Leis, anteriormente referidos na *Secção 2.1.3*, a procura de aplicações pelas empresas para o desenvolvimento do negócio está cada vez mais a aumentar, para que possam entrar na dinâmica do comércio electrónico. Mas os custos inerentes na adopção destas aplicações é um dos maiores entraves para os clientes, que têm receio do processo de digitalização seja demorado e consuma muitos recursos, bem como os seus gastos não tenham retorno.

Para as grandes e médias empresas este não é um factor de grande problema, conseguindo com alguma facilidade aplicações que lhes convenha e sem grandes entraves. Os grandes impulsionadores para o desenvolvimento de aplicações mais simples e 'baratas' são as pequenas e micro empresas (PME), que tentam entrar num mercado onde as grandes e médias empresas lideram. Regra geral, as PME contêm poucos recursos humanos e financeiros, o seu conhecimento em termos de tecnologias recentes nem sempre é muito, e esperam sempre que os investimentos realizados tragam o devido retorno financeiro.

Foi neste seguimento que a Primavera BSS (*Business Software Solutions*) disponibilizou para o mercado nacional um software de gestão totalmente gratuita, o Primavera Express. Como referido em [38], a Primavera BSS desde sempre adoptou mecanismos electrónicos assentes em plataformas desenvolvidas sobre *Internet*, que permitam a digitalização dos processos de negócio de uma empresa. Nesse sentido, disponibilizam soluções para todas as empresas, incluindo agora o Primavera Express para empresas que se estejam a iniciar no mercado nacional.

Como referido em [38], a Primavera BSS disponibiliza aos seus clientes soluções de Contabilidade, Recursos Humanos, Gestão Comercial, CRM, *Business Inteligence*, *Enterprise Portals* entre outras soluções, como pode ser visto na Figura 3.1, permitindo às empresas obter soluções nos vários sectores de actividades.

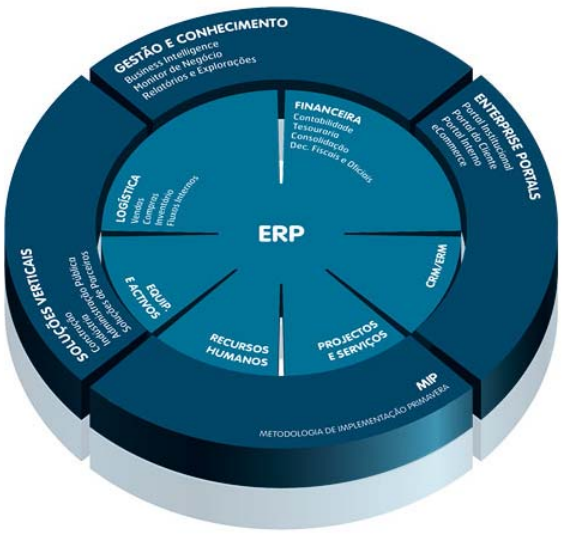

Figura 3.1 - Esquema Global da Soluçao PRIMAVERA. [38]

3.2. Solução/Produto

O Primavera Express foi desenvolvido orientado para o comércio e serviços, permitindo gerir várias áreas tais como as de vendas e recebimentos, de stocks, e o relacionamento com os clientes.

De forma a poder ser disponibilizado gratuitamente, algumas funcionalidades das outras soluções Primavera não são contempladas nesta solução mas, como referido anteriormente, contendo as funcionalidades necessárias para empresas em início de actividade. Como referido em [39] e anteriormente neste capítulo, uma empresa que utilize o Primavera Express tem acesso a soluções de gestão, tais como a gestão de vendas/pos, stocks, contas correntes e bancos de clientes e de facturação. Ao nível fiscal, esta é uma solução que cumpre todas as obrigações fiscais e legais, disponibilizando todos os relatórios fiscais a serem entregues pelas empresas às entidades oficiais. Toda a documentação electrónica utilizada por esta solução é guardada na base de dados do sistema. Tem como suporte de base de dados o *Microsoft SQL Server* (*Microsoft Desktop Engine/Microsoft Sql Server Express*). [39]

Como a sua base tecnológica é equivalente às restantes soluções, isto permite as empresas migrarem para outra solução/produto Primavera sem dificuldades. Também tem a vantagem de ser um produto de fácil instalação e utilização, através de manuais simples e descritivos fáceis de entender pelos clientes. De referir que esta solução funciona exclusivamente em ambientes mono-utilizador. [39]

3.3. Funcionalidades no Cenário de Mediação Electrónica

O Primavera Express tem uma série de funcionalidades abrangendo as seguintes áreas de funcionamento: Facturação/Venda ao Balcão, Caixa, Stocks, Informação de Gestão e outras características.

A área de maior interesse para este trabalho é a mediação electrónica, mais concretamente a Facturação Electrónica, e este software vai ao encontro do que é pretendido permitindo a emissão de vários documentos de vendas tais como: facturas, vendas a dinheiro, notas de crédito, notas de débito, guias de remessa e de transporte. Pode-se também proceder à emissão de facturas Pró-forma e propostas. Permite também o tratamento de devoluções. Ainda relacionado com a facturação, é permitido ao utilizador a escolha de múltiplos modos de pagamento, emissão de recibos e notas de pagamento, e actualização de stocks integrada com a facturação. [39]

Em relação a informação de gestão, de salientar a consulta de valores a pagar e receber e listagem de documentos emitidos. Outra característica também importante é a possibilidade de emissão de documentos com IVA incluído ou excluído. [39]

3.4. Tabelas

De entre as possibilidades de integração com outros sistemas, a escolhida foi através da integração/comunicação com a base de dados do Primavera Express.

As tabelas envolvidas no cenário de mediação electrónica, utilizadas pelo sistema ERP Primavera Express, estão representadas nas Figura 3.2 e Figura 3.3, correspondendo respectivamente às tabelas utilizadas para as vendas e às tabelas utilizadas para as compras. A informação obtida das tabelas utilizadas nas vendas é posteriormente convertida e enviada para o cliente, e a informação introduzidas nas tabelas utilizadas nas compras é obtida através do documento recebido do fornecedor.

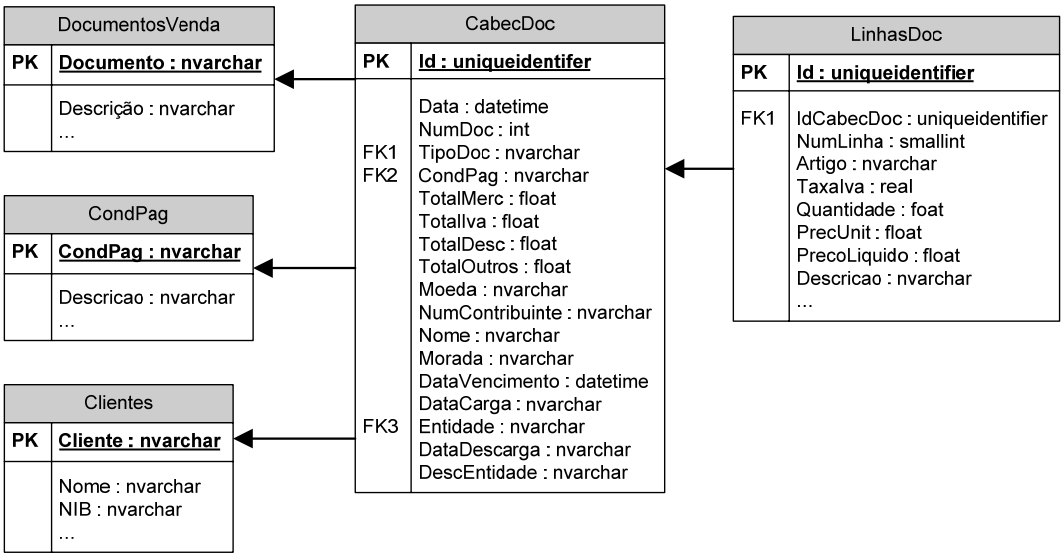

Figura 3.2 - Diagrama Relacional das vendas

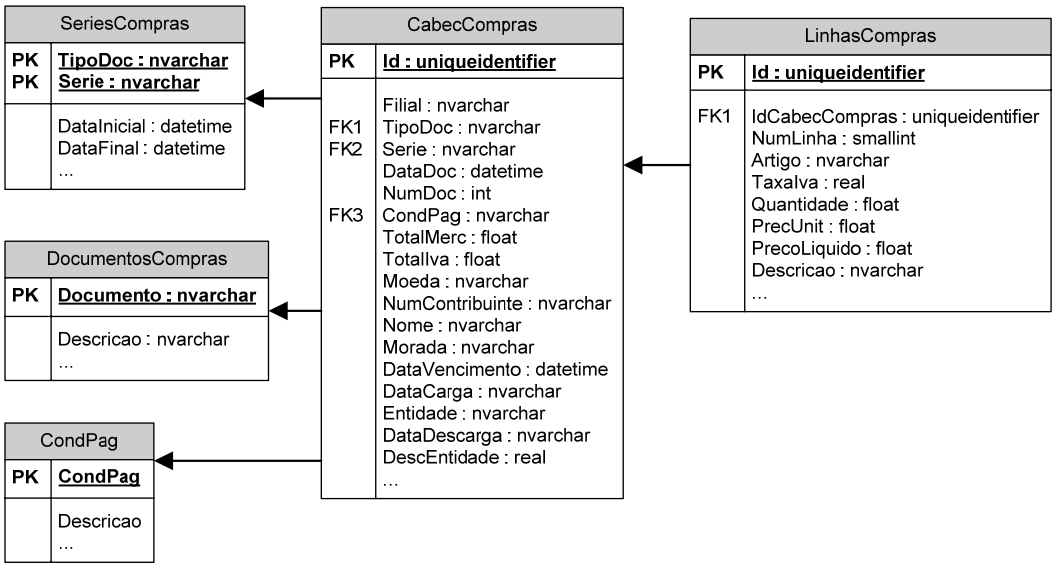

Figura 3.3 - Diagrama Relacional das compras

A tabela CabecDoc contém registo dos cabeçalhos dos documentos de venda emitidos através do ERP Primavera, contendo campos tais como a data de emissão do documento, tipo de documento, número do documento, etc. Já a tabela LinhasDoc contém registo referente às linhas que dizem respeito aos vários itens (produtos) presentes num documento de venda. A tabela LinhasDoc está relacionada com a tabela CabecDoc, na medida em que as linhas existentes na tabela LinhasDoc correspondem a um cabeçalho existente na tabela CabecDoc.

Por outro lado as tabelas CabecCompras e LinhasCompras correspondem a registos de documentos de venda, contendo informação em muito igual às tabelas CabecDoc e LinhasDoc respectivamente. Finalmente, a tabela Clientes contém registo dos clientes da empresa, com os quais a mesma tem transacções comerciais. A tabela CabecDoc está ligada à tabela Clientes, na medida em que um documento de venda está sempre relacionado a um cliente, sendo o remetente desse mesmo documento. A tabela Clientes contém toda a informação necessária para o preenchimento dos campos correspondentes ao cliente para quem se destina o documento de venda.

A Tabela 3.1 mostra a relação entre os campos presentes na base de dados, relacionados com as tabelas anteriormente referidas, e os campos de um documento transaccional, que no caso deste projecto consiste numa factura electrónica. Essas relações são especificadas no repositório de metadados (*Secção 4.3*) e posteriormente tratadas pelo adaptador, de forma a ser possível converter de um tipo de dados para outro (*Secção 5.4.3*).

Tabela 3.1 - Relações entre os campos de uma factura e a informação na Base de Dados

Informação Factura		**Informação Base de Dados do PRIMAVERA EXPRESS (Vendas – Para Envio)**		**Informação Base de Dados do PRIMAVERA EXPRESS (Compras - Recebido)**	
Atributo	**Definição**	**Tabela**	**Campo**	**Tabela**	**Campo**

Data Emissão	Data de Emissão da Factura	CabecDoc	Data	CabecCompras	DataDoc
Número Factura	Número de ordem da linha da Factura	CabecDoc	NumDoc	CabecCompras	NumDoc
Nome Fornecedor		Empresas	IDNome	CabecCompras	Nome
Morada Fornecedor		Empresas	IDMorada	CabecCompras	Morada
Número Fiscal Fornecedor		Empresas	IFNIF	CabecCompras	NumContribuinte
Nome Cliente		CabecDoc	Nome	Clientes	NIB
Morada Cliente		CabecDoc	Morada	Empresas	IDNome
Número Fiscal Cliente		CabecDoc	NumContribuinte	Empresas	IDMorada
NIB		Clientes	NIB	Empresas	IFNIF
Número Item	Número de ordem do item na factura	LinhasDoc	NumLinha	LinhasCompras	NumLinha
Referência Produto	Código identificador do bem ou serviço prestado	LinhasDoc	Artigo	LinhasCompras	Artigo
Nome do Bem ou Serviço	Denominação usual do bem transmitido ou serviço prestado	LinhasDoc	Descricao	LinhasCompras	Descricao
Quantidade		LinhasDoc	Quantidade	LinhasCompras	Quantidade
Preço Unitário		LinhasDoc	PrecUnit	LinhasCompras	PrecUnit
Taxa Imposto	IVA	LinhasDoc	TaxaIva	LinhasCompras	TaxaIva
Preço Líquido de Imposto	Preço do bem ou serviço com valor tributável	LinhasDoc	PrecoLiquido	LinhasCompras	PrecoLiquido
Unidade de Medida	Moeda utilizada na factura	CabecDoc	Moeda	CabecCompras	Moeda
Descontos	Desconto aplicado ao bem ou serviço	CabecDoc	DescEntidade	CabecCompras	DescEntidade
Documento	Venda a Dinheiro, Factura, Factura/Recibo, nº guia de remessa, nº guia de devolução (nota de crédito e de débito), guias de transporte, facturas pró-forma, orçamento	CabecDoc	TipoDoc	CabecCompras	TipoDoc
Data de Disponibilização	Data em que os bens foram colocados à disposição do adquirente	CabecDoc	DataCarga	CabecCompras	DataCarga
Valor Total IVA	Valor total do imposto sobre os itens facturados	CabecDoc	TotalIva	CabecCompras	TotalIva
Valor Total Bruto	Valor total dos itens facturados sem a tributação aplicável	CabecDoc	TotalMerc	CabecCompras	TotalMerc
Valor Total Líquido	Valor total dos itens facturados com a tributação aplicável	CabecDoc	TotalMerc + TotalIva	CabecCompras	TotalMerc + TotalIva
Pagamento	Condições Pagamento	CabecDoc	CondPag	CabecCompras	CondPag
Data de Recebimento		CabecDoc	DataDescarga	CabecCompras	DataDescarga
Data limite de pagamento		CabecDoc	DataVencimento	CabecCompras	DataVencimento

Para além das tabelas anteriormente referidas, existem outras tabelas necessárias para escrever na base de dados do Primavera Express, como se pode verificar na Figura 3.2 e na Figura 3.3:

- SeriesCompras – Tabela de registo das séries dos documentos de compra e respectivas configurações.
- DocumentosCompra – Tabela de registo dos tipos de documento de compra e respectivas configurações.

- DocumentosVendas – Tabela de registo de documentos de venda e respectivas configurações.
- CondPag – Tabela de registo das condições de pagamento.

3.5. Resumo

Este sistema é, de todos aqueles existentes no mercado português, aquele que melhor se enquadra no contexto de utilização do adaptador para mediação electrónica, pelo que ficou decidido criar um adaptador personalizado partindo das suas funcionalidades e modelo relacional. Este sistema é bastante acessível tanto a PME como a grandes empresas, contendo as funcionalidades necessárias para gerir os processos de negócio da empresa, mas sem adicionar custos. A sua obtenção é gratuita, podendo qualquer empresa o obter através da Internet na página do Primavera BSS, mais especificamente na página [39].

No capítulo seguinte, irá ser abordado o repositório de metadados ou também referido como um ficheiro de configurações dos sistemas, bastante importante para funcionamento do adaptador no âmbito deste trabalho.

4. Repositório de Metadados

4.1. Motivação

O objectivo deste adaptador é fornecer um modelo normalizado em que cada Sistema ERP implementa e configura um repositório de metadados de acordo com a sua lógica relacional para obtenção de documentos utilizados na troca de documentos. Este adaptador serve essencialmente para facilitar o acesso a sistemas legados, convertendo os seus dados para um formato canónico e proporcionar desta forma a comunicação entres vários sistemas legados. Esta conversão é efectuada por configuração em ficheiros XML, com o objectivo de não implicar programação para integrar um novo ERP.

Este adaptador surge como uma solução para os problemas gerados na troca de mensagens entre sistemas legados, e normalmente, incompatíveis entre si. A estrutura (ou formato, ou *schema*) das mensagens, assim como a sua interpretação, não é normalizada e está portanto dependente tanto da aplicação que envia, como da aplicação que recebe a mensagem. Esta necessidade resulta em custos e acaba por limitar a integração à troca de dados relativamente simples. Para resolver este problema é acordado um formato único para as mensagens trocadas, e os sistemas passam apenas a ter de converter os seus dados do formato relacional para esse formato canónico. Para além de resolver este problema, o adaptador resolve o inconveniente de necessidade de programação ou alteração do código por parte dos programadores dos sistemas para codificar as mensagens que envia, assim como para descodificar as mensagens que recebe. O adaptador executa automaticamente a codificação e descodificação das mensagens, bem como todos os processos adjacentes, retirando esse esforço ao programador. Tal é conseguido através de um repositório de metadados que contém informação relacionada com parâmetros de conexão com as bases de dados dos sistemas, bem como as operações, parâmetros e resultados esperados num processo de codificação e descodificação de dados para leitura e escrita na base de dados.

O repositório de metadados serve essencialmente para gerir “dados sobre dados”, de forma a facilitar outras formas de integração baseadas em dados, particularmente importante para este trabalho. Este repositório de metadados descreve a localização e o formato dos dados no Sistema ERP, em termos de entidades e métodos, e é escrito por um programador que tenha conhecimento do Sistema em si.

O elemento integrador entre o adaptador e o Sistema ERP é uma base de dados que corre sobre a ferramenta *SQL Server* e, consequentemente, com a utilização da linguagem de programação SQL.

Resumindo, o repositório de metadados consiste num armazenamento de dados sobre dados, num repositório especializado, que inclui:

- Formatos dos dados para as mensagens;
- Transformações entre formatos;
- Regras para processamento;
- Arquitectura das integrações com as bases de dados.

Ao contrário de uma base de dados tradicional – onde se guardam enormes quantidades de dados relativamente simples – um repositório está vocacionado para armazenar relativamente pequenas quantidades de dados muito complexos.

Numa altura em que tanto a arquitectura tecnológica da empresa como a integração entre os diversos sistemas são mais importantes para o conjunto de sistema em si, o repositório de metadados assume um papel preponderante. No limite, o repositório pode armazenar informação sobre todas as aplicações e sistemas de informação de uma empresa, transformando-se, neste caso, num autêntico repositório empresarial.

Essencialmente, o adaptador fornece um método de execução normalizada capaz de ler a informação contida no repositório de metadados e desta forma conseguir obter dados e introduzir dados no Sistema ERP de uma forma normalizada. Esta informação é convertida para um formato de dados também normalizada, de acordo com um formato designado pelos Sistemas intervenientes, e utilizada para construção de mensagens trocadas entre estes Sistemas.

4.2. Princípios Gerais

O Business Data Catalog ou BDC [40] é uma ferramenta recente de integração do *Microsoft Office SharePoint Server 2007* [42]. É um serviço compartilhável que permite ao *Office SharePoint Server 2007* navegar pelos dados do negócio através de aplicações servidores *back-end* sem qualquer implementação.

Como é referido em [43], um dos objectivos principais de desenvolvimento do BDC é permitir navegar pelos dados do negócio de várias aplicações *Line-of-Business* (LOB) tais como o *SAP*, *Siebel*, e bases de dados (*SQL Server* ou *Oracle)* no *Office SharePoint Server 2007* com o mínimo de esforço de implementação. De forma a conseguir alcançar este objectivo, o BDC fornece acesso aos dados subjacentes através de um modelo de metadados que fornece um modelo consistente e simplificado do objecto cliente, que descreve a localização e o formato dos dados dentro do sistema LOB em termos de entidades e métodos. O BDC fornece também ferramentas para ler estes metadados e obter os dados do sistema LOB, que são retornado num formato normalizado. O acesso aos dados é feito apenas num sentido, isto é, permite efectuar apenas leituras ao sistema LOB.

