Faculdade de Engenharia da Universidade do Porto

Plataforma para Suporte de Aplicações Ocasionalmente Conectadas em Sistemas ERP Baan

José Augusto Cerqueira da Silva

Dissertação submetida para satisfação parcial dos requisitos
do Grau de Mestre em Engenharia Informática

Dissertação realizada sob a orientação do
Professor Doutor Américo Lopes de Azevedo,
do Departamento de Engenharia Electrotécnica e de Computadores da
Faculdade de Engenharia da Universidade do Porto
e
co-orientação do
Professor Doutor António Miguel Pontes Pimenta Monteiro,
do Departamento de Engenharia Informática da
Faculdade de Engenharia da Universidade do Porto

Porto, Novembro de 2009

Resumo

A crescente tendência verificada na mobilidade da força de trabalho constitui um dos desafios actuais para as organizações e respectivos sistemas de informação, no sentido da obtenção de mais-valias competitivas. As soluções de mobilidade para sistemas *Enterprise Resource Planning (ERP)* surgem neste contexto como uma evolução natural, tornando os sistemas de informação empresariais mais abrangentes e capazes de responder às necessidades suscitadas por uma crescente tendência da mobilidade da força de trabalho.

Para que as aplicações externas ao sistema *ERP* possam executar funcionalidades deste é necessário que exista uma infra-estrutura que suporte esta integração.

Este trabalho visou precisamente a concepção da plataforma de suporte necessária à execução de funcionalidades existentes nos sistemas *ERP Baan*, por aplicações externas, privilegiando o modo ocasionalmente conectado. Este modo de funcionamento, particularmente útil às aplicações móveis, permite ultrapassar a natureza tipicamente síncrona das aplicações clientes dos sistemas *ERP*, permitindo ao utilizador móvel uma maior produtividade e eficiência na utilização das aplicações empresariais, ainda que sujeitas a algumas limitações impostas neste modo desconectado.

O trabalho de dissertação foi desenvolvido em três fases. Uma fase envolveu o estudo das principais características dos sistemas *ERP* e das aplicações ocasionalmente conectadas, no sentido de ajudar a determinar algumas das necessidades da plataforma a ser desenvolvida. Este estudo incluiu igualmente a análise de algumas tecnologias relevantes para a persecução dos objectivos, boas práticas e padrões de desenho. Numa segunda fase é determinado um conjunto de funcionalidades base que a plataforma de suporte deve disponibilizar e fundamentalmente é proposta a sua arquitectura. Finalmente, a terceira fase diz respeito ao desenvolvimento e implementação dos diversos componentes de *software* necessários à concretização da plataforma, demonstrando o seu bom funcionamento.

O contributo deste trabalho permitiu a concepção e implementação de uma arquitectura capaz de facilitar a interacção de aplicações externas com sistemas *ERP* Baan, privilegiando o modo ocasionalmente conectado, embora não fique limitado unicamente a este.

Abstract

Nowadays, enterprise information systems are facing major challenges caused by an increasing trend of enterprise mobile workers. In this context, Enterprise Resource Planning (ERP) mobile solutions are a natural evolution towards dynamic and efficient information systems capable of satisfying the needs of those workers.

The work of this dissertation consists precisely on the conception of the necessary infra-structure to support the execution of Baan's ERP session's logic by external applications, mainly occasionally connected applications. The occasionally connected type of interaction is especially useful for mobile applications, allowing mobile workers to be more productive and efficient in the use of enterprise applications.

This dissertation work was developed in three phases. One phase involves the study of ERP systems and the occasionally connected application's major characteristics. The goal of this study was to determine some of the platform's requirements. This study also encompasses relevant technologies to objectives fulfillment, best practices, and design patterns. The second phase consists on the platform's requirements elaboration and essentially on its architecture proposal. Finally, the third phase involves the conception, development and test of the platform's software components.

The contribution of this work resulted in the creation of a platform that allows external applications to interact with Baan's ERP session's logic in an occasionally connected scenario.