Embora o adaptador deste trabalho, assim como o BDC, utilize um modelo de metadados para acesso aos dados na base de dados, o modelo de metadados utilizado no âmbito deste trabalho serve também para efectuar as conversões do formato relacional para o formato canónico, e vice-versa. Outro aspecto de diferenciação em comparação como BDC e que, para além de efectuar leituras à base de dados, também é permitido efectuar escritas na base de dados. De referir que, a Ao contrário do BDC, o adaptador implementado neste trabalho apenas fornece meios para integrar os dados com o sistema *SQL Server*.

Quer a conexão entre o BDC e a base de dados do sistema, e o adaptador e a base de dados do sistema é obtida utilizando ferramentas ADO.NET. Para tal, os metadados necessitam que sejam

definidos os métodos para executar as operações SQL. Tipicamente, os autores dos metadados com habilidade equivalente aos programadores das bases de dados, conseguem descrever a API das aplicações de negócio utilizando o modelo dos metadados. [41]

Como referido em [44], o primeiro passo para utilização do BDC é escrever um ficheiro XML contendo os metadados para conexão com o sistema *back-end*. Depois de escrito os metadados para o BDC, define-se os dados que se pretende obter em termos de entidades. O formato dos metadados do BDC permite definir associações entre entidades em cenários em que existe uma relação de um-para-muitos, tal como pode acontecer entre clientes e facturas.

A definição de uma entidade contém identificadores, propriedades e métodos. Os métodos definem como o BDC interage com pontos de entrada expostos pelo sistema *back-end*. Para um sistema *back-end* que é uma base de dados tal como o *SQL Server* ou *Oracle*, os métodos definem os nomes das operações SQL.

Uma vez construído o ficheiro XML com os metadados requeridos pelo BDC para um sistema *back-end*, é necessário importá-lo para o BDC dentro do espaço de um SSP particular para criar o que é conhecido como uma aplicação BDC. Este processo é realizado importando os processos utilizando as *Web pages* administrativas do SSP. Pode-se alternativamente importar o ficheiro XML com os metadados do BDC utilizando código personalizado ao encontro do modelo administrativo do objecto BDC.

Após importado os metadados requeridos para criar uma aplicação BDC, existem diversas técnicas para gerir e mostrar os seus dados dentro de um portal. Como referido em [42], o *Office SharePoint Server 2007* interliga com um conjunto de peças *Web* do negócio que podem rapidamente ser adicionadas às páginas para efectuar perguntas e posteriormente mostrar dados do BDC. Pode-se também adicionar novas colunas às listas e às bibliotecas originais baseadas numa entidade definida numa aplicação BDC. Um utilizador que edite uma coluna baseada numa entidade BDC é automaticamente apresentada automaticamente com uma interface utilizador tornando possível perguntar o sistema *back-end*. O BDC foi projectado para integrar com o *Office SharePoint Server 2007 Search Service*. Por exemplo, um sistema *back-end* e as suas entidades podem ser definidos como uma fonte segura de modo que o serviço de indexação do *Office SharePoint Server 2007* percorra todos os seus dados e construa índices para o motor de procura do *Offce SharePoint Server 2007*. Este transformou-se numa característica poderosa pois permite aos utilizadores descobrir dados do sistema *back-end* sobre coisas como clientes e transacções comerciais quando colocadas *queries* padrão de procura nos portais de procura do *Office SharePoint Server 2007* assim como nos sites padrão do *Windows SharePoint Services 3.0*.

O adaptador ao contrário do BDC não foi projectado nem implementado para ser utilizado por um portal. O adaptador foi projectado para integrar os dados num cenário de mediação electrónica, sem qualquer interface para os utilizadores para além da interface do Sistema ERP.

Finalmente, as entidades BDC podem ser alcançadas utilizando código personalizado escrito de acordo com o modelo do objecto BDC. Isto torna possível escrever *Web Parts* personalizados assim

como outros componentes do lado do utilizador e serviços que executem as suas próprias *queries* BDC. Um aspecto agradável de escrever código para efectuar perguntas às entidades BDC é que não é necessário nos preocuparmos com a gestão das conexões ou se o acesso ao sistema *back-end* é feito através de *Web Services* ou através de *ADO.NET*. Todos esses detalhes são abstraídos pelos metadados do BDC e pelo motor de execução do BDC. [41][42]

4.3. Funcionalidades

Os metadados têm duas finalidades: descrever a API do sistema, e tornar a API do sistema compreensível de forma a torná-la fácil de utilizar. Com a utilização do modelo de metadados, os sistemas passam a poder integrar os dados de múltiplas fontes utilizando as configurações introduzidas no repositório de metadados, e converte-los para o formato desejado.
A Figura 4.1 mostra os componentes de configuração existentes no modelo de metadados.

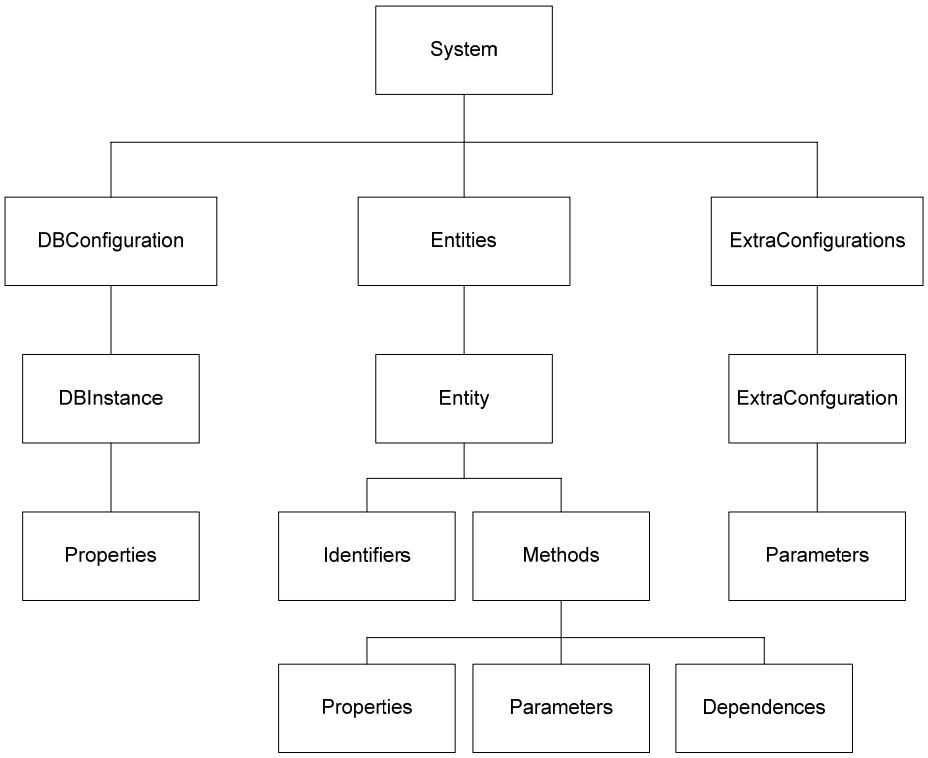

Figura 4.1 - Esquema do repositório de metadados

A componente *DBConfiguration* corresponde aos parâmetros necessários para comunicação com a base de dados, que é utilizado pelo adaptador para obtenção ou escrita de informação na mesma. Esta informação é parametrizada no nó *DBConfiguration*, adicionando o seguinte excerto de XML (Figura 4.2) as vezes que seja necessário para adicionar as configurações de todas as bases de

dados que irão ser utilizadas. Os parâmetros correspondem ao nome da base de dados, o servidor da base de dados e finalmente a autenticação e a senha de acesso à base de dados.

```
<DBInstance Name="nome_indicativo_da_basedados">
   <Properties>
      <Property Name="DataSource" Type="System.String">server_name</Property>
      <Property Name="Database" Type="System.String">database_name</Property>
      <Property Name="Login">
         <PropertyInstance Name="UID" Type="System.String">username</PropertyInstance>
         <PropertyInstance Name="PWD" Type="System.String">password</PropertyInstance>
      </Property>
   </Properties>
</DBInstance>
```

Figura 4.2 - Nó *DBConfiguration* para configuração de conexões com a base de dados

A componente *Entity* corresponde ao conjunto de configurações necessárias de forma a obter ou introduzir informação no sistema, que foi anteriormente configurado na componente *DBConfiguration*. Cada *Entity* contém pelo menos um *Method*, que nos permite introduzir um leque de operações possíveis no sistema. Uma entidade consiste num título (definido no elemento das propriedades), num identificador único (que iguala a uma chave primária), e em métodos que definem tanto os atributos de uma entidade assim como indica ao adaptador como obter ou introduzir os dados da base de dados. O seguinte excerto XML (Figura 4.3) mostra como definir um elemento *Entity*.

```
<Entity Name="entity_name">
   <Identifiers>
      <Identifier Name="identifier_name" Type="identifier_type"></Identifier>
   </Identifiers>
   <Methods>
      <Method Name="method_name" Source="database_name">
         <Properties>
            <Property Name="FormatString" Type="System.String">query_to_execute</Property>
         </Properties>
         <Parameters>
            <Parameter Direction="In" Name="parameter_name">
               <TypeDescriptor TypeName="parameter_type"
                  IdentifierName="parameter_identifier_name">
               </TypeDescriptor>
            </Parameter>
            <Parameter Direction="Return" Name="object_name_to_store_information">
               <TypeDescriptor TypeName="System.Data.IDataReader"
               IsCollection="ArrayList_or_single_value">
                  <TypeDescriptors>
                     <TypeDescriptor TypeName="object_attribute_type"
                        Name="object_attribute_name" TypeDBName="db_attribute_type"
                        DBName="db_attribute_name"></TypeDescriptor>
                  </TypeDescriptors>
               </TypeDescriptor>
            </Parameter>
         </Parameters>
         <Dependences>
            <Dependence Name="dependence_method_name" Query="query_to_execute">
               <DependenceProperties>
                  <DependenceProperty Name="MainAttribute" Type="main_attribute_type"
                     Identifier="main_attribute_identifier"></DependenceProperty>
                  <DependenceProperty Name="DependenceIdentifierAttribute"
                     Type="dependence_attribute_type"
                     Identifier="dependence_attribute_identifier"></DependenceProperty>
```

```
                <DependenceProperty Name="ResultAttribute" Type="result_attribute_type"
                    Identifier="result_attribute_identifier"></DependenceProperty>
            </DependenceProperties>
        </Dependence>
    </Dependences>
  </Method>
 </Methods>
</Entity>
```

Figura 4.3 - Nó *Entity* para configuração de operações sobre a base de dados e conversão de dados

Como se pode observar pelo excerto de XML, um método consiste num nome indicativo da operação que irá ser realizada e na indicação da base de dados que irá ser utilizada para tal operação. Um método contém um elemento *Properties* que define a *query* utilizada para acesso aos dados da base de dados, um elemento *Parameters* que define os parâmetros de entrada bem como os dados retornados pela *query*, e finalmente poderá conter o elemento *Dependences* que define relações com outras tabelas, através de atributos que são chaves estrangeiras para essas tabelas.

O elemento *Parameters* pode conter parâmetros de entrada e/ou parâmetros de saída. Numa situação de introdução na base de dados basta indicar os atributos que contém a informação a ser guardada, não sendo necessário indicar o resultado esperado. Na situação contrária, em que a informação é obtida através da base de dados, já pode ser necessária indicar os atributos utilizados como parâmetros de entrada bem como os atributos de retorno. Um parâmetro de entrada consiste na indicação do tipo de dados correspondente ao tipo de dados do atributo na base de dados e num identificador que corresponde ao nome do objecto que contém a informação a ser utilizada. Um parâmetro de retorno consiste na indicação do tipo de dados do atributo a ser guardado, bem como o nome em que irá ser guardado, no tipo de dados do atributo que é lido da base de dados e o nome do mesmo atributo na base de dados. Esta configuração para os parâmetros de retorno é utilizada para conversão do tipo de dados utilizado na base para o tipo de dados na forma canónica. Isto é, ao detectar que o tipo de dados utilizado na base de dados difere do tipo de dados em que é pretendido guardar efectua-se a conversão.

No contexto da conversão dos dados, existe um sub-elemento para cada um dos parâmetros de retorno, *Conversion*, que consiste em conversões como alterar o tipo de dados utilizado na base de dados para um tipo de dados que se pretende, mas sim conversões intermédias para se poder converter o atributo para o tipo de dados pretendido. A Figura 4.4 mostra como definir um elemento deste tipo.

```
<TypeDescriptor TypeName="object_attribute_type" Name="object_attribute_name"
   TypeDBName="db_attribute_type" DBName="db_attribute_name"></TypeDescriptor>
   <Conversion DBType=" db_attribute_type " ConversionType="conversion_type"></Conversion>
</TypeDescriptor>
```

Figura 4.4 - Nó *Convertion* para conversão de dados do formato relacional para um dado formato

Um exemplo poderia ser o seguinte: o atributo é uma data, em que tanto o tipo de dados na base de dados como o tipo de dados em que o atributo irá ser enviado é igual, formato alfanumérico. Porém, o formato alfanumérico utilizada pela base de dados não é completamente representativa da data, isto é, não contém todos os valores de uma data, e ao efectuar a conversão de *String* para *DateTime* tal problema é resolvido. Temporariamente o tipo de dados na base de dados passa para o tipo de dados para que foi convertido, e é verificado se esse mesmo tipo é igual ao tipo de dados em que é pretendido para a construção do documento a enviar. Isto torna-se bastante útil para integração com outros sistemas que utilizam tipos de dados diferentes.

O elemento *Dependences*, como anteriormente referido, define o atributo que é chave estrangeira para outra tabela, e desta forma, obter a informação pretendida na tabela relacionada de acordo com essa mesma chave estrangeira e um parâmetro de entrada. Para cada atributo em que se pretende obter informação da tabela relacionada é necessário indicar qual o nome identificador em que se pretende guardar essa mesma informação (*MainAttribute*), indicar o nome identificador do atributo que é chave estrangeira para utilização do seu valor para a *query* de procura (*DependenceIdentifierAttribute*), e finalmente indicar o nome identificador do atributo que contém a informação que se pretende (*ResultAttribute*) e que irá ser guardada no atributo indicado no *MainAttribute*.

Finalmente, e apenas se necessário, poderá ser adicionado ao ficheiro o elemento *ExtraConfigurations*, após as entidades, que define configurações extra para obtenção de informação necessária para o funcionamento do adaptador, mas que não irá ser utilizada para a construçao do documento a ser enviado ou que é recebido. Isto é, são configurações auxiliares. Um *ExtraConfiguration* consiste num nome indicativo da operação de configuração que irá ser realizada, bem como por uma *query* de procura ou execução, como se pode verificar na Figura 4.5. Este mesmo elemento contém um elemento *Parameters* que define os parametros de entrada e/ou de retorno de acordo com a operação que é pretendida, em tudo igual ao utilizado nos métodos. Um aspecto que difere do *Parameters* utilizado nos métodos é o tipo de retorno, isto é, neste caso é possível retornar apenas um ordinal ou um objecto, como acontece nos métodos. No caso de retornar um ordinal, os campos *TypeDBName* e *DBName* não são preenchidos.

```
<ExtraConfigurations>
   <ExtraConfiguration Name="operation_name" Query="query_to_execute">
      <Parameters>
         <Parameter Direction="In/Return" Name="object_or_ordinal_name">
            <TypeDescriptor TypeName="Object_or_Ordinal"
            IsCollection="result_is_collection?">
               <TypeDescriptors>
                  <TypeDescriptor TypeName="object_attribute_type"
                     Name="object_attribute_name" TypeDBName="db_attribute_type"
                     DBName="db_attribute_name"></TypeDescriptor>
               </TypeDescriptors>
            </TypeDescriptor>
         </Parameter>
      </Parameters>
   </ExtraConfiguration>
</ExtraConfigurations>
```

Figura 4.5 - Nó *ExtraConfiguration* para configurações extra

Neste trabalho é utilizado para saber especificamente como se obtém os atributos necessários para detectar documentos introduzidos recentemente no sistema.

4.4. Resumo

Neste capítulo foi explicada a estrutura e base de funcionamento do ficheiro de configuração do adaptador, ou também referenciado como repositório de metadados, ficheiro esse fundamental para o funcionamento deste trabalho.

Foi abordada a motivação que levou à implementação de um repositório de metadados para configuração de aspectos de conexão e operações de leitura e escrita na base de dados, focando os princípios gerais em que se baseia e aspectos de funcionalidade.

Este repositório de metadados é a base de todo o funcionamento do adaptador implementado, indicando o modelo relacional do Sistema ERP onde se encontra integrado, e também o formato do documento electrónico a ser trocado com os seus parceiros financeiros. Desta forma torna-se possível a comunicação sem problemas de integração entre empresas com sistemas diferentes, muitas das vezes legados.

No capítulo seguinte irá ser descrita a arquitectura e organização do adaptado, aspectos de implementação e respectivas soluções, tanto de implementação própria como em relação às tecnologias de integração utilizadas para a implementação deste adaptador.

5. Solução Tecnológica de Integração

Neste capítulo pretende-se descrever as principais opções arquitecturais tomadas para a implementação do adaptador para mediação electrónica. Para tal apresenta-se a arquitectura do *Invoice Adapter* implementado, descrevendo os módulos que o constituem e como se relacionam. Será também explicado a lógica de desenvolvimento e funcionamento, e como este adaptador torna possível a integração com outros sistemas.

Inicialmente descreve-se um possível cenário de utilização deste adaptador, que visa introduzir alguns conceitos essenciais para uma melhor compreensão do tema exposto, e que é ilustrado na Figura 5.1. Este cenário é composto por um *Message Broker* que estabelece a ligação entre vários Sistemas ERP. O *Message Broker* é responsável pelo suporte para modelação de processos de negócio e pela comunicação e encaminhamento seguro de documentos trocados entre dois ou mais Sistemas ERP. Por outro lado, o *Invoice Adapter* é uma componente integrada em cada Sistema ERP, onde são tratados os aspectos de integração e conversão dos dados para possibilitar a troca de documentos entre os Sistemas ERP na rede, geralmente sistemas legados. Desta forma, em vez de cada sistema estar ligado ponto-a-ponto a todos os outros sistemas, necessita apenas de estar ligado a um interlocutor único de comunicação, o *Message Broker*.

Este trabalho não inclui o desenvolvimento do *Message Broker*, mas antes o estudo e desenvolvimento de um adaptador que é integrado com os Sistemas ERP. Como é possível observar na Figura 5.1, cada Sistema ERP contém uma aplicação extra, o *Invoice Adapter*, que é parametrizada e configurada de acordo com o sistema a que está associada. Outro aspecto importante é a utilização da framework WCF entre o *Message Broker* e o Sistema ERP.

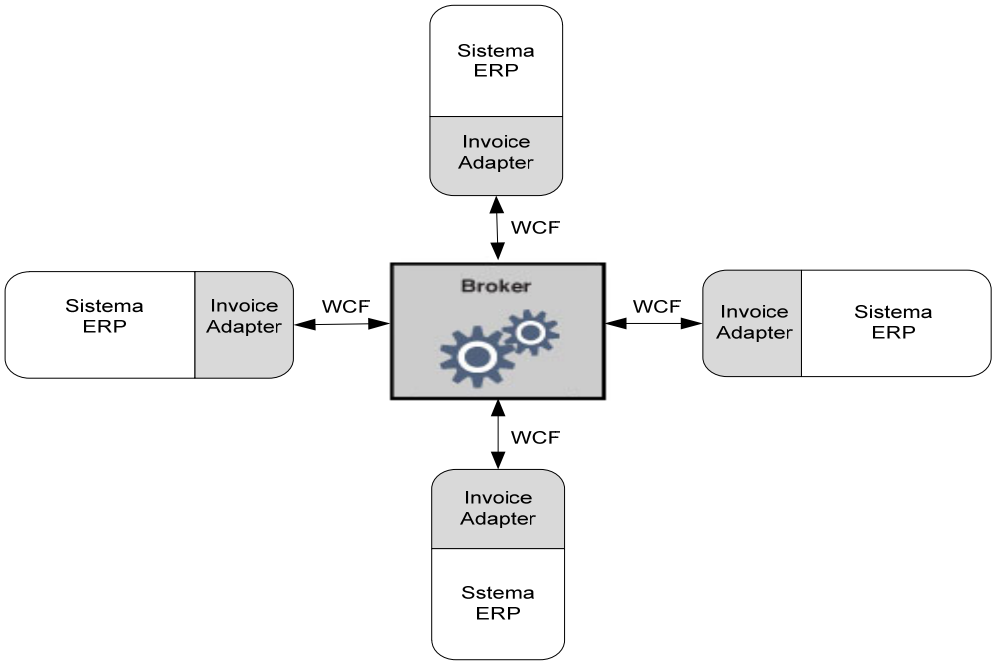

Figura 5.1 - Cenário de utilização do Invoice Adapter

O WCF foi escolhido pois fornece uma *framework* de programação unificada para a construção rápida de aplicações seguras, fiáveis e orientadas a serviços, com suporte a *standards* de comunicação como sejam *Ws-Security*, *Ws-Trust*, *Ws-Addressing*, entre outros. Na *Secção 0* é explicado o funcionamento do WCF neste projecto.

Devido à variedade e ao elevado número de Sistemas ERP que existem no mercado, tornou-se necessário procurar conceber e desenvolver um adaptador que resolvesse o problema de incompatibilidades entre sistemas que pretendam comunicar e trocar informação. Este adaptador surge como uma tentativa de solucionar este problema.

Um Sistema ERP (S1) que pretenda enviar uma factura para um outro Sistema ERP (S2) pode deparar-se com o problema de os seus dados não serem "compreendidos", dificultando ou impossibilitando o processamento dos mesmos pelo Sistema S2. O que para o Sistema S1 é um valor lógico e correcto, para o Sistema S2 pode não significar nada ou não fazer sentido. Para solucionar problemas como este, passa a existir um adaptador em cada sistema interveniente numa comunicação entre o sistema e o *Message Broker* de comunicação. Estes adaptadores são preparados para conseguir obter ou inserir informação no Sistema ERP em que se encontram, e de transformar os dados utilizados pelo sistema para uma forma canónica (situação de envio de um documento) ou a operação inversa de transformação de uma forma canónica para o tipo de dados utilizado pelo sistema (situação de recepção de um documento).

Cada adaptador tem apenas conhecimento do Sistema ERP a que está integrado, mas nenhum conhecimento sobre os outros Sistemas ERP presentes na rede. Para o funcionamento de um adaptador apenas é necessário saber lidar com o Sistema ERP em que se encontra e efectuar a transformação dos dados de e para o formato canónico acordada entre os Sistema ERP intervenientes na rede.

Outro aspecto muito importante é a capacidade deste adaptador de, para além de possibilitar a integração de um Sistema ERP com outros Sistemas ERP na troca de mensagens, poder ser utilizado com vários tipos de Sistemas ERP. Isto é, com o repositório de metadados e o comportamento do adaptador a depender da informação existente no repositório de metadados, torna-se o adaptador genérico e, desta forma, possível de ser utilizado sem necessidade de reescrever código específico aos vários Sistemas ERP existentes no mercado.

5.1. Visão Global do Invoice Adapter

O *Invoice Adapter* é constituído por três componentes fundamentais, a **Camada de Comunicação**, a **Camada de Construção ou Leitura de Mensagens** e finalmente a **Camada de Comunicação com Base de Dados e Conversão de Dados**. Na Figura 5.2 é possível observar o modo como estas camadas se encontram estruturadas, bem como as ligações entre si.

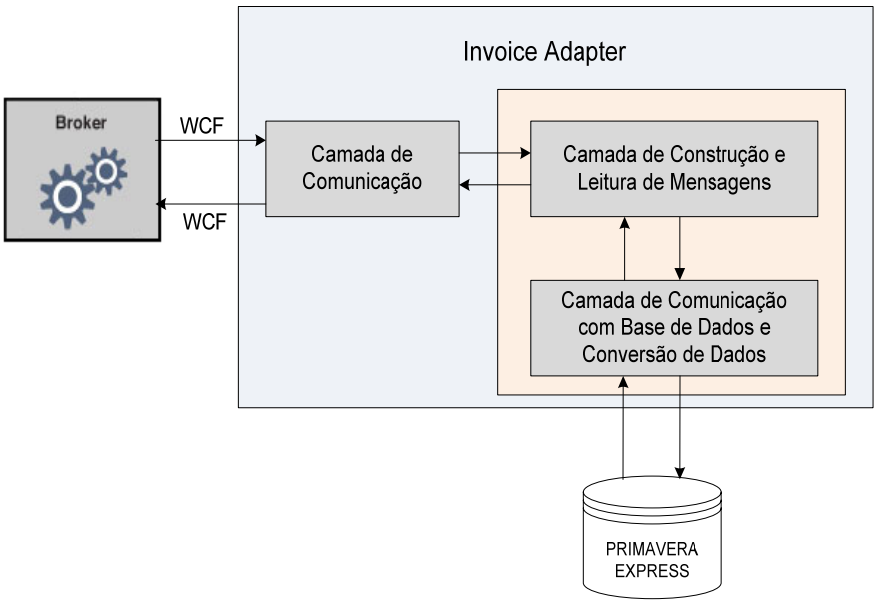

Figura 5.2 - Visão Global do Invoice Adapter

A **Camada de Comunicação** consiste na componente responsável pela comunicação com o *Broker* e por fazer a gestão das mensagens recebidas e das mensagens a enviar para o *Broker*. A *Secção 5.2* apresenta a sua descrição, focando a *framework* WCF e a gestão de mensagens com a tecnologia MSMQ para guardar em filas de espera as mensagens recebidas e por enviar.

A **Camada de Construção e Leitura de Mensagens** consiste na componente responsável pelo tratamento das mensagens que passam pelo adaptador, bem como pela gestão de acontecimentos na recepção e envio de mensagens. É também responsável pela leitura do repositório de metadados, envio da informação para a Camada de Conversão de Dados e Comunicação com a Base de Dados, sendo a sua descrição apresentada na *Secção 5.3*.

Finalmente, a **Camada de Comunicação com Base de Dados e Conversão de Dados** consiste na componente responsável pelo tratamento e conversão dos dados, tanto para o formato canónico como para o formato relacional da base de dados do Sistema ERP. É também responsável pela leitura dos dados da base de dados para enviar, bem como pela introdução de novos dados, recebidos de outro Sistema ERP, na base de dados. A descrição deste módulo é apresentada na *Secção 5.4*.

Como se pode verificar na Figura 5.3, uma mensagem recebida na Camada de Comunicação é guardada numa fila de espera, e posteriormente enviada para a Camada de Construção e Leitura de Mensagens. Na Camada de Construção e Leitura de Mensagens os campos da mensagem são lidos e associados a um objecto genérico guardado em memória. Na fase seguinte, já na Camada de Comunicação com Base de Dados e Conversão de Dados, a informação contida no objecto genérico é convertida para o tipo de dados utilizado pela base de dados do Sistema ERP e escrita na mesma.

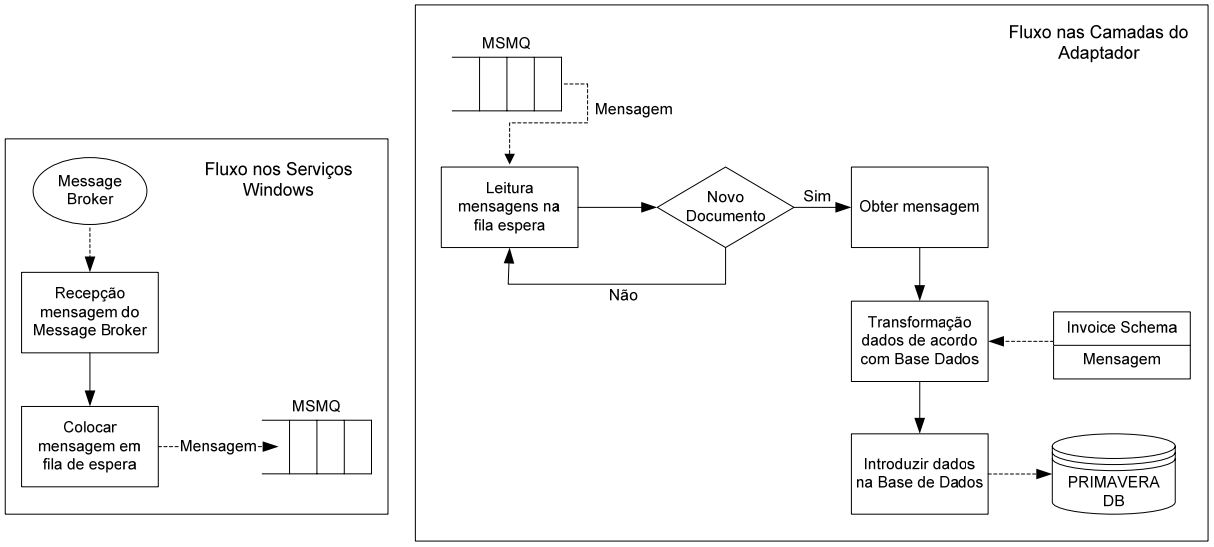

Figura 5.3 - Fluxo de execução na chegada de uma Mensagem ao *Invoice Adapter*

Como se pode verificar na Figura 5.4, na situação inversa, uma nova mensagem detectada pelo adaptador na base de dados do Sistema ERP gera um evento na Camada de Construção e Leitura de Mensagens. Esta camada comunica com a Camada de Comunicação com Base de Dados e Conversão de Dados para efectuar a leitura dos dados da base de dados (de acordo com um identificador recebido da Camada de Construção e Leitura de Mensagens) e conversão dos dados para uma forma canónica, enviando de volta estes últimos dados sobre a forma de um objecto genérico. A Camada de Construção e Leitura de Mensagens procede à construção da mensagem para ser enviada e envia essa mensagem para a Camada de Comunicação. Finalmente a mensagem é enviada pela Camada de Comunicação usando WCF.

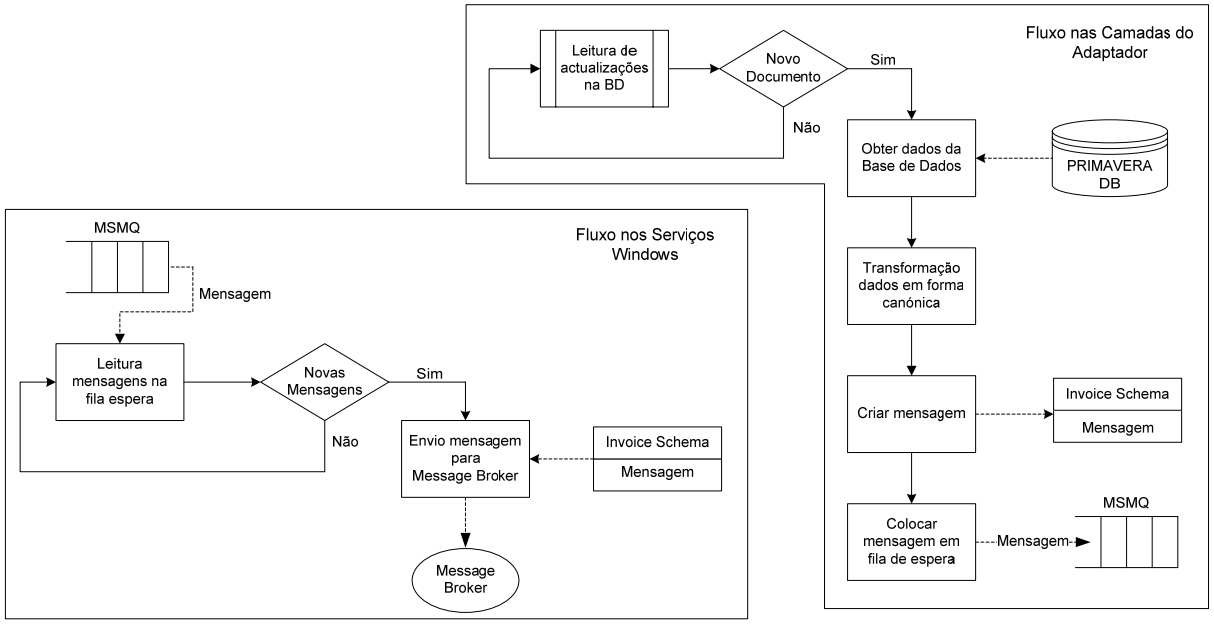

Figura 5.4 - Fluxo de execução de envio de uma Mensagem no Invoice Adapter

5.2. Camada de Comunicação

Este módulo surge como uma camada separada dos restantes módulos. Como se pode verificar na Figura 5.5, este módulo é constituída por quatro Serviços Windows, que estão sempre em execução para efectuar a gestão relacionada com comunicação. Em nomenclatura de Padrões de Desenho, cada um destes serviços implementa um *Observer*. Na *Secção 5.2.2* é apresentado o funcionamento dos Serviços Windows em mais pormenor.

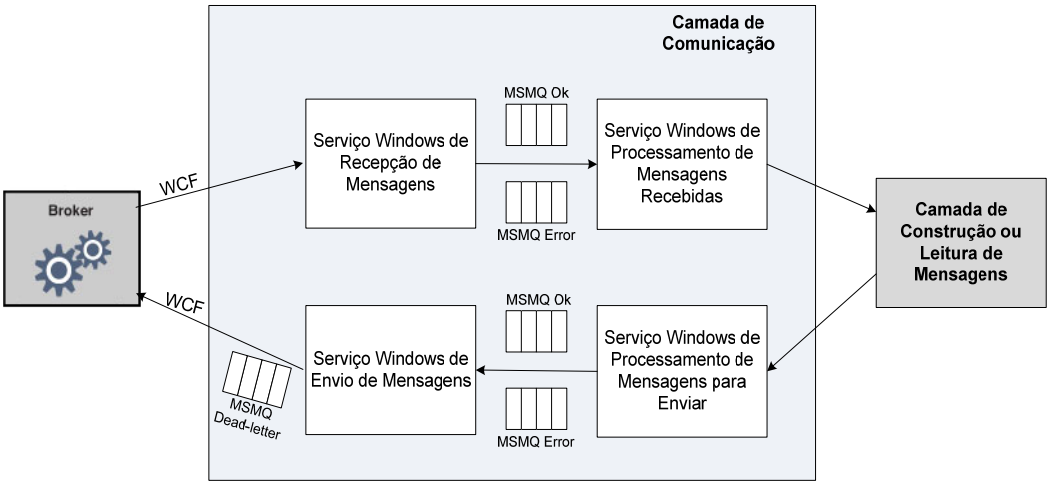

Figura 5.5 – Arquitectura da Camada de Comunicação

O Serviço Windows de Recepção de Mensagens é o componente responsável pela recepção de mensagens vindas do *Message Broker* através via WCF. Este serviço implementa um Servidor WCF, estando sempre à espera de novas mensagens para introdução no Sistema ERP. Por outro lado o Serviço Windows de Envio de Mensagens é o componente responsável pelo envio de mensagens para o *Message Broker* também através usando WCF. Este Serviço invoca os serviços disponibilizados pelo Servidor WCF do lado do *Message Broker* para enviar novas mensagens, assim que tratadas e transformadas pelas outras Camadas do *Invoice Adapter*. Em ambos os Serviços são utilizadas filas de espera (MSMQ) para colocar as mensagens recebidas ou para obter as mensagens a serem enviadas através do WCF. Na *Secção 5.2.3* é explicado em pormenor o funcionamento destas filas de espera de mensagens utilizadas nesta Camada.

O Serviço Windows de Processamento de Mensagens Recebidas é o componente responsável pela obtenção das mensagens colocadas na fila de espera pelo Serviço Windows de Recepção de Mensagens e gerar eventos na Camada de Construção e Leitura de Mensagens responsáveis pelo tratamento das mensagens recebidas. Isto é, são gerados eventos neste Serviço que desencadeiam operações na Camada de Construção e Leitura de Mensagens de leitura do conteúdo da mensagem recebida e carregamento do mesmo num objecto genérico (*Invoice*) guardado em memória, utilizado posteriormente pela Camada de Comunicação com a Base de Dados e Conversão de Dados para

conversão dos dados do formato canónico para o formato relacional e posterior escrita na base de dados.

O Serviço Windows de Processamento de Mensagens para Enviar é o componente responsável pela detecção de novos documentos na base de dados do Sistema ERP, documentos para serem enviados para o *Message Broker*. Numa primeira execução, este Serviço necessita de aceder à base de dados do Sistema ERP de forma a obter informação sobre os últimos documentos emitidos. Esse acesso é feito invocando um método na Camada de Construção e Leitura de Mensagens que trata de todas as operações necessárias para obter os últimos documentos emitidos. Ao receber a lista dos documentos a serem enviados, os mesmos são colocados na fila de espera de mensagens, lida posperiormente pelo Serviço de Envio de Mensagens e logo enviadas para o *Message Broker*.

5.2.1. Windows Communication Foundation

5.2.1.1. Visão Geral

O Windows Communication Foundation ou WCF [46] é a componente da plataforma .NET 3.0 vocacionada para a realização de sistemas distribuídos. O WCF fornece uma framework única para a construção rápida de aplicações seguras, fiáveis e orientadas a serviços. Contudo é suficientemente flexível para acomodar outros paradigmas. Agrega também, num modelo unificado, um conjunto de funcionalidades e paradigmas que anteriormente estavam dispersos por diversas tecnologias, tais como: *ASP.NET Web Services, Enterprises Services, .NET Remoting* e *MSMQ*.

Como referido em [48] [50], a segurança é uma propriedade essencial em todos os sistemas distribuídos, sendo um dos aspectos mais focados pelas especificações WS-*. O Windows Communication Foundation (WCF) reflecte esta importância, suportando um subconjunto significativo das especificações WS-*, tais como os protocolos *WS-Addressing*, *WS-Reliable* e *WS-Security*, e fornecendo várias opções e pontos de extensão para as diversas facetas da segurança. Contudo, esta diversidade e flexibilidade resultam também numa aparente maior complexidade de utilização do WCF. O WCF utiliza mensagens SOAP para a comunicação entre dois processos, possibilitando a interoperabilidade das aplicações baseadas em WCF com qualquer outro processo que comunique via mensagens SOAP. Quando se dá o caso de comunicação com processos que não utilize WCF, é utilizado codificação XML, já no caso de comunicação com processos que utilizem WCF as mensagens SOAP são codificadas num formato binário optimizado.

Como referido em [45], o WCF é composto por três partes: *Service Class*, *Host* e *Endpoint*. O *Service Class* como o próprio nome indica, é a classe que expõem os seus métodos no serviço. O *Host* é onde o WCF é hospedado, podendo ser num IIS, num *Windows Service, WAS* ou outro processo.

Como pode ser visto na Figura 5.6, um serviço WCF disponibiliza um conjunto de *Endpoints*. Cada *Endpoint* é um portal de comunicação com o “mundo”. O cliente é um programa que troca mensagens com um ou muitos *Endpoints*. Um cliente pode também disponibilizar um *Endpoint* para receber mensagens de um serviço num padrão bidireccional de troca de mensagens.

Um serviço *Endpoint* contém um *Address*, um *Binding* e um *Contract*. O *Endpoint Address* é um endereço de rede onde o *Endpoint* reside. O *Endpoint Binding* especifica como o *Endpoint* comunica com o "mundo" incluindo aspectos como o protocolo de transporte (ex. TCP, HTTP), codificação (ex. texto, binário) e segurança (ex. SSL, segurança de mensagem SOAP). O *Endpoint Contract* especifica o que é comunicado pelo *Endpoint* e é essencialmente um conjunto de mensagens organizadas em operações que contêm padrões de troca de mensagens básicos (MEPs – *Message Exchange Patterns*) tais como unidireccional, bidireccional, ou pedido/resposta.

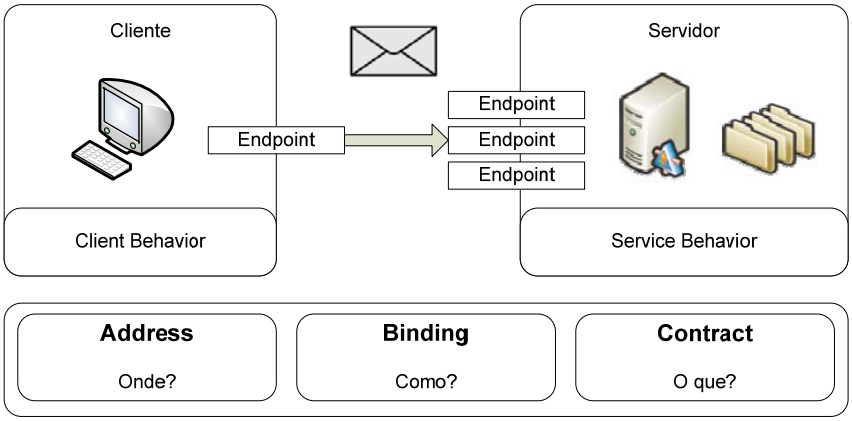

Figura 5.6 – Componentes do WCF [28]

5.2.1.2. Implementação

Como referido em [49], no WCF o serviço descreve as operações que executa num contrato, que expõe essas operações publicamente como metadados. O cliente contém uma definição do contrato do serviço e uma classe *proxy* para aceder ao serviço. O código do *proxy* é gerado através dos metadados do serviço utilizando a ferramenta *Service Model Metadata Utility Tool* (*Svcutil.exe*).
O serviço é hospedado através do Serviço Window *WindowsServiceReceivingMessages*, responsável pela recepção de mensagens do *Message Broker*. Ao criar um Serviço Windows (explicado com mais pormenor posteriormente na *Secção 5.2.2*), são gerados dois arquivos no projecto: o arquivo *service1.cs*, que contém a implementação do serviço, e o arquivo *program.cs*, que instancia e *hosts* o Serviço Windows. Para hospedar o serviço WCF no Serviço Windows, é necessário apenas implementar o código necessário nos métodos *OnStart()* e *OnStop() do Serviço Windows*, como se pode verificar na Figura 5.7. É necessário proceder à configuração do tipo de conta para *Network Service* do Serviço Windows no processo de criação do instalador para funcionamento do serviço WCF. Como o paradigma de início de um Serviço Windows é semelhante ao início dos seus serviços dentro do *ServiceHost* do WCF, a duração do serviço WCF fica vinculada à duração do Serviço Windows. [47]

```
ServiceHost host;

protected override void OnStart(string[] args){
   Type serviceType = typeof(TradeService);
   host = new ServiceHost(serviceType);
   host.Open();
}

protected override void OnStop(){
   if(host != null)
      host.Close();
}
```

Figura 5.7 - Código para proceder ao *host do serviço WCF no Serviço Windows*

A componente de segurança é assegurada através da assinatura e cifra da mensagem trocada entre o Sistema ERP e o *Message Broker*. Para tal é utilizado o protocolo *WS-Security* do *WS-**.

Como explicado em [50], o *WS-Security* é um conjunto de especificações que fornece os principais serviços de segurança tais como a integridade da mensagem, a confidencialidade da mensagem e a sua autenticação. No cenário deste trabalho, existe confiança entre o servidor e o cliente, onde o serviço autentica-se perante o cliente, e a interacção entre os dois é cifrada e assinada. Adicionalmente, toda a segurança é implementada utilizando uma aproximação de segurança baseada na mensagem. A segurança da comunicação entre os intervenientes é assegurada através da utilização de certificados digitais, que garantem a confidencialidade e a autenticidade das mensagens trocadas.

É utilizado o *binding WSHttpBinding*, que representa um *binding* que pode utilizar HTTP ou HTTPs para transporte, suportando transacções distribuídas, confidencialidade e segurança nas mensagens. Este *binding* é utilizado para comunicação através de *Web Services*, possuindo, como referido anteriormente, segurança a nível da mensagem

São utilizados pares de chaves privadas/públicas para a comunicação. A chave privada do Sistema ERP é guardado num ficheiro encriptado, podendo só o mesmo lhe aceder. Tanto este ficheiro como a chave pública do *Message Broker* são enviados pelo *Message Broker*, fora do cenário de mediação electrónica. Estas chaves são armazenadas pelo adaptador no repositório de certificados digitais no *Windows*. A chave privada é armazenada na pasta *Personal* e a chave pública na pasta *Trusted People* do *Current User*.

O Serviço Windows responsável pelo envio de mensagens (Cliente), utiliza a sua chave privada para assinar a mensagem, para identificação no *Message Broker* utilizando a sua chave pública. A chave pública do *Message Broker* é utilizada para cifrar a mensagem, garantindo que apenas o mesmo consegue decifrar a mensagem com a sua chave privada. Na situação inversa, o Serviço Windows responsável por receber mensagens (Servidor) utiliza a sua chave privada para identificação do cliente que enviou a mensagem, e utiliza a chave pública do cliente para decifrar a mensagem. A obtenção das chaves para assinatura e cifra da mensagem fica a cargo da *framework* WCF, sendo necessário apenas a indicação da localização e função das chaves.

Para que todas as aplicações possam ler os certificados digitais, é utilizado a norma *X.509*, que define o formato dos certificados.

5.2.1.3. Serialização e Deserialização

Para proceder à leitura e construção da mensagem a ser trocada na comunicação via WCF é efectuada a deserialização do ficheiro XML correspondente à mensagem. Para tal foi utilizada a ferramenta *xsd.exe* que consiste numa ferramenta de definição de esquemas XML. Esta operação gera classes a partir de um esquema XSD, onde as classes contêm tipos de argumentos correspondentes àqueles especificados no esquema XSD. Estas classes são utilizadas para ler e escrever a informação correspondente a um documento electrónico utilizado na comunicação. Por outro lado, existe ainda a operação classes para XSD, que é usada para gerar um esquema XML a partir de tipos em arquivos *assembly*. O esquema XML criado pela ferramenta define o formato XML.

5.2.2. Serviços Windows

Um Serviço Windows [49] é uma aplicação que é inicializada quando o Sistema Operativo *Microsoft Windows* é iniciado e corre em *background* enquanto o Sistema Operativo esteja igualmente em execução. Estes serviços podem ser parados e reiniciados, e não têm qualquer interface ao utilizador. Estas características tornam os serviços ideais para serem utilizados em servidores ou sempre que haja necessidade de funcionamento de longa duração, que não interfira com os outros utilizadores que estão a trabalhar no mesmo computador, como é o caso deste projecto.

Os *namespaces* da *framework* .NET necessários para este tipo de projecto são o *System.Configuration*, *System.Configuration.Install* e o *System.Management*. Embora grande parte do trabalho seja feito automaticamente existe ainda a necessidade de implementar código, que controle quais os comandos que podem ser emitidos para os serviços e que acções devem ser tomadas quando esses comandos são recebidos. Os comandos que podem ser emitidos para um serviço incluem iniciar, parar, recomeçar, e finalizar o serviço; pode-se também executar comandos personalizados, isto é, comando implementados pelo utilizador. É ainda necessário preencher a propriedade *ServiceName*, com o nome que se pretende dar ao serviço, criar instaladores necessários para a aplicação serviço, e finalmente reescrever e especificar o código para os métodos *OnStart* e *OnStop* para personalizar o comportamento do serviço, como pode ser visto na Figura 5.8.

```
protected override void OnStart(string[] args){
   timerEvent.Start(); }

protected override void OnStop(){
timerEvent.Stop(); }

public void DisplayTimeEvent(object source, ElapsedEventArgs e){
   string result = brokerInterface.submitMessage(messageToSend);
   … }
```

Figura 5.8 - Código do Serviço Windows de Envio de Mensagens

Por omissão os Serviços Windows executam sobre a conta virtual "*LocalSystem*", que tem direitos administrativos no sistema. *LocalSystem* tem alguns privilégios que nenhuma outra conta administrativa tem, tais como criar contas personalizadas e trancar páginas na memória. Como este utilizador não é "real", isto representa alguns desafios se existir necessidade de guardar dados específicos do utilizador na aplicação, pois não existe uma directoria própria para este utilizador. Outro aspecto importante é que não tem acesso a ficheiros partilhados na rede e recursos similares; se um serviço necessitar de aceder a ficheiros na rede, geralmente é necessário ser configurado para executar como um utilizador do domínio com acesso a esses ficheiros. Mas os Serviços Windows podem também ser configurados para correr sobre qualquer utilizador, embora um utilizador que não seja o corrente requer que seja armazenada uma senha. É o caso deste projecto, em que os serviços executam com a conta de um utilizador com permissões de administrador. Qualquer alteração na senha provocará o não funcionamento do serviço, até que a senha fornecida para o serviço seja também alterada. A aproximação mais segura para a implementação de serviços é configurá-los para correrem sobre contas com permissões limitadas. Os outros dois utilizadores virtuais disponibilizados, *LocalService* e *NetworkService*, são fornecidos para essa finalidade.
Depois de criada e compilada, a aplicação deve ser instalada recorrendo ao comando *InstallUtil.exe* na linha de comando do *Microsoft Visual Studio*, dando como argumento o directório do ficheiro executável do serviço, como se pode verificar na Figura 5.9. Ao contrário de alguns tipos de projectos, neste é necessário criar componentes de instalação para as aplicações serviços. Os componentes de instalação instalam e registam o serviço no servidor e criam uma entrada para o serviço no *Windows Services Control Manager*.

```
installutil IST.TFC.BrokerAdapter.WindowsServiceProcessingMessagesToSend.exe
```

Figura 5.9 - Código para instalação de um Serviço Windows

A interface *Service Control Manager* faz a gestão da inicialização, pausa, finalização e recomeço dos serviços. Uma aplicação que pretenda ser um serviço precisa de ser implementada de forma a comunicar com mensagens de início, fim, pausa, recomeço, etc. do *Service Control Manager*.

5.2.3. Microsoft Message Queue (MSMQ)

O MSMQ [52] é a implementação de filas de espera da Microsoft, actualmente na versão 3.0. Uma fila MSMQ é uma fila de espera para mensagens, que como o próprio nome indica, permite armazenar mensagens numa fila para mais tarde serem processadas. As mensagens são colocadas na fila por uma aplicação "Produtora" e recuperadas e processadas por uma aplicação "Consumidora". Uma fila de espera de mensagens fornece um protocolo de comunicação assíncrona, o que significa que a aplicação remetente e a aplicação receptora da mensagem não necessitam de interagir com a fila de espera ao mesmo tempo. A aplicação remetente não fica bloqueada à espera que alguém obtenha a mensagem e retorne uma resposta, enviam e continua o seu processamento sem se preocupar mais

com a mensagem e em quem a obtém. As mensagens colocadas na fila de espera são armazenadas até que a aplicação receptora as obtenha. Estas duas aplicações podem encontrar-se na mesma máquina, através de uma rede, ou em diferentes máquinas que nem sempre estão ligadas/conectadas. As implementações das filas de espera de mensagens neste projecto são feitas internamente, isto é, dentro de aplicações num mesmo sistema. Tais filas de espera existem apenas dentro do contexto do sistema, isto é, para as finalidades do sistema.

A MSMS é considerada *failsafe* pois procede a novas tentativas de colocação ou obtenção de mensagens na fila se uma primeira transmissão falhar. Assegura uma entrega de confiança colocando mensagens que não conseguem alcançar o seu destino numa fila de espera e então volta a tentar enviar as mesmas assim que o destino é alcançável. Suporta também segurança e definição de prioridades. Filas de espera *Dead Letter* podem ser criadas para situações de envio que excedem o tempo máximo ou para outro tipo de falhas. Isto permite aos utilizadores terem a certeza que as suas mensagens são devidamente colocadas ou recuperadas da fila e chegam correctamente ao seu destino.

Utilizadas neste projecto, a MSMQ também suporta transacções, o que permite operações múltiplas em filas de espera, em que todas as operações são envolvidas numa única transacção, assegurando assim que ou todas as operações são realizadas ou então que nenhuma tenha efeito. Nenhuma transacção deve ser executada parcialmente, isto é, ou se executam todas as operações ou nenhuma. No contexto das mensagens, o conjunto de mensagens passa a ser tratado como um todo, só sendo efectuado o seu envio se todas as entregas forem possíveis. Adicionalmente, a utilização de uma fila transaccional garante que, se o servidor onde a mesma está residente for desligado por qualquer motivo, as mensagens não se perdem, pois foram internamente persistidas para disco rígido.

Primeiro que tudo é necessário efectuar à criação e configuração da fila de espera de mensagens. Existe uma diferença entre criar uma fila de espera ou criar uma instância do componente *Message Queuing*, que se refere a uma fila de espera já existente no sistema operativo, como mostrado no código presente na Figura 5.10. No caso de já existir uma fila de espera com a mesma identificação, é criada uma instância *MessageQueue* que referencia essa mesma fila de espera. No caso de não existir uma fila de espera no sistema com essa identificação, é criada uma nova com a respectiva identificação. Aceder a filas de espera através da *framework* .NET é efectuada através do objecto *System.Messaging.MessageQueue*.

```
System.Messaging.MessageQueue msgsToSendQueue = null;
if (MessageQueue.Exists(@".\private$\msgstosendqueue"))
   /* cria uma instancia MessageQueue, que referencia
    * para msgsToSendQueue ja existente */
   msgsToSendQueue = new System.Messaging.MessageQueue(@".\private$\msgstosendqueue");
else
{
   /* cria uma nova fila privada de nome msgsToSendQueue */
   msgsToSendQueue = MessageQueue.Create(@".\private$\msgstosendqueue", true);
   msgsToSendQueue.SetPermissions(userName, MessageQueueAccessRights.FullControl,
      AccessControlEntryType.Allow);
}
```

Figura 5.10 - Criação de uma fila de espera de mensagens

Após a *message queue* ser criada fica pronta para ser utilizada. Foi implementada uma aplicação produtora para proceder à introdução de mensagens na fila de espera de uma forma transaccional, como se pode verificar a seguir na Figura 5.11, para garantir que a mensagem apenas é colocada na fila de espera caso tudo corra bem. Em caso de erro ou excepção a mensagem é introduzida numa *message queue* de erro (msgstosendqueue_Error).

No *.NET Framework 2.0* existe uma *namespace System.Transactions* que fornece uma estrutura de transacções totalmente integrada ao *.NET Framework*. A classe *TransactionScope* é um das classes incluidas no *System.Transactions*, que permite criar uma instância de *TransactionScope* que garante que nenhuma das operações dentro do *scope* desse *TransactionScope* são executadas parcialmente, isto é, ou se executam todas as operações ou nenhuma. Isto torna-se bastante importante pois o modelo de desenvolvimento está distribuido em camadas, e desta forma a transacção atravessa as várias camadas que estejam implicadas no processo. Qualquer erro que seja detectado, em qualquer uma das camadas, abortara a restante execução.

```
try
{
   /* coloca a mensagem a enviar na MSMQ "msgsToSendQueue" */
   using (TransactionScope ts = new TransactionScope())
   {
      using (MessageQueue mq = new MessageQueue(@".\private$\msgstosendqueue"))
      {
         mq.Send(message, MessageQueueTransactionType.Automatic);
      }
      ts.Complete();
   }
}

catch (Exception ex)
{
   /* Coloca a mensagem na MSMQ de erro "msgsToSendQueueError" */
   using (TransactionScope ts = new TransactionScope())
   {
      using (MessageQueue mq_error = new
         MessageQueue(@".\private$\msgstosendqueue_Error"))
      {
         mq_error.Send(message, MessageQueueTransactionType.Automatic);
      }
      ts.Complete();
   }
}
```

Figura 5.11 - Escrita de mensagens na fila de espera e situação de erro

Finalmente e consequentemente, é implementado o código para consumir as mensagens existentes na fila de espera. Assim, de entre as várias formas de obter as mensagens que existem numa fila de espera, a escolhida foi a utilização do método *GetMessageEnumerator2*. Para grandes filas de espera deve ser utilizado este método pois, ao contrário do método *GetAllMessges* que coloca todas as mesagens lidas da fila de espera em memória, retira uma mensagem de cada vez e coloca apenas a mensagem actual em memória e retira a próxima mensagem e coloca-a em memória apenas depois de ser chamada a operação *MoveNext*. Outro aspecto importante da utilização do método

GetMessageEnumerator2 é que se tem acesso a qualquer mensagem que seja adicionada à fila de espera, mesmo que seja adicionada depois de chamado o método *GetMessageEnumerator2*, isto assumindo à partida que novas mensagens são adicionadas no fim da fila de espera.
Como mensagens, são utilizados objectos serializáveis. Isto significa que é utilizada uma classe costumizada .NET (BrokerCom.BrokerMessaging.Message) instânciada, e que posteriormente é colocada na fila de espera para que outras aplicações a possam ler. Esta classe costumizada é referida com mais pormenor na *Secção 5.3*. Para enviar uma instância de um objecto para a fila de espera basta apenas invocar o método *Send* e passar o objecto instânciado como parâmetro.
Para obter uma mensagem que contém um objecto serializável é necessário indicar que tipo de objecto a mensagem contém, configurando para isso o *formatter* da *MessageQueue*. Isto é feito atribuindo à propriedade *Formatter* da *MessageQueue* um objecto do tipo *System.Messaging.XmlMessageFormatter*. Assim que a *MessageQueue* contenha um objecto do tipo *MessageContent* (no contexto deste projecto o objecto *MessageContent* é *BrokerCom.BrokerMessaging.Message*) é possível configurar o *XmlMessageFormatter* para o tipo do objecto customizado, como se pode observar na Figura 5.12.

```
mq.Formatter = new XmlMessageFormatter ( new Type[] { typeof ( BrokerCom.BrokerMessaging.Message
   ) } );
```

Figura 5.12 - Formatação do tipo de mensagem que é guardada na fila de espera para serialização e deserialização.

Após a formatação da *MessageQueue* já é possível obter o objecto serializado contido em cada mensagem. Ainda assim, é também necessário efectuar um *cast* para o valor retornado através da propriedade *message.Body* para o objecto *MessageContent* que neste caso é *BrokerCom.BrokerMessaging.Message* como se pode verificar na Figura 5.13. Neste ponto, o conteúdo da mensagem é uma versão deserializado do objecto *BrokerCom.BrokerMessaging.Message* original enviado para a fila de espera, e todo as propriedades e valores do objecto são acessíveis.

```
BrokerCom.BrokerMessaging.Message receivedMessage = message.Body as
   BrokerCom.BrokerMessaging.Message;
```

Figura 5.13 - Cast do objecto retornado para o tipo do objecto serializável

5.3. Camada de Construção ou Leitura de Mensagens

Esta camada é responsável pela construção de mensagens para enviar para a Camada de Comunicação ou pela leitura de mensagens recebidas da Camada de Comunicação. Como se pode ver pela Figura 5.14, esta camada é ainda responsável pela gestão de operações a serem executadas na Camada de Comunicação com a Base de Dados e Conversão de Dados, em situações de leitura da base de dados para obtenção de registo para construção de mensagens e em situações de escrita na base de dados para introdução de dados lidos de mensagens.

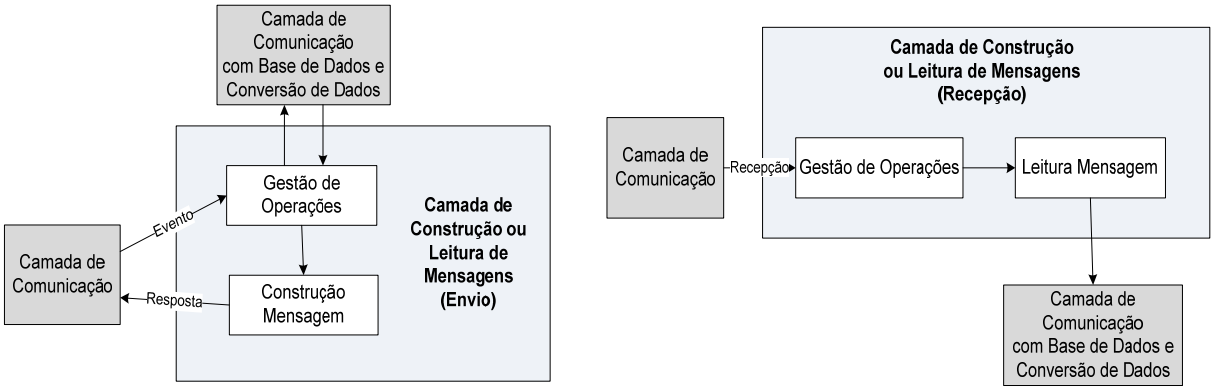

Figura 5.14 - Arquitectura da Camada de Construção ou Leitura de Mensagens

O conteúdo das mensagens está contido no objecto obtido através do processo de deserialização do documento XML, documento anteriormente referido na *Secção 5.2.1.3*. O processo de deserialização gera classes a partir do esquema do documento XML, contendo tipos de argumentos correspondentes àqueles especificados no documento e cujos valores das variáveis de instâncias são lidas igualmente do documento XML. Desta forma, e no contexto do projecto, o objecto *Message* contém todos os componentes de uma factura electrónica, bem como os valores correspondentes como se pode verificar na Figura 5.15.

```
[System.Xml.Serialization.XmlTypeAttribute(AnonymousType = true)]
public partial class Message{
   private MessageMessageHeader messageHeaderField;
   private MessageInvoiceDocument invoiceDocumentField;

   public MessageMessageHeader MessageHeader{
      get { return this.messageHeaderField; }
      set { this.messageHeaderField = value; }
   }

   public MessageInvoiceDocument InvoiceDocument{
      get { return this.invoiceDocumentField; }
      set { this.invoiceDocumentField = value; }
   }
}

public partial class MessageMessageHeader {
   private string messageIDField;
   private string messageTypeField;
   private string stateField;
   private int adapterIDField;
   …
}
public partial class MessageInvoiceDocument{
   private MessageInvoiceDocumentHeader headerField;
   private MessageInvoiceDocumentSupplier supplierField;
   private MessageInvoiceDocumentBuyer buyerField;
   private MessageInvoiceDocumentTermsOfSale termsOfSaleField;
   private MessageInvoiceDocumentItems itemsField;
   private MessageInvoiceDocumentSummary summaryField;
   …
}
```

```
public partial class MessageInvoiceDocumentHeader{
   private string documentIDField;
   private int documentNumberField;
   private string documentDateTimeField;
   private string documentTypeField;
…
}

public partial class MessageInvoiceDocumentSupplier{
   private string nameField;
   private string addressField;
   private int taxpayerIDNumberField;
   private string bankAccountIDNumberField;
…
}

public partial class MessageInvoiceDocumentBuyer{
   private string nameField;
   private string addressField;
   private int taxpayerIDNumberField;
   private string bankAccountIDNumberField;
…
}

public partial class MessageInvoiceDocumentSummary{
   private double totalAmountField;
   private double totalTaxableAmountField;
   private double totalTaxFreeAmountField;
…
}

public partial class MessageInvoiceDocumentTermsOfSale{
   private string noTaxMotiveField;
   private string despachDateField;
   private string deliveryDateField;
   private string expiryDateField;
   private string currencyField;
   private string paymentConditionField;
…
}


public partial class MessageInvoiceDocumentItems{
   private MessageInvoiceDocumentItemsItem itemField;
…
}

public partial class MessageInvoiceDocumentItemsItem{
   private int itemNumberField;
   private string referenceNumberField;
   private string itemDescriptionField;
   private double quantityField;
   private double unitPriceField;
   private double discountField;
   private int taxAmountField;
   private double discountAmountField;
…
}
```

Figura 5.15 - Classes correspondentes aos campos de uma factura electrónica

Tirando partido das propriedades *.NET* (*Get* e *Set*) presentes em cada uma das classes, é possível obter ou escrever o valor de cada campo da mensagem. Para organizar melhor esta informação no processo de leitura ou então para proceder à escrita de uma forma também mais organizada, foi implementado um objecto genérico de nome *Invoice*, como se pode verificar na Figura 5.16.

```
public class Invoice
{
   private Guid _invoiceId;    //Guid do Documento que retiro da BD
   private ArrayList _invoice; // ArrayList com os varios GenericStore (header, supplier
      ou items)

   public void addInvoicePart(GenericStore invoicePart)
   {
      _invoice.Add(invoicePart);
   }
   …
}
```

Figura 5.16 - Objecto *Invoice*

O objecto *Invoice* contém toda a informação correspondente a um documento, tanto para ser enviado como para ser introduzido na base de dados do Sistema ERP. Um objecto deste tipo contém o identificador do documento na base de dados do Sistema ERP, bem como um conjunto de objectos genéricos que contêm a informação respeitante a cada componente de um documento, o *GenericStore*. Na situação de escrita de um documento na base de dados o identificador do mesmo é criado para posterior introdução na base de dados. Um documento é, geralmente, dividido em várias partes tais como informação respeitante ao cabeçalho, ao fornecedor, ao cliente, aos itens que compõem a encomenda, aos termos da venda e a um sumário, como se pode verificar na Figura 5.15. Cada um destes componentes é guardado num objecto *GenericStore* (Figura 5.17).

```
public class GenericStore
{
   private string _entityName; // nome indicativo do componente do documento
   private Object _entityStore; // Objecto contem o nome, valor e tipo do attributo de
      cada um dos componentes do documento
   private bool _isCollection; // variavel indicativa se o objecto e uma Arraylist ou não

   public void addAttributes(string attributeName, Object attributeValue, string
      attributeType)
   {
      Hashtable ht = _entityStore as Hashtable;
      ArrayList aux = new ArrayList();
      aux.Add(attributeValue);
      aux.Add(attributeType);
      ht.Add(attributeName, aux);
      _entityStore = ht;
   }

   public void setEntityStoreType(bool isCollection){
      _isCollection = isCollection;
      if (_isCollection)
         _entityStore = new ArrayList(); // ArrayList com varias Hashtables, para situacao
            de varios itens
      else
         _entityStore = new Hashtable(); // Hashtable em que a Key é o nome do atributo, e
            como valor tem uma ArrayList em que na posicao "0"-> valor e na "1"-> tipo
   }
   …
}
```

Figura 5.17 - Objecto *GenericStore*

A forma como é guardada a informação neste objecto difere um pouco de acordo com a situação em que se encontra, isto é, na situação de leitura dos campos da base de dados ou da leitura dos

campos da mensagem recebida. Em ambos os casos os campos são guardados de acordo com a estrutura configurada no repositório de metadados. Isto é, para a leitura através da base de dados a estrutura tem como base as operações de obtenção de dados da base de dados, para a leitura da mensagem recebida a estrutura tem como base as operações de escrita de dados na base de dados. Esta estrutura é configurada no repositório de metadados, contendo informação sobre as tabelas, as relações entre tabelas e os campos que compõem essa mesma operação na base de dados do Sistema ERP. Desta forma os campos são guardados no objecto de acordo com uma identificação e tipo especificado no repositório de metadados.

5.3.1. Processamento do Repositório de Metadados

A *.NET Framework* possui um *namespace* para trabalhar com XML, o *System.XML*. Este *namespace* contém, entre outras, as classes *XMLTextReader* e a classe *XMLTextWritter* responsáveis pela manipulação de documentos XML, basicamente a leitura e a escrita. A classe *XMLTextReader* fornece processamento directo e tokenizado de XML, possibilitando somente a leitura, e o acesso é feito num único sentido (para a frente). Permite compilar um documento em formato XML e criar um objecto em memória que representa esse documento, que no caso deste projecto corresponde ao objecto *Metadata*. A partir desse objecto, as classes correspondentes aos elementos do documento XML (Figura 5.18) fornecem uma série de métodos para aceder aos atributos do documento XML como sendo uma árvore. Esta classe *XMLTextReader* foi utilizada para efectuar o processamento do repositório de metadados, escrito sobre XML.

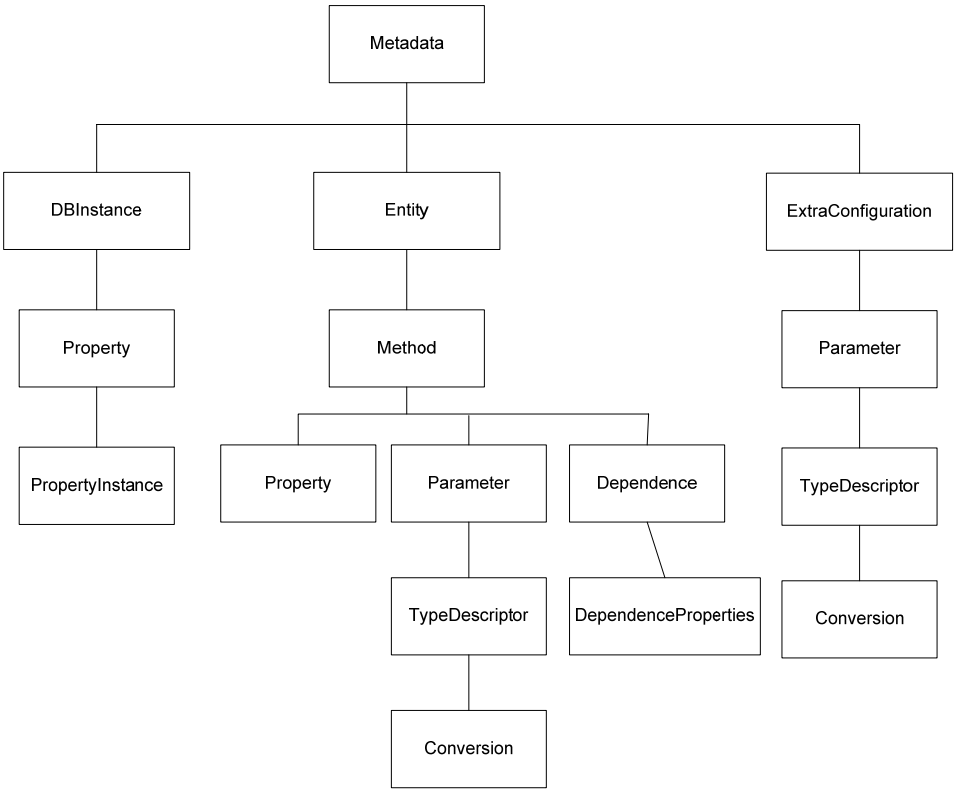

Figura 5.18 - Classes Correspondentes aos elementos do repositório de metadados

O nó actual refere-se ao nó onde o *reader* se encontra posicionado. O *reader* avança usando qualquer um dos métodos de leitura ou propriedades que reflectem o valor do nó actual. O *XMLTextReader* fornece informação sobre localização em termos de linha e carácter, informação que pode ser útil no diagnóstico de um erro no XML.
Para processar os dados contidos no documento XML, é garantido que cada registo contém um tipo de nó que pode ser determinado pela propriedade *NodeType*. Depois da enumeração *NodeType* retornar o tipo do nó, pode-se testar o mesmo para verificar se é um elemento ou um tipo de um documento XML. Se for qualquer um destes dois tipos, processa-se o nó utilizando as propriedades *Name* e *Value* para obter detalhes sobre o nó. A propriedade *Name* retorna o nome do nó (por ex. *Entity*), enquanto que a propriedade *Value* retorna o valor do nó (texto do nó) do nó actual.

5.3.2. Gestão de Operações

Esta camada é responsável por muitas das operações realizadas durante a execução do *Invoice Adapter*, tais como pela verificação e obtenção de novos documentos no Sistema ERP, leitura e escrita de mensagens como explicado anteriormente, e também pela invocação de métodos responsáveis pela introdução ou leitura de informação do Sistema ERP na Camada de Comunicação com a Base de Dados e Conversão de Dados.

5.3.2.1. Verificação e obtenção de novos documentos no Sistema ERP

A comparação entre novos documentos emitidos e documentos previamente tratados tornou-se mais complicado do que o esperado, tendo sido tomadas algumas decisões como explicado de seguida. Na análise feita aos dados produzidos pelo Primavera Express verificou-se que os campos do tipo data não tinham a granularidade pretendida, o que dificultou a tarefa de detecção de novos documentos num mesmo dia. Isto é, a data de emissão de um documento continha apenas informação relacionada com o ano, dia e mês, não contendo informaçao sobre hora, minutos e segundos. A solução adoptada consistiu na utilização de um objecto LastInvoiceInformation (Figura 5.19) contendo a última contagem de documentos presentes na base de dados, a data do último documento emitido e um conjunto de objectos do tipo GenericConfigurationInformation (Figura 5.20) contendo informação relacionada com os documentos. Este objecto genérico contém a identificação dos campos necessários para identificação de um documento, bem como os seus respectivos valores na base de dados, obtidos atavés do repositório de metadados. Desta forma, passam a conter os atributos e valores correspondentes aos últimos documentos emitidos na base de dados, que foram anteriormente convertidos e transformados em mensagens enviadas para o *Message Broker*.

```
public class LastInvoiceInformation{
   int _invoicesCount;
   DateTime _lastInvoiceDateTime;
   ArrayList _invoicesInformation;

   public void addInvoiceInformation(GenericConfigurationInformation invoiceInformation)
   {
      _invoicesInformation.Add(invoiceInf);
   }
   … }
```

Figura 5.19 - Objecto *LastInvoiceInformation*

```
public class GenericConfigurationInformation{
   private Hashtable _configurationInformation;

   public void addInvoiceInformation(string attributeName, Object attributeValue, string
      attributeType){
      ArrayList attribInfo = new ArrayList();
      attribInfo.Add(attributeValue);
     attribInfo.Add(attributeType);
     _configurationInformation.Add(attributeName, attribInfo);
   }
   … }
```

Figura 5.20 - Objecto Genérico *GenericConfigurationInformation*

Para detectar a existência de novos documentos na base de dados é feita uma comparação entre a contagem actual de documentos com a contagem obtida anteriormente no início de processamento do adaptador. Um dos primeiros passos do adaptador é obter os documentos existentes na base de dados, e preenchimento do objecto *LastInvoiceInformation* com esses documentos, a contagem de documentos, e a data do último documento. Com esta informação pode-se proceder à detecção de novos documentos e processamento dos mesmos, comparando os documentos emitidos desde a data do último documento (novos documentos), com os documentos anteriormente tratados.

5.3.2.2. Invocações à Camada de Comunicação com a Base e Dados e Conversão de Dados

Existem dois tipos de acontecimentos que geram a comunicação desta camada com a Camada de Comunicação com a Base de Dados e Conversão de dados (Camada da Base de Dados), o recebimento de mensagens com informação a ser escrita na base de dados e a existência de novos documentos na base de dados para serem obtidos e depois enviados. Como referido anteriormente, ao chegarem novas mensagens ao adaptador, a informação contida nessas mesmas mensagens é guardada no objecto *Invoice*. Esse objecto serve essencialmente para guardar a informação de uma forma organizada para quando é invocada na Camada da Base de Dados a função responsável pela escrita na base de dados. Os campos da mensagem são guardados no objecto *Invoice* de acordo com a informação contida no repositório de metadados, que indica a identificação de cada campo a ser guardado no objecto. Este objecto é reencaminhado para a Camada da Base de Dados, que é responsável por escrever na base de dados o seu conteúdo de acordo com a informação obtida através do repositório de metadados correspondente aos campos e tabelas onde deve ser guardada.

O outro acontecimento é na detecção de novos documentos na base de dados, caso em que é necessário ler a informação contida nos mesmos. Como referido na *Secção 5.3.2.1*, existe um mecanismo de detecção de novos documentos, e em que é utilizado o campo de identificação do documento na tabela para leitura do mesmo posteriormente. Com a identificação do novo documento, é invocada uma função na Camada da Base de Dados que obtém toda a informação contida na base de dados relacionada com esse documento. Novamente, a indicação dos campos e tabelas onde deve ser lida a informação relacionada com o documento é obtida através do repositório de metadados. Essa informação é guardada no objecto *Invoice* com a identificação e estrutura obtida através do repositório de metadados, e enviada de volta para a Camada de Construção ou Leitura de Mensagens.

5.4. Camada de Comunicação com Base de Dados e Conversão de Dados

Esta camada é responsável pela comunicação com a persistência dos dados de um Sistema ERP, guardados numa base de dados legada. Genérica e bastante complexa, esta camada serve essencialmente para integração de dados, suportando várias funcionalidades tais como:

- Extracção de dados de uma base de dados, transformação ou conversão dos mesmos num formato normalizado e carregamento dos dados num objecto genérico para posteriormente ser utilizado pela Camada de Construção ou Leitura de Mensagens, como mostrado na Figura 5.21;
- Obtenção de dados de um objecto genérico, transformação ou conversão dos mesmos num formato relacional e carregamento dos dados numa base de dados, como mostrado na Figura 5.22.

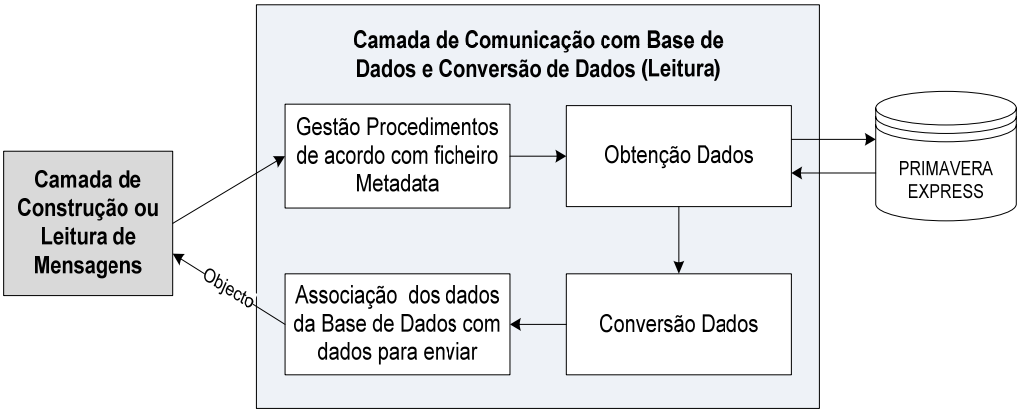

Figura 5.21 – Arquitectura da Camada de Comunicação com a Base de Dados e Conversão de Dados (leitura de dados).

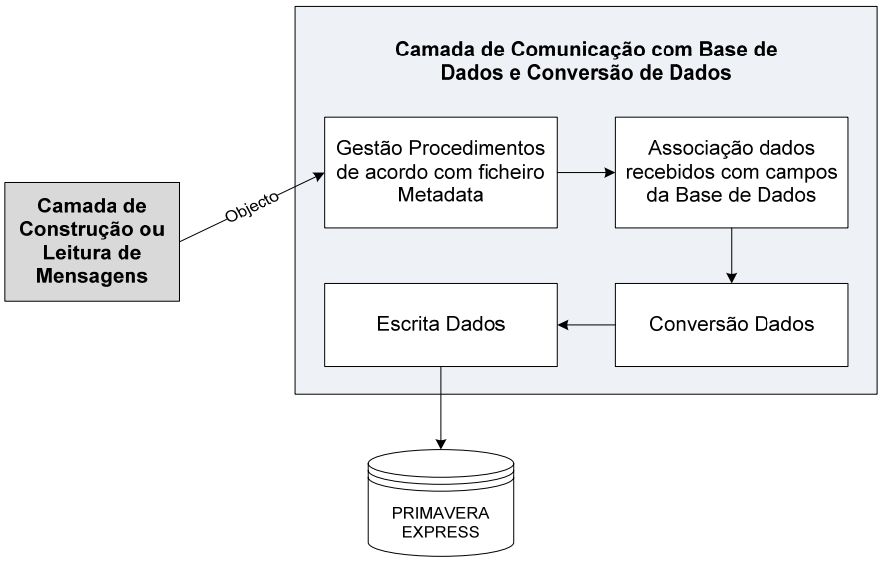

Figura 5.22 - Arquitectura da Camada de Comunicação com a Base de Dados e Conversão de Dados (escrita de dados).

Os acessos às bases de dados são feitos de forma eficiente e adaptável a qualquer tipo de base de dados, em qualquer Sistema ERP de acordo com um repositório de metadados, como referido na *Secção 4*.

A escolha da base de dados como elemento integrador do adaptador com o Sistema ERP deve-se a factores tais como:

- As bases de dados são sistemas robustos, confiáveis, que nunca perdem os dados;
- As bases de dados têm, por definição, a versão mais recente dos dados; as bases de dados já têm interfaces para serem acedidas por aplicações, local ou remotamente e, para além disso, essas interfaces são normalmente bastantes fáceis de utilizar, baseadas em normas como *SQL*, e disponíveis para vários sistemas operativos e linguagens de programação;
- As bases de dados, principalmente se forem relacionais, têm um formato bem conhecido, que garante algumas restrições de integridade, e relativamente fáceis de navegar.

O sistema de gestão da Base de Dados assumido é o *Microsoft SQL Server*. Para aceder e manipular os dados da base de dados *SQL Server* utilizaram-se *queries*. As *queries* fornecem uma maneira de obter e alterar dados de uma base de dados. A linguagem de programação utilizada para manipular os dados na Base de Dados *SQL Server* foi *SQL*, em que as instruções utilizadas foram :

- SELECT: instrução que permite obter os registos armazenados na Base de Dados;
- INSERT: instrução que permite adicionar novos registos à Base de Dados;
- UPDATE: instrução que permite modificar os registos existentes na Base de Dados.

A linguagem de programação *SQL* é uma linguagem normalizada para acesso a bases de dados relacionais, actualmente suportada por virtualmente todos os sistemas de bases de dados relacionais existentes. É apenas uma linguagem, não uma interface normalizada com a base de dados. Isto é, um programador pode escrever facilmente em *SQL* uma pergunta à base de dados, cuja resposta será (de acordo com o modelo relacional) um conjunto de valores. É neste contexto, e para que seja

possível enviar *queries* de qualquer tipo (como anteriormente enumeradas) e receber resposta de volta da base de dados que foi utilizada a tecnologia *ADO.NET*, interface com o servidor de base de dados *Microsoft SQL Server*.

5.4.1. ADO.NET

A *.NET Framework* fornece um conjunto de classes para facilitar o acesso das aplicações a bases de dados de diversos tipos, cujo nome é *ADO.NET [53]*. Para ter acesso às funcionalidades do *ADO.NET* é necessário importar o conjunto de classes fornecidas pelo *SQL Server .NET Data Provider*: *System.Data* e *System.Data.SqlClient*.

Os principais objectos utilizados do *SqlServer .NET Data Provider* são:

- *SqlConnection*: representa uma única conexão persistente ao *SQL Server* origem dos dados;
- *SqlCommand*: representa algum comando que pode ser executado, tipicamente uma *query*;
- *SqlDataReader*: representa a maneira mais rápida para obter informação de uma base de dados *Sql Server*.

Os objectos anteriormente mencionados foram utilizados nas operações de leitura, escrita e consulta de informação das diversas entidades sobre a base de dados. Antes de mais, referir que toda a informação necessária para estabelecer conexões com a base de dados, bem como *queries* de execução e respectivos parâmetros, e finalmente campos que irão ser retornados é obtida através do repositório de metadados em componentes específicos, como referido na *Secção 4*.

Para abrir as ligações com o repositório de dados do Sistema ERP é utilizado um objecto *SqlConnection*. Ao estabelecer esta conexão é necessário definir a propriedade *ConnectionString* do objecto *SqlConnection*. Para tal são definidos parâmetros tais como o nome do servidor da base de dados, o nome da base de dados que contém a informação que se pretende, e parâmetros de autenticação.

Para operações de leitura e escrita de dados é utilizado o objecto *SqlCommand* com a *query* a ser executada mediante um critério de leitura ou escrita definido na operação a ser efectuada sobre a base de dados. Ao contrário do que sucede na maioria dos casos, a definição de um critério de leitura ou escrita é definida de uma forma genérica, através de uma *query* dinâmica. Isto é, como a ideia da utilização de um repositório de metadados é poder definir estes parâmetros de uma forma genérica, a forma como são definidos os critério de leitura ou escrita é igualmente efectuada de uma forma genérica.

Para efectuar a leitura dos registos devolvidos numa *query*, é utilizado o objecto *SqlDataReader*. O conjunto de resultados, o dito *resultset*, contido num *SqlDataReader* é *forward-only* e *read-only*, o que significa que, só se podem ler os registos de um *resultset* sequencialmente a partir do início ao fim, não se podendo modificar quaisquer dados nem voltar a ler registos anteriores. Cada registo do

conjunto de resultados é colocado no objecto genérico *GenericStore*, objecto criado dinamicamente para guardar de forma genérica os dados vindos da base de dados para posterior manipulação e conversão para a criação do documento para ser trocado.
Para efectuar operações de escrita é utilizado o objecto genérico *GenericStore*, objecto anteriormente criado e preenchido no momento de manipulação da mensagem recebida com os dados da mesma. Tal como na operação de leitura, é igualmente necessário utilizar um objecto *SqlConnection* e *SqlCommand*, definindo as suas propriedades e critérios de escrita. Ao invés da operação de leitura, onde é utilizado o método *ExecuteReader()* do objecto *SqlCommand*, na operação de escrita utiliza-se o método *ExecuteNonQuery()*, pois este método não retorna um *resultset* aquando da execução da *query* (inserção de dados na base de dados).

5.4.2. Leitura e Escrita Genérica da Base de Dados

O acesso à base de dados do Sistema ERP é efectuado recorrendo aos objectos que contêm informação do repositório de metadados, como referido anteriormente na *Secção 5.3.1*. Os parâmetros introduzidos no ficheiro de configuração relacionados com conexão, obtenção e escrita de dados na base de dados são utilizados por esta camada para, desta forma, tornar o acesso genérico sem necessidade de programação. O processo de acesso à base de dados não é visível ao utilizador, executando o mesmo automaticamente de acordo com o evento que o despoletou, isto é, na chegada de mensagens do *Message Broker* ou detecção de novas mensagens na base de dados.
A *query* a ser executada na base de dados é baseada numa string obtida através da classe *Property* e, no caso de existirem parâmetros de entrada na classe *Parameter*, na conjunção da mesma com estes parâmetros do método em execução. O resultado de execução da *query* depende da existência e indicação de campos de retorno, obtidos também através da classe *Parameter*.
Para a situação de leitura de dados, normalmente existe apenas um parâmetro de entrada, correspondente à chave primária da tabela que se pretende obter informação. A informação retornada pela *query* corresponde aos campos indicados como parâmetros de retorno. Estes campos são obtidos de acordo com uma identificação correspondente à identificação do campo na base de dados. A informação é guardada no objecto *Invoice* (referido na Figura 5.3), de acordo com a identificação e tipo de dados obtido através dos parâmetros de retorno. Em situações de diferença entre o formato relacional e formato normalizado dos dados, é feita uma conversão de acordo com a classe *Conversion,* como explicado na *Secção 5.4.3*. No caso de existência de campos com dependências com outras tabelas, é utilizado a informação contida na classe *Dependence*, que indica qual o campo que contém dependência e respectivo valor, valor esse que é utilizado para obter a informação na tabela dependente. Normalmente isto traduz-se num campo que contém uma chave estrangeira para outra tabela, e a informação que se pretende corresponde a um outro campo ou na situação inversa, num campo que contém informação normal e o que se pretende na tabela dependente é a sua chave estrangeira. A informação obtida é guardada no campo que tem dependência, substituindo assim a informação que continha anteriormente.

Para a situação de escrita de dados na base de dados, os parâmetros de entrada correspondem aos dados da mensagem recebida. Estes mesmos dados são obtidos do objecto *Invoice* recorrendo à identificação dos campos contidos na classe *Parameter*. Quer no caso dos formatos dos dados serem diferentes, como no caso de existirem campos com dependências com outras tabelas, são efectuados os mesmos passos que na situação de leitura de dados na base de dados. De referir apenas que as dependências podem ser utilizadas para obtenção de dados que não são contemplados numa mensagem, sendo assim obtidos através de campos presentes na mensagem. Um exemplo ocorrido durante este trabalho, deveu-se ao facto de não estar disponível um campo necessário à escrita numa tabela, e como tal, foram utilizados alguns campos da mensagem para obter o mesmo na base de dados.

5.4.3. Conversão dos Dados

A conversão ou transformação dos dados entre dois formatos, do formato relacional para um formato normalizado e acordado previamente entre os Sistemas ERP intervenientes é um dos aspectos mais importantes deste trabalho. É com esta solução que se propõem resolver problemas de incompatibilidades entre Sistemas diferentes.

A solução adoptada parte da utilização de elementos e atributos específicos no repositório de metadados tais como o elemento *Conversion*, referido anteriormente na *Secção 4.3*. O elemento *TypeDescriptor* corresponde a cada campo a ser lido ou escrito na base de dados e a informação contida nos seus atributos *TypeName* e *TypeDBName* corresponde ao tipo de dados em que se pretende transformar e o tipo de dados utilizado na base de dados, respectivamente. É feita uma comparação entre estes dois formatos, e caso sejam diferentes, é feita a conversão do formato relacional para o formato normalizado.

O elemento *Conversion*, por outro lado, serve para conversões extras e antes da conversão para o formato normalizado. Isto é, podem existir situações em que é necessário transformar o formato de um campo para um certo tipo para se poder proceder à conversão para o formato final. Um exemplo de utilização deste tipo de conversão, como se pode verificar na Figura 5.23, pode ser a conversão do formato da data do formato alfanumérico para o formato das datas. Após esta conversão é que é feita a conversão para o formato normalizado, que neste caso seria novamente para o formato alfanumérico. De salientar apenas o formato normalizado pretendido *"System.String.English"*, que corresponde a converter uma data do formato de data actual para o formato inglês antes de converter para o formato alfanumérico.

```
<TypeDescriptor TypeName="System.String.English" Name="DespatchDate"
  TypeDBName="System.String" DBName="DataCarga">
    <Conversion DBType="System.String" ConversionType="System.DateTime"></Conversion>
</TypeDescriptor>
```

Figura 5.23 - Exemplo de utilização de elementos no repositório de metadados para conversão de formatos de dados

Este tipo de conversões de formato de dados estão frequente presentes em produtos de EAI ou ESB's (como é o caso do *Message Broker* com que o adaptador interage), existindo "*mappers*" que geralmente permitem especificar visualmente as conversões a efectuar. No caso do adaptador, a funcionalidade pretendida é simples e *lightweight*, não fazendo sentido recorrer a um daqueles produtos.

5.5. Resumo

Neste capítulo foram explorados todos os passos relativos à implementação do adaptador. Após levantamento dos requisitos para o adaptador a desenvolver e das principais decisões ao nível tecnológico terem sido tomadas, procedeu-se à implementação do mesmo.

Foram focadas as várias camadas em que se divide a arquitectura deste adaptador, correspondendo a uma camada responsável pela comunicação, a uma camada responsável pela leitura e construção das mensagens recebidas, e finalmente uma camada responsável pela comunicação com a base de dados e conversão dos dados. Para cada uma destas camadas é explicado o fluxo de funcionamento, bem como tecnologias utilizadas para a sua implementação e decisões tomadas.

O modelo relacional do Sistema ERP é obtido através de um repositório de metadados, sendo através deste ficheiro que todos os aspectos de comunicação com a base de dados são obtidos, bem como relacionados com a conversão dos dados, de forma genérica.

No capítulo seguinte irá ser apresentado um caso de estudo para ilustrar o funcionamento do adaptador descrito ao longo deste capítulo. Como a principal contribuição deste trabalho é a descoberta e implementação de um adaptador capaz de integrar vários Sistemas ERP diferentes num cenário de mediação electrónica (factura electrónica), os exemplos vão se limitar à demonstração do funcionamento do mesmo na recepção e envio de mensagens.

6. Caso de Estudo

Nos capítulos anteriores foram abordados os conceitos e tecnologias mais relevantes utilizados para a implementação de um adaptador de integração entre empresas. Neste Capítulo apresenta-se um caso de estudo para o adaptador implementado de acordo com o que foi explicado nos Capítulos anteriores.

O caso de estudo utilizado refere-se à emissão de facturas electrónicas, aquando de uma transacção comercial entre duas empresas, e envio da mesma. Neste caso irá ser mostrado exemplo de um documento, factura electrónica, utilizado pelo Sistema ERP Primavera Express, e esquemas do fio de funcionamento do adaptador. Serão também mostradas algumas imagens do resultado do processamento em algumas ferramentas utilizadas.

6.1. Integração com PRIMAVERA EXPRESS

6.1.1. Cenário de Negócio

Esta secção descreve um cenário comum de desenvolvimento de repositório de metadados e utilização do adaptador implementado numa situação real.

6.1.1.1. Desenvolvimento do Repositório de Metadados

Uma analista do negócio do Sistema ERP Primavera Express define os requisitos de negócio e comunica os resultados ao autor do repositório de metadados. O autor dos metadados cria o ficheiro de configuração relacionado com os metadados de acordo com os requisitos de negócio do sistema envolvido, definindo as entidades e configurações extra. Os componentes relacionados com aspectos de conexão com o sistema é implementado posteriormente por um administrador do sistema envolvido.

6.1.1.2. Utilização do Adaptador

Com o repositório de metadados construído, e o adaptador instalado do servidor do Sistema ERP Primavera Express que tem comunicação com os sistemas exteriores, o mesmo estará pronto para futuras utilizações. Como exemplo, uma série de produtos vendidos a um cliente, e consequente produção de uma factura electrónica. Essa mesma factura electrónica traduz-se em campos e tabelas na base de dados do Primavera Express, que são obtidos e convertidos para um formato canónico através da informação obtida do repositório de metadados. Após construída uma mensagem num formato acordado entre os intervenientes, a mesma é enviada para o *Message Broker* contendo um identificador do sistema para sua identificação perante o *Message Broker*.

Na outra situação possível, a compra de produtos a um fornecedor em que é recebida uma mensagem no formato acordado. É obtida a informação correspondente a uma factura electrónica,

sendo essa informação convertida para o formato relacional da base de dados do Primavera Express, utilizando o repositório de metadados. Utilizando novamente o repositório de metadados, este repositório indica ao adaptador onde introduzir a informação já convertida nas tabelas e campos da base de dados do Primavera Express.

6.1.2. Documentos Envolvidos

Nesta secção irá ser mostrado um modelo de uma factura utilizada para mediação electrónica (Figura 6.1), bem como os campos e valores presentes na base de dados (Figura 6.2, Figura 6.3, Figura 6.4, Figura 6.5) para construção de uma mensagem para posterior envio para o *Message Broker*.

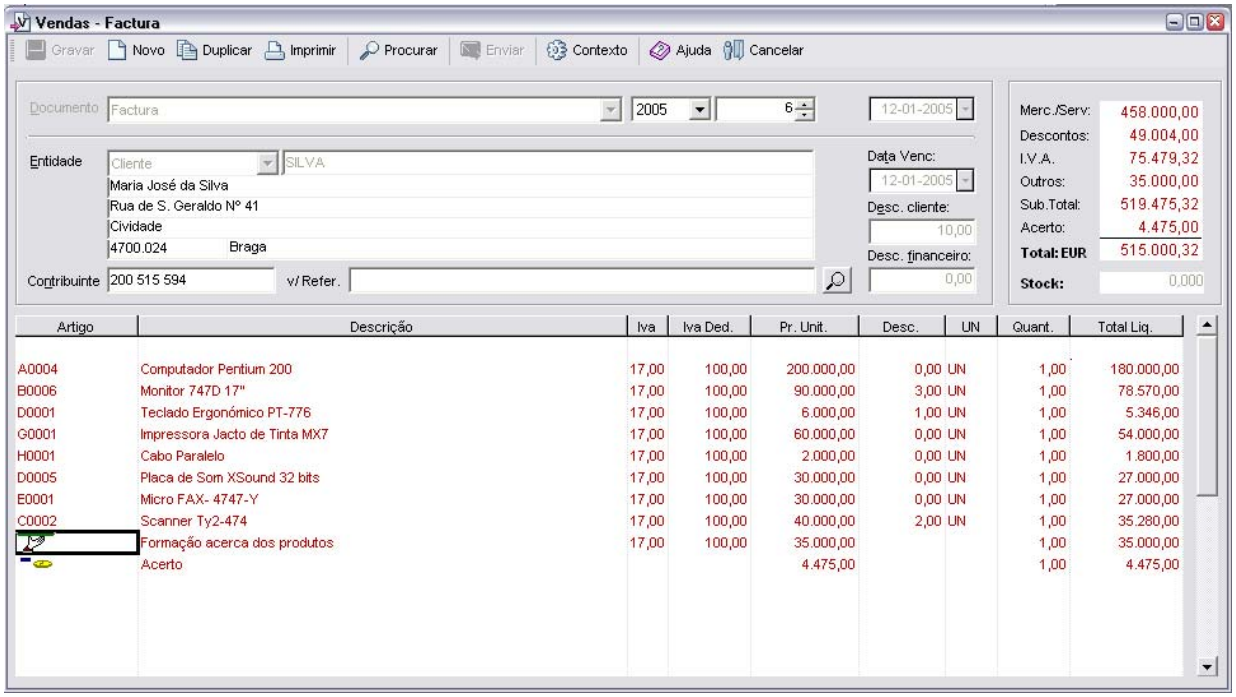

Figura 6.1 - Exemplo de uma factura utilizada para mediação electrónica

Data	TipoD...	NumDoc	CondPag	TotalMerc	Totall...	TotalDesc	TotalOutros	Moeda	DataVencimento	DataCarga	DataDescarga
2005-01-12 00:00:00.000	FA	6	8	458000	75479	49004	30525	EUR	2005-01-12 00:00:00.000	14-01-2005	25-01-2005

Figura 6.2 - Valores na tabela CabecDoc utilizados para preenchimento dos campos do cabeçalho no documento para enviar

Entidade	Nome	Morada	NumContribuinte	DescEntidade
SILVA	Maria José da Silva	Rua de S. Geraldo Nº 41	200 515 594	10

Figura 6.3 - Valores na tabela CabecDoc utilizados para preenchimento dos campos do cliente no documento para enviar

IDNome	IDMorada	IFNIF
Empresa de Demonstração (PRIVA)	Lisboa	123456789

Figura 6.4 - Valores na tabela Empresas utilizados para preenchimento dos campos do fornecedor no documento para enviar

NumLinha	Artigo	Taxal...	Quantidade	PrecUnit	DescontoComercial	Desconto1	Descricao
2	A0004	17	1	200000	20000	0	Computador Pentium 200
7	D0005	17	1	30000	3000	0	Placa de Som XSound 32 bits
4	D0001	17	1	6000	654	1	Teclado Ergonómico PT-776
3	B0006	17	1	90000	11430	3	Monitor 747D 17"
9	C0002	17	1	40000	4720	2	Scanner Ty2-474
5	G0001	17	1	60000	6000	0	Impressora Jacto de Tinta MX7
10	NULL	17	1	35000	0	0	Formação acerca dos produtos
8	E0001	17	1	30000	3000	0	Micro FAX- 4747-Y
6	H0001	17	1	2000	200	0	Cabo Paralelo
11	NULL	0	1	4475	0	0	Acerto
1	NULL	0	0	0	0	0	NULL

Figura 6.5 - Valores na tabela LinhasDoc utilizados para preenchimento dos campos dos itens no documento para enviar

Os campos obtidos através das tabelas anteriormente referenciadas correspondem a campos necessários para o preenchimento do documento electrónico a enviar. A correspondência destes campos com os campos de uma factura electrónica é mostrada na *Secção 4.2*, bem como o seu formato relacional.

Os campos estão principalmente relacionados com características do cliente, do fornecedor (Primavera Express) e dos produtos que estão a ser comercializados.

6.2. Configurações do Repositório de Metadados

O mais importante para o bom funcionamento do adaptador é a construção de um ficheiro de configuração para o repositório de metadados. Este ficheiro irá conter toda a informação necessária para acesso aos dados do Sistema ERP Primavera Express, bem como informação para conversão e construção do documento que irá ser transaccionado. Este ficheiro de configuração é feito sobre XML.

6.2.1. Caso Particular

Nesta secção irá ser mostrado como construir o ficheiro de configuração, de forma a obter a informação presente na factura anteriormente mostrada na Figura 6.1. Esta factura foi produzida através do sistema ERP Primavera Express. Como referido na *Secção 0*, este sistema tem como suporte de base de dados o *SQL*, e uma fonte de dados própria criada no processo de instalação no servidor *SQL Server* com o nome de "*PRIEXPRESS*". Dentro dessa fonte de dados são criadas bases de dados de acordo com as empresas que utilizem o sistema, isto é, existem tantas bases de dados quantas as empresas que utilizem este sistema. No caso deste projecto foi utilizada a base de dados criada no processo de instalação como uma base de dados de demonstração, contendo de início já informação relacionada com uma empresa fictícia. De referir a existência de uma base de dados com o nome "*PRIEMPRE*", criada pelo sistema, onde se encontra informação relacionada com as empresas intervenientes neste sistema.

Prosseguindo para a construção do ficheiro de configuração, o primeiro passo é introduzir os parâmetros de conexão com a base de dados "*PRIEMPRE*", e a base de dados utilizada para demonstração. O código que se segue na Figura 6.6 refere-se a base de dados utilizada para

demonstração, sendo que o código para a outra base de dados apenas difere nos parâmetros, e não na estrutura do documento.

```
<DBConfiguration>
  <DBInstance Name="PRIDEMO">
    <Properties>
    <Property Name="DataSource"
      Type="System.String"><nome_servidor>\PRIEXPRESS</Property>
    <Property Name="Database" Type="System.String">PRIDEMO</Property>
    <Property Name="Login">
      <PropertyInstance Name="UID" Type="System.String">sa</PropertyInstance>
      <PropertyInstance Name="PWD" Type="System.String">primavera </PropertyInstance>
    </Property>
    </Properties>
  </DBInstance>
</DBConfiguration>
```

Figura 6.6 - Configuração dos parametros de conexão com as bases de dados do Primavera Express

Embora o ficheiro de configurações contenha mais do que vai ser mostrado de seguida, apenas vão ser abordadas duas entidades nesta secção, uma respeitante aos parâmetros necessários para a obtenção de informação da base de dados, e outra entidade respeitante aos parâmetros necessários para introduzir informação na base de dados. Será também abordado o elemento *ExtraConfigurations* respeitante a operações fora do âmbito normal das entidades (leitura ou escrita da base de dados de informação relacionada com uma factura electrónica), mas necessária para o funcionamento da operação de obtenção dos últimos documentos emitidos pelo sistema Primavera Express. Finalmente, será mostrado como é feita a associação entre duas tabelas e obtenção do campo pretendido.

Como já foi referido na *Secção 4.3*, uma entidade é um objecto do negócio, tal como os clientes ou produtos, e os métodos são as operações relacionadas com uma entidade. Uma entidade pertence a um único sistema e deve ter um nome identificador único. As entidades contêm identificadores e métodos. A Figura 6.7 mostra o resultado do passo seguinte, onde irá ser definido uma entidade de nome *Buyer* na base de dados *PRIDEMO*, e é define um método de nome *GetBuyer* de forma a obter os dados do cliente para onde a factura electrónica irá ser enviada.

```
<Entities>
  <Entity Name="Buyer">
    <Identifiers>
      <Identifier Name="Entidade" Type="System.String"></Identifier>
    </Identifiers>
    <Methods>
      <Method Name="GetBuyer" Source="PRIDEMO">
        <Properties>
          <Property Name="FormatString" Type="System.String">SELECT Entidade, Nome, Morada,
            NumContribuinte FROM CabecDoc WHERE Id = {0} </Property>
        </Properties>
        <Parameters>
          <Parameter Direction="In" Name="InvoiceId">
            <TypeDescriptor TypeName="System.Guid" IdentifierName="InvoiceId">
            </TypeDescriptor>
          </Parameter>
          <Parameter Direction="Return" Name="Header">
<TypeDescriptor TypeName="System.Data.IDataReader" IsCollection="false">
```

```
          <TypeDescriptors>
             <TypeDescriptor TypeName="System.String" Name="Name"
               TypeDBName="System.String" DBName="Nome"></TypeDescriptor>
             <TypeDescriptor TypeName="System.String" Name="Address"
               TypeDBName="System.String" DBName="Morada"></TypeDescriptor>
             <TypeDescriptor TypeName="System.String" Name="TaxpayerIdentificationNumber"
               TypeDBName="System.String" DBName="NumContribuinte"></TypeDescriptor>
             <TypeDescriptor TypeName="System.String" Name="BankAccountIDNumber"
               TypeDBName="System.String" DBName="Entidade"></TypeDescriptor>
          </TypeDescriptors>
        </TypeDescriptor>
        </Parameter>
      </Parameters>
      <Dependences>
        <Dependence Name="GetBuyerBankAccountIDNumber" Query="SELECT NIB FROM Clientes
          WHERE Cliente = '{0}' ">
          <DependenceProperties>
             <DependenceProperty Name="MainAttribute" Type="System.String"
               Identifier="BankAccountIDNumber"></DependenceProperty>
             <DependenceProperty Name="DependenceIdentifierAttribute"
               Type="System.String" Identifier="Entidade"></DependenceProperty>
             <DependenceProperty Name="ResultAttribute" Type="System.String"
               Identifier="NIB"></DependenceProperty>
          </DependenceProperties>
        </Dependence>
      </Dependences>
    </Method>
  </Methods>
</Entity>
```

Figura 6.7 - Exemplo de código relacionado com métodos para leitura de informação da base de dados.

De referir o uso do elemento Dependences para resolver a questão da informação relacionada com o campo NIB se encontrar na tabela Clientes. Para relacionar as duas tabelas é usado o campo Entidade, utilizando o valor guardado no campo BankAccountIDNumber através do TypeDescriptor relacionado com o mesmo, e desta forma obter o campo NIB correspondente a essa Entidade.

De seguida, como mostrado na

Figura 6.8, irá ser definida uma entidade de nome Item na base de dados PRIDEMO, e define um método de nome SetItem de forma a introduzir os dados respeitantes a um item de uma factura electrónica recebida na base de dados do Primavera Express.

```
<Entity Name="Item">
  <Identifiers>
    <Identifier Name="Id" Type="System.Guid"></Identifier>
  </Identifiers>
  <Methods>
    <Method Name="SetItem" Source="PRIDEMO">
      <Properties>
         <Property Name="FormatString" Type="System.String">INSERT INTO
           LinhasCompras([NumLinha],[Artigo],[TaxaIva],[Quantidade],[PrecUnit],[DescontoCom
           ercial],[PrecoLiquido],[Descricao],[IdCabecCompras],[Id]) VALUES ({0})
           </Property>
      </Properties>
      <Parameters>
         <Parameter Direction="In" Name="Item">
           <TypeDescriptor TypeName="System.Int16"
             IdentifierName="InvoiceLineNumber"></TypeDescriptor>
```

```
          <TypeDescriptor TypeName="System.String"
            IdentifierName="ItemNumber"></TypeDescriptor>
          <TypeDescriptor TypeName="System.Single"
            IdentifierName="TaxFee"></TypeDescriptor>
          <TypeDescriptor TypeName="System.Double"
            IdentifierName="Quantity"></TypeDescriptor>
          <TypeDescriptor TypeName="System.Double"
            IdentifierName="PriceBase"></TypeDescriptor>
          <TypeDescriptor TypeName="System.Double"
            IdentifierName="PaymentDiscount"></TypeDescriptor>
          <TypeDescriptor TypeName="System.Double"
            IdentifierName="InvoiceLineTaxableAmount"></TypeDescriptor>
          <TypeDescriptor TypeName="System.String"
            IdentifierName="ItemDescription"></TypeDescriptor>
          <TypeDescriptor TypeName="System.Guid"
            IdentifierName="InvoiceId"></TypeDescriptor>
          <TypeDescriptor TypeName="System.Guid"
            IdentifierName="InvoiceItemId"></TypeDescriptor>
        </Parameter>
      </Parameters>
    </Method>
  </Methods>
</Entity>
```

Figura 6.8 - Exemplo de código relacionado com métodos para introdução de informação na base de dados.

Finalmente, como se pode verificar na

Figura 6.9, irá ser definido duas configurações extra com os nomes GetLastInvoiceInformation e GetInvoicesCounting, sendo que a primeira define os parâmetros necessários para obter os últimos documento emitidos, e a segunda define os parametrs necessários para obter a última contagem de documentos no sistema Primavera Express. A principal característica, e por isso referenciadas as duas, é a diferença no género de resultado retornado por ambas, isto é, a primeira retorna um objecto e a segunda retorna apenas um valor. O Objecto retornado contém os campos especificados na configuração, que correspondem a campos para identificação do documento.

```
<ExtraConfigurations>
   <ExtraConfiguration Name="GetLastInvoiceInformation" Query="SELECT Data, TipoDoc,
    NumDoc, Id FROM CabecDoc WHERE Data &gt;= {0} ORDER BY Data">
      <Parameters>
         <Parameter Direction="In" Name="InvoiceDateTime">
            <TypeDescriptor TypeName="System.DateTime"
              IdentifierName="InvoiceDateTime"></TypeDescriptor>
         </Parameter>
         <Parameter Direction="Return" Name="GenericInvoiceInformation">
            <TypeDescriptor TypeName="Object" IsCollection="true">
              <TypeDescriptors>
                 <TypeDescriptor TypeName="System.DateTime" Name="InvoiceDateTime"
                   TypeDBName="System.DateTime" DBName="Data"></TypeDescriptor>
                 <TypeDescriptor TypeName="System.String" Name="InvoiceType"
                   TypeDBName="System.String" DBName="TipoDoc"></TypeDescriptor>
                 <TypeDescriptor TypeName="System.Int32" Name="InvoiceNumber"
                   TypeDBName="System.Int32" DBName="NumDoc"></TypeDescriptor>
                 <TypeDescriptor TypeName="System.Guid" Name="InvoiceId"
                   TypeDBName="System.Guid" DBName="Id"></TypeDescriptor>
              </TypeDescriptors>
            </TypeDescriptor>
         </Parameter>
      </Parameters>
   </ExtraConfiguration>
```

```
    <ExtraConfiguration Name="GetInvoicesCounting" Query="SELECT COUNT(*) FROM
     CabecDoc">
       <Parameters>
          <Parameter Direction="Return" Name="InvoicesCount">
             <TypeDescriptor TypeName="Ordinal" IsCollection="false">
               <TypeDescriptors>
                  <TypeDescriptor TypeName="System.Int32" Name="InvoicesCount"
                    TypeDBName="" DBName=""></TypeDescriptor>
               </TypeDescriptors>
             </TypeDescriptor>
          </Parameter>
       </Parameters>
    </ExtraConfiguration>
 </ExtraConfigurations>
```

Figura 6.9 - Exemplo de código relacionado com configurações extra para obtenção dos últimos documentos emitidos na base de dados.

6.3. Exemplos de Funcionamento

Para iniciar a aplicação adaptador é necessário proceder à instalação dos quatro Serviços Windows. Para tal basta recorrer à consola do *Microsoft Visual Studio*, introduzindo a seguinte instrução, como pode ser visto na Figura 6.10.

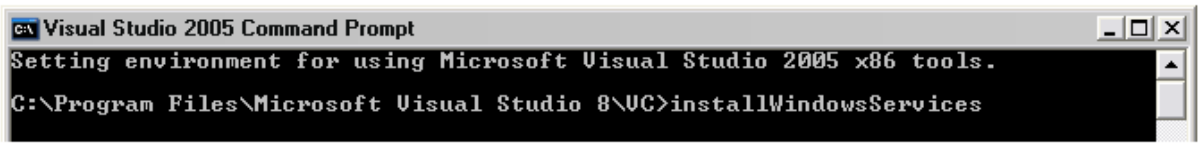

Figura 6.10 - Instrução para instalação dos Serviços Windows

Após os serviços se encontrem instalados, procede-se à sua inicialização através da ferramenta de administração correspondente aos serviços (Figura 6.11). A partir deste momento todo o processamento é efectuado automaticamente pelo adaptador, quer na recepção de mensagens, quer na detecção de novas facturas, seu processamento e posterior envio. Este facto leva a que não seja possível mostrar imagens do funcionamento da ferramenta, sendo desta forma utilizado esquemas e apenas algumas imagens do estado de algumas ferramentas utilizadas.

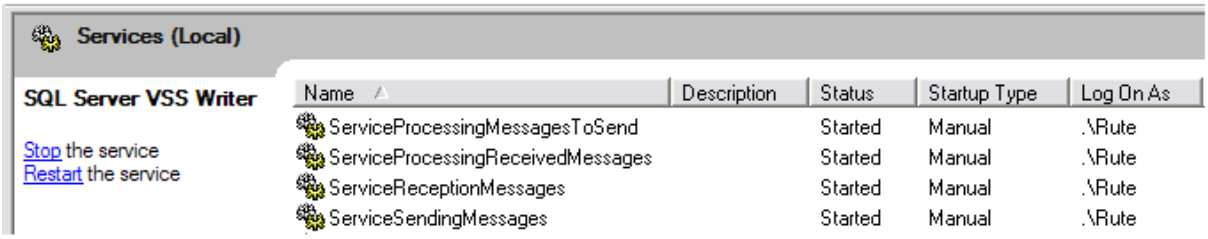

Figura 6.11 - Janela dos Serviços Windows já instalados no sistema

Primeiramente irá ser mostrado o funcionamento do adaptador aquando da detecção de uma nova factura, anteriormente referida na *Secção 6.1.2*, seu processamento e envio para o *Message Broker*. Posteriormente irá ser mostrado o funcionamento do adaptador aquando da chegada de uma nova factura ao sistema.

6.3.1. Detecção, Leitura, Processamento e Envio de uma nova Factura no Sistema.

A Figura 6.12 mostra o esquema de funcionamento dos Serviços Windows no envio de uma nova mensagem, correspondente a uma nova factura, para o *Message Broker*.

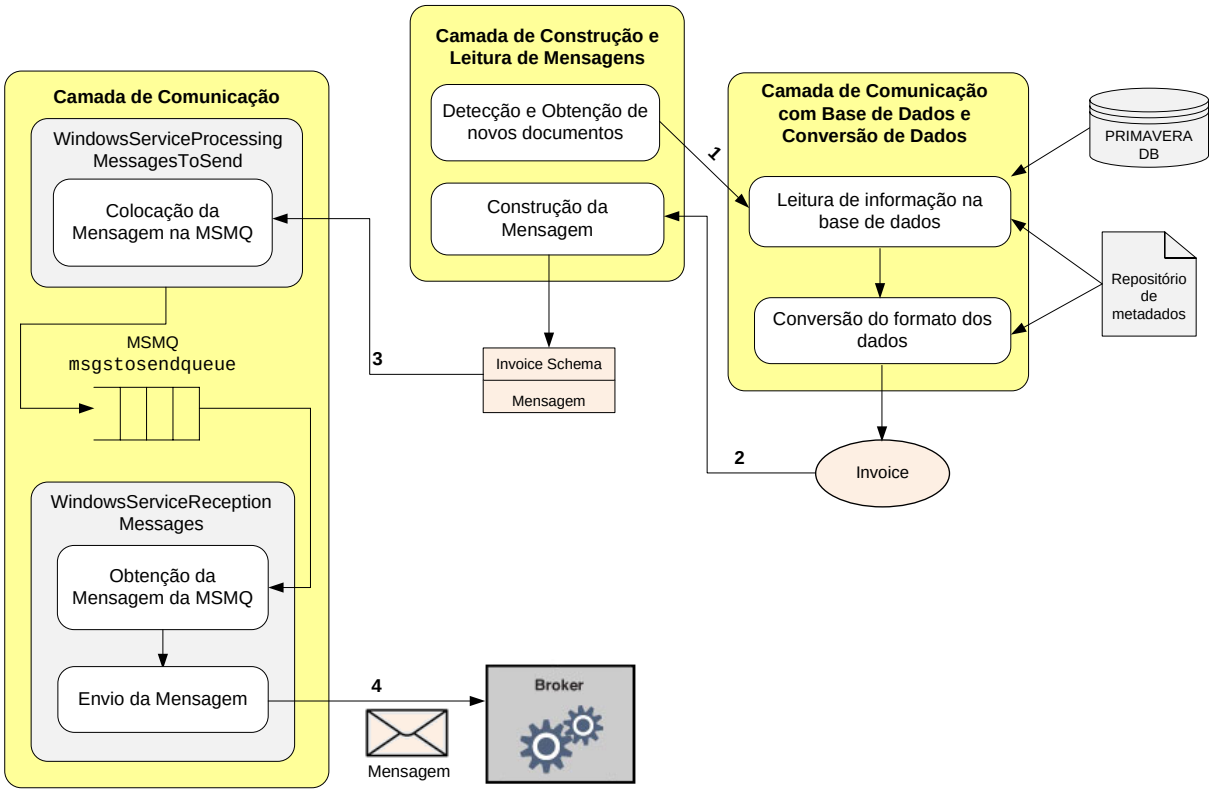

Figura 6.12 - Esquema de funcionamento dos Serviços Windows para envio de Mensagens

Como referido anteriormente, são utilizadas filas de espera de mensagens aquando da existência de mensagens por enviar. A Figura 6.13 mostra essas filas de espera com os nomes *msgtosendqueue* (para quando tudo corre bem na inserção da mensagem na fila de espera), *msgtosendqueue_deadletter* (quando ocorre um erro no processo de obtenção da mensagem na fila de espera pelo serviço responsável pelo envio) e *msgtosendqueue_error* (quando ocorre um erro no processo de inserção da mensagem na fila de espera pelo serviço responsável pelos eventos de processamento da mensagem).

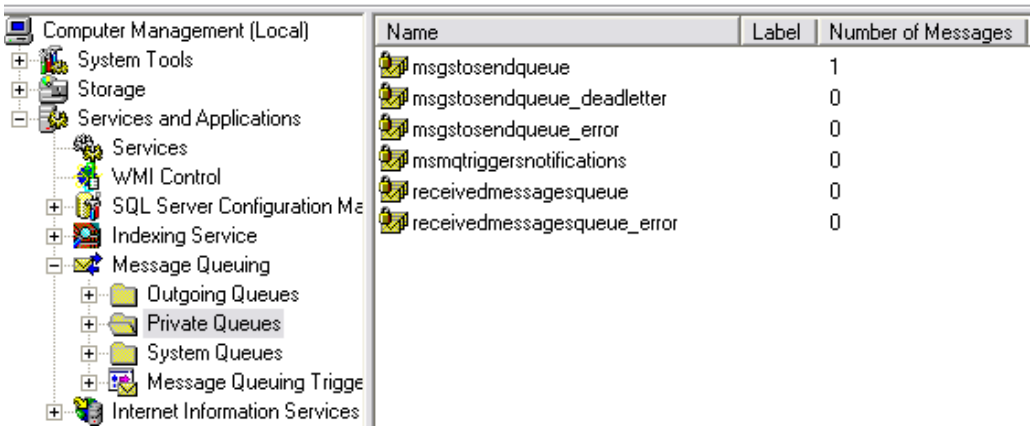

Figura 6.13 - Janela de gestão do computador relacionada com as filas de espera de mensagens – existência de mensagem a ser enviada

Foi implementado um mecanismo de *Log*, de nome *Log4Net [54]*, para obter retorno da execução dos Serviços Windows. Um exemplo do conteúdo dos *Logs* para os serviços de envio de mensagens pode ser verificado na Figura 6.14 e na Figura 6.15.

```
2007-09-14 02:21:52,385 [5] INFO
    IST.TFC.BrokerAdapter.WindowsServiceProcessingMessagesToSend.ServiceProcessingMessagesTo
    Send - Log initialized
2007-09-14 02:22:52,742 [5] INFO
    IST.TFC.BrokerAdapter.WindowsServiceProcessingMessagesToSend.ServiceProcessingMessagesTo
    Send - Number of new messages to send: 1
2007-09-14 02:22:57,299 [5] INFO
    IST.TFC.BrokerAdapter.WindowsServiceProcessingMessagesToSend.ServiceProcessingMessagesTo
    Send - Enviada mensagem para a MSMQ de envio para o cliente - Maria José da Silva
```

Figura 6.14 - Log do resultado de execução do *WindowsServicePrcessingMessagesToSend* e o resultado do processamento das restantes camadas

```
2007-09-14 02:22:06,956 [1] INFO
    IST.TFC.BrokerAdapter.WindowsServiceSendingMessages.ServiceSendingMessages - Log initialized
2007-09-14 02:23:10,117 [4] INFO
    IST.TFC.BrokerAdapter.WindowsServiceSendingMessages.ServiceSendingMessages - Obtida a
    mensagem da MSMQ para enviar para o Cliente - Maria José da Silva
```

```
2007-09-14 02:23:15,365 [4] INFO
    IST.TFC.BrokerAdapter.WindowsServiceSendingMessages.ServiceSendingMessages – Mensagem
    enviada com sucesso!
```

Figura 6.15 - Log do resultado de execução do *WindowsServiceSendingMessages*

6.3.2. Recepção, Processamento e Introdução de uma nova Factura no Sistema

A Figura 6.16 mostra o esquema de funcionamento dos Serviços Windows na recepção de uma nova mensagem correspondente a uma nova factura.

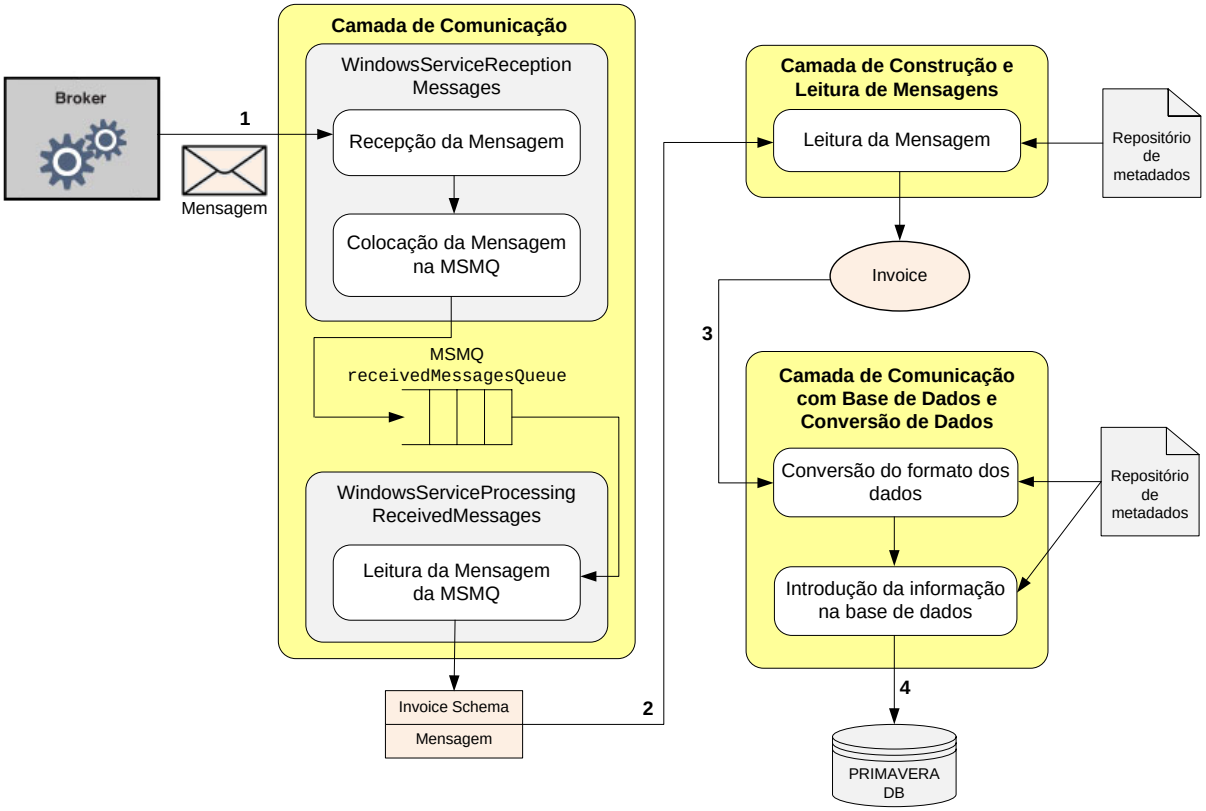

Figura 6.16 - Esquema de funcionamento dos Serviços Windows de Recepção de Mensagens

Também na execução do processo de recepção de mensagens são utilizadas filas de espera de mensagens. A Figura 6.17 mostra essas filas de espera respectivamente com os nomes *receivedmessagesqueue* (para quando tudo corre bem na inserção da mensagem na fila de espera) e *receivedmessagesqueue_error* (quando ocorre um erro no processo de inserção da mensagem na fila de espera pelo serviço responsável pelos eventos de processamento da mensagem).

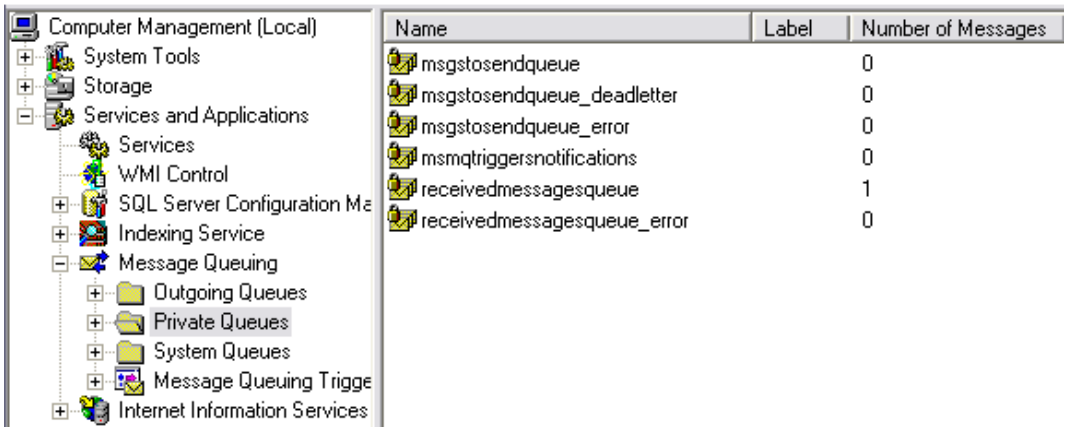

Figura 6.17 - Janela de gestão do computador relacionada com as filas de espera de mensagens – existência de mensagem recebidas

Um exemplo do conteúdo dos *Logs* para os serviços de recepção de mensagens pode ser verificado na Figura 6.18 e na Figura 6.19.

```
2007-09-13 18:56:16,469 [1] INFO
    IST.TFC.BrokerAdapter.WindowsServiceReceptionMessages.ServiceReceptionMessages – Log initialized.
2007-09-13 18:56:23,389 [4] INFO
    IST.TFC.BrokerAdapter.WindowsServiceReceptionMessages.ServiceReceptionMessages – Enviada mensagem
    d22f8293-b300-4864-8720-8b1c54ce14f3 para a MSMQ de recepção.
```

Figura 6.18 - *Log* do resultado de execução do *WindowsServiceReceptionMessages*

```
2007-09-14 01:04:34,777 [5] INFO
    IST.TFC.BrokerAdapter.WindowsServiceProcessingReceivedMessages.ServiceProcessingReceivedMessages - Log
    initialized
2007-09-14 01:05:37,988 [5] INFO
    IST.TFC.BrokerAdapter.WindowsServiceProcessingReceivedMessages.ServiceProcessingReceivedMessages -
    Obtida a mensagem da MSMQ recebida do Cliente - José Maria Fernandes & Filhos, Lda.
2007-09-14 01:05:38,228 [5] INFO
    IST.TFC.BrokerAdapter.WindowsServiceProcessingReceivedMessages.ServiceProcessingReceivedMessages - --->
    Invoice introduzido com sucesso na BD!
```

Figura 6.19 - Log do resultado de execução do *WindowsServicePrcessingReceivedMessages* e o resultado do processamento das restantes camadas

A seguir é mostrado o resultado da introdução de informação respeitante a uma mensagem recebida do *Message Broker*, situação de compra, na Figura 6.20, Figura 6.21 e Figura 6.22.

Filial	Serie	TipoDoc	NumDoc	Entidade	DataDoc	DataVencimento	CondPag	TotalMerc	TotalIva	Moeda
000	2007	FA	0	J.M.F.	2007-07-26 00:00:00.000	2007-07-26 00:00:00.000	3	200	0	EUR

DataCarga	DataDescarga	Id
13-02-2007	15-02-2007	7F68012C-0B74-44D3-AD23-FDBDC237EF35

Figura 6.20 - Valores na tabela CabecCompras introduzidos de acordo com os campos do cabeçalho no documento recebido

NumContribuinte	Nome	Morada
293829383333	José Maria Fernandes & Filhos, Lda.	43984, KINGS STREET

Figura 6.21 - Valores na tabela CabecCompras introduzidos de acordo com os campos do fornecedor no documento recebido

NumLinha	Artigo	Taxalva	Quantidade	PrecUnit	DescontoComercial	PrecoLiquido	Descricao
1	C0002	21	1	399,04	2	312,85	Scanner Ty2-474

Figura 6.22 - Valores na tabela LinhasCompras introduzidos de acordo com os campos dos itens no documento recebido

6.4. Resumo

Neste capítulo foi apresentado um cenário de desenvolvimento do repositório de metadados bem como de utilização do adaptador, e respectivo tipo de documento utilizado no âmbito da mediação electrónica (no caso deste trabalho, correspondente a uma factura electrónica).

Para melhor compreensão do repositório de metadados são mostrados excertos de código de como construir o ficheiro de configuração, de acordo com o documento apresentado anteriormente. Para tal, é mostrado os campos e valores presentes na base de dados para o documento em questão.

Finalmente são mostrados esquemas do funcionamento do adaptador em ambas as fases, quer na recepção quer no envio de um documento, bem como imagens do estado de algumas ferramentas utilizadas, com o intuito de mostrar o fluxo de processamento do adaptador implementado neste trabalho. Os resultados obtidos foram bastante satisfatórios, correspondendo àquilo que era pretendido para o trabalho.

7. Conclusão

Neste capítulo abordam-se dois temas, o trabalho futuro e uma conclusão geral. O trabalho futuro aborda o conjunto de alterações, evoluções e extensões que se poderão efectuar sobre o trabalho á realizado, com vista a melhorá-lo e expandir o seu âmbito.

7.1. Trabalho Futuro

Uma sugestão para trabalho a ser desenvolvido de forma a tirar maior partido deste adaptador passa pela implementação de código que proceda ao envio e recepção de mensagens de confirmação ou erro indicativas do resultado do processamento da mensagem no cliente e *Message Broker* e resultado da comunicação. Através da utilização das duas vias de comunicação com o *Message Broker*, estar sempre à espera de uma mensagem do tipo ACK ou ERROR, ou por outro lado, assim que recepcionada um documento envio de mensagem ACK ao *Message Broker* e, aquando do bom processamento da mensagem, envio de ACK para o parceiro.

Seria interessante uma abordagem mais avançada no que se refere à camada de comunicação com a base de dados, responsável pela leitura ou escrita de dados na base de dados e responsável pela conversão dos formatos dos dados. Relacionado com aspectos de comunicação com a base de dados, alteração dos métodos responsáveis pela conexão com a base de dados de forma a permitir outros servidores de bases de dados, para além do *Microsoft SQL Server*. Para tal teria de ser diferenciado o tipo de servidor de base de dados no repositório de metadados, e utilização de métodos e parâmetros de conexão de acordo com esse mesmo servidor. Esta camada é o ponto principal do adaptador, visto tornar possível a realização do objectivo principal de tornar possível a troca de mensagens entre sistemas com diferentes formatos dos dados.

Ao nível da facturação electrónica, proceder ao melhoramento de aspectos legais tais como à existência de uma entidade certificadora externa, responsável pela emissão e distribuição de certificados digitais, tornado desta forma a troca de certificados mais segura. A solução passaria pela implementação desta mesma entidade, incluindo segurança ao nível da troca de mensagens com os clientes quer ao nível da disponibilização de certificados digitais, para troca de mensagens entre clientes.

7.2. Considerações

Este trabalho descreveu uma arquitectura e uma metodologia para o desenvolvimento de um adaptador de integração para cenários de mediação electrónica.

Os objectivos delineados inicialmente foram atingidos, produzindo uma versão do adaptador que permite a integração de diferentes Sistemas ERP em cenários de mediação electrónica. O adaptador implementado recebe documentos electrónicos num formato normalizado através do Message

Broker, procede à sua leitura, processamento e conversão dos dados, e finalmente à sua introdução na base de dados do Sistema ERP. Todo o processamento relacionado com a mensagem é feito de acordo com o modelo relacional especificado no ficheiro de configuração (repositório de metadados). No sentido inverso, o adaptador implementado consegue proceder à leitura de informação na base de dados, também recorrendo ao ficheiro de configuração, posterior processamento e conversão dos dados, e finalmente à construção de um documento electrónico a ser enviado para o Message Broker.

Depois de uma análise efectuada ao estado-da-arte, e ao resultado obtido, conclui-se que este adaptador tem as seguintes vantagens:

- Motivador para integração entre Sistemas ERP díspares;
- Abstracção do funcionamento do adaptador para os utilizadores;
- Execução automática em Serviços Windows;
- Utilização de filas de espera de mensagens para armazenamento temporário de mensagens para envio ou recebidas, solucionando problema de elevada carga de gestão de mensagens;
- Comunicação segura, recorrendo ao protocolo *WS-Security* do WCF;
- Utilização de Assinaturas Digitais e Certificados Digitais para garantir autenticação e confidencialidade;

e as seguintes desvantagens:

- Desempenho consideravelmente baixo;
- Restrições ao nível de integração interna com o Sistema ERP, considerando apenas integração com bases de dados.

8. Referências

8.1. Documentos

[1] P. Marques, Troca de Informação de Negócio para Negócio – Do EDI ao XML/EDI e EBXML, Universidade Fernando Pessoa, 2003.

[2] D. Linthicum, B2B Application Integration – E-Business – Enable Your Enterprise, Addison-Wesley Information Technology Series, 2001.

[3] F. Cummins, Enterprise Integration – An architecture for enterprise application and systems integration, John Wiley & Sons, 2000.

[4] G. Samtani, M. Healey, S. Samtani, B2B Integration: A Practical Guide to Collaborative E-Commerce, Imperial College Press, 1992.

[5] S. Damodaran, "B2B integration over the Internet with XML: RosettaNet successes and challenges". Em Proceedings of 13th international World Wide Web Conference, pp. 188-195, 2004.

[6] I. Gorton e A. Liu, "Architectures and Technologies for Enterprise Application Integration". Em Proceedings of 26th International Conference on Software Engineering, 2004.

[7] M. M. Silva, Integração de Sistemas de Informação, FCA – Editora de Informática, 2003.

[8] Martin Hepp: Ontologies: State of the Art, Business Potential, and Grand Challenges, Hepp/De Leenheer/de Moor/Sure. (Eds.): Ontology Management: Semantic Web, Semantic Web Services, and Business Applications, ISBN 978-0-387-69899-1, pp. 3-22, Springer, 2007.

[9] S. Bechhofer, F. Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, L. Stein, "OWL Web Ontology Language", W3C Proposed Recommendation, 2003.

[10] K. Qureshi, "Enterprises Application Integration". Em Proceedings of International Conference on Emerging Technologies, pp. 340-345, 2005.

[11] H. M. Sneed, "Using XML to integrate existing software systems into the Web", Em Proceedings of Computer Software and Applications Conference, pp. 167-172, 2002.

[12] A. Vasconcelos, M.M. da Silva, A. Fernandes, J. Tribolet, An Information System Architectural Framework for Enterprise Application Integration. Em Proceedings of 37th Hawaii Conference on System Sciences, 2004.

[13] M. Chowdhury e M. Zafar, Integration of Legacy Systems in Software Architecture. Em Proceedings of Workshop at SIGSOFT, pp. 110-104, 2004.

[14] J. Lee, K. Siau., and S. Hong, Enterprise integration with ERP and EAI. Commun. of the ACM, Vol. 46, No. 2, pp. 54-60, 2003.

[15] P. Robertson, Integrating legacy systems with modern corporate applications. Commun. of the ACM, Vol. 40, N. 5, pp. 39-46, 1997.

[16] UMIC, *Guia da Factura Electrónica*, 2006.

[17] Anacom, *O Comércio Electrónico em Portugal – O Quadro Legal e o Negócio*, 2004.

[18] Aliança Digital, *Guia sobre a Factura Electrónica*, 2005

[19] Prológica, *Enquadramento Económico, Jurídico e Tecnológico*, 2006

[20] F. Silva e J. Alves, ERP e CRM: Da empresa à e-empresa - soluções de informação reais para

empresas globais, Edições Centro Atlântico, 2000.

[21] A. Albuquerque, M. Logato, R. Martins, Quebra de Paradigmas nas Transacções Comerciais B2B, MGR e-technologies, 2006

[22] M. Themistocleous, Z. Irani, R. O'Keefe e R. Paul, ERP Problems and Application Itegration Issues: An Empirical Survy. Em Proceedings of 34th Hawaii International Conference on System Sciences, 2001

[23] M Themistocleus, Z. Irani, P. Love, Enterprise Application Integration: An Emerging Technology for Integrating ERP and Supply Chains. Em Proceedings of European Conference on Information Systems, 2002.

[24] S. Dorda, K. Wallnau, R. Seacord, J. Robert, A Survey of Legacy System Modernization Approaches, CMU/SEI-2000-TN-003, 2000

[25] B. Medjahed, B. Benatallah, A. Bouguettaya, A. Ngu, A. Elmagarmid, Business-to-business interactions: issues and enabling technologies, *The VLDB Journal* 12, pp. 59-85, 2003

[26] C. Induruwana, Using An Aspect Layer in SOA for EAI, Pending Publication Workshop ICSOC, 2005

[27] Tibco Software Inc. Extending the benefits of SOA beyond the enterprise, 2006

[28] I. Leitão, Uma loja virtual emWindows Communication Foundation (Indigo), Tecnologias de Middleware, Pós-Graduação em Informática, 2006

8.2. Websites

[29] Resolução do Conselho de Ministros nº 137/2005, de 17 de Agosto de 2005
http://www.anacom.pt/template20.jsp?categoryId=212482&contentId=411608

[30] Decreto-Lei nº 256/2003, de 21 de Outubro de 2003
http://www.anacom.pt/template20.jsp?categoryId=100939&contentId=169711

[31] Resolução do Conselho de Ministros nº 137/2005, de 29 de Julho de 2005
http://www.anacom.pt/template20.jsp?categoryId=212482&contentId=411608

[32] Decreto-Lei nº 196/2007 de 15 de Maio de 2007
http://www.iapmei.pt/iapmei-leg-03.php?lei=5453

[33] Decreto-Lei nº 375/99, de 18 de Setembro de 1999, Relativo à Factura Electrónica
http://www.apdt.org/guia/L/Lfa/dl37599.htm

[34] Decreto Regulamentar nº 16/2000, de 2 de Outubro de 2000
http://www.apdt.org/guia/L/Lfa/dreg.htm

[35] GS1 Portugal – Factura Electrónica
http://www.gs1pt.org/produtos_solucoes/factura_electronica/factura_electronica.htm#p1

[36] UMIC – Factura Electrónica
http://www.umic.pt/index.php?option=com_content&task=view&id=34&Itemid=62

[37] PHC Software, *Factura Electrónica 2008 - Descritivo completo*, 2008
http://www.phc.pt/enews/DescricaoCompletaFacturaElectronica.pdf

[38] Primavera BSS
http://www.primaverabss.com/pt/PortalRender.aspx?PageID=e9c879ed-63c7-4483-8db8-cc6a47ab64be
http://www.primaverabss.com/pt/PortalRender.aspx?PageID={b011e1d2-82d3-4e9f-a372-

eabba4db3884}

[39] Primavera Express
http://www.primaverabss.com/pt/PortalRender.aspx?PageID={906727d5-b773-491d-b5e6-5d089089d110}

[40] Microsoft Corporation, *Business Data Catalog*, 2006
http://msdn2.microsoft.com/en-us/library/ms563661.aspx

[41] Microsoft Corporation, *Business Data Catalog: Architecture*, 2006
http://msdn2.microsoft.com/en-us/library/ms499729.aspx

[42] Thomas Rizzo, *Visão geral do Microsoft Office SharePoint Server 2007*, 2006
http://www.microsoft.com/technet/technetmag/issues/2007/01/SharePoint/default.aspx?loc=pt

[43] Microsoft Corporation, *Business Data Catalog: Overview*, 2006
http://msdn2.microsoft.com/en-us/library/ms551230.aspx

[44] Creating Business Data Catalog Entities in SharePoint Server 2007,
http://msdn2.microsoft.com/en-us/library/bb410048.aspx

[45] Microsoft Corporation, *Windows Communication Foundation Architecture Overview*, 2006
http://msdn2.microsoft.com/en-us/library/Aa480210.aspx

[46] Microsoft Corporation, *What Is Windows Communication Foundation?*, 2006
http://msdn2.microsoft.com/en-us/library/ms731082.aspx

[47] C. Peiris e D. Mulder, Hospedando e consumindo serviços WCF, 2007
http://www.microsoft.com/brasil/msdn/Tecnologias/netframework/Hosting_WCF_Services.mspx

[48] Keith Brown, *Security in Windows Communication Foundation*, 2006
http://msdn.microsoft.com/msdnmag/issues/06/08/securitybriefs/default.aspx
http://www.microsoft.com/brasil/msdn/tecnologias/seguranca/securitybriefs.mspx

[49] Fundamental Windows Communication Foundation Concepts
http://msdn2.microsoft.com/en-us/library/ms731079.aspx

[50] J. Olamendy, WCF Application implementing the Anonymous client over Certificate WS-Security scenario, C# Corner, 2006
http://www.c-sharpcorner.com/UploadFile/john_charles/WCFApplicationimplementingtheAnonymousclient_over11172006131249PM/WCFApplicationimplementingtheAnonymousclient_over.aspx

[51] Microsoft Corporation, *Windows Services*
http://msdn2.microsoft.com/en-us/library/d56de412(VS.80).aspx

[52] Microsoft Corporation, *Microsoft Message Queuing*, 2003
http://www.microsoft.com/windowsserver2003/technologies/msmq/default.mspx

[53] Microsoft Corporation, *ADO.NET*
http://msdn2.microsoft.com/en-us/library/h43ks021(VS.71).aspx

[54] Log4Net
http://logging.apache.org/log4net/

[55] iWay Universal Adapter Suite
http://www.iwaysoftware.com/products/index.html

[56] Attunity Connect
http://www.attunity.com/attunity_connect