Índice

Resumo ..... **iii**

Abstract ..... **v**

Índice ..... **vii**

Lista de figuras ..... **xi**

Lista de tabelas ..... **xiii**

Abreviaturas ..... **xv**

Capítulo 1 - Introdução **1**

1.1 Enquadramento 3

1.2 Motivações 4

1.3 Objectivos Principais 4

1.4 Metodologia 5

1.5 Estrutura da Dissertação 6

Capítulo 2 - Sistemas ERP **9**

2.1 Enterprise Resource Planning - ERP 11

2.1.1 Definição e Principais Conceitos 11

2.1.2 Perspectiva Histórica e Principais Fornecedores 15

2.2 O Sistema ERP Baan 17

2.2.1 Introdução 17

2.2.2 Arquitectura do Sistema 21

2.2.3 O software Baan como Servidor de Automação OLE 25

2.2.4 Execução das sessões via Application Function Server - AFS 31

2.2.5 Suporte ao Processamento de XML 40

Capítulo 3 - Aplicações Ocasionalmente Conectadas **47**

3.1 Conceitos e Cenários de Utilização 49

3.1.1 Aplicação Ocasionalmente Conectada 49

3.1.2 Cenários Comuns de Utilização 50

3.2 Desenho da Arquitectura 51

3.2.1 Abordagem Centrada nos Dados (Data-Centric) 52

3.2.2 Abordagem Orientada aos Serviços (Service-Oriented) 54

3.3 Suporte da Smart Client Software Factory - SCSF 56
3.3.1 O Bloco Aplicacional Connection Monitor 57
3.3.2 O Bloco Aplicacional Endpoint Catalog 64
3.3.3 O Bloco Aplicacional Disconnected Service Agent 68

Capítulo 4 - Requisitos 75
4.1 O Requisito – definição, processo e especificação 77
4.2 Âmbito 79
4.2.1 Descrição 79
4.2.2 Objectivos 80
4.2.3 Benefícios Esperados 80
4.3 Descrição Geral 81
4.3.1 Perspectiva do produto / Arquitectura 81
4.3.2 Interfaces 87
4.3.3 Funcionalidades do produto 89
4.3.4 Características dos utilizadores 89
4.3.5 Restrições 90
4.3.6 Pressupostos e dependências 90
4.4 Requisitos Específicos 91
4.4.1 Requisitos Funcionais 91
4.4.2 Requisitos Não Funcionais 97

Capítulo 5 - Implementação 99
5.1 Metodologia 101
5.2 Plataforma de Implementação 103
5.3 O Integration Management System – IMS 104
5.3.1 A DLL amimsdll00001 105
5.3.2 Execução e Controlo 108
5.3.3 Sessões Desenvolvidas 114
5.3.4 Diagrama de Instalação 118
5.4 Camada de Acesso ao IMS 119
5.4.1 Os Componentes OleAutomationServer e Baan Configuration 121
5.4.2 O subsistema Controlo e Configuração do Acesso 122
5.4.3 O subsistema Fachada do IMS 126
5.4.4 Diagrama de Instalação 129
5.5 Classe facilitadora e respectiva decoração com atributos 130
5.5.1 Objectivo 130
5.5.2 Classes de atributos criadas 130

Capítulo 6 - Resultados Obtidos **133**
6.1 Metodologia 135
6.2 Exemplo de utilização da plataforma 136
6.3 Revisão da documentação de Requisitos 144
6.4 Testes de Sistema 145
6.4.1 Testes funcionais do subsistema IMS 145
6.4.2 Testes funcionais da Camada de Acesso ao IMS 148
6.4.3 Testes de Carga 151
6.5 Resumo e Análise dos Resultados Obtidos 153

Capítulo 7 ..... **157**
7.1 Conclusões 159
7.2 Trabalho Futuro 162

Referências e Bibliografia **163**

Anexos **171**

Anexo A - Subsistema Integration Management System - IMS 173
A.1 – Código da DLL amimsdll00001 175
A.2 – Código do programa amimslaunmon 181
A.3 – Código do programa amimsprocreq 183

Anexo B - Subsistema Camada de Acesso ao IMS 189
B.1 – Código do susbsistema Controlo e Configuração do Acesso 191
B.2 – Código do susbsistema Fachada do IMS 197

Anexo C - Código Utilizado nos Testes de Sistema 207
C.1 – Código do BO para inserir Ordem de Venda amimsboteste 209
C.2 – Código da classe facilitadora OrdemVendaBaan 211
C.3 – Código do BO de Teste da Execução amimsexecteste 212
C.4 – Código da classe facilitadora BBOteste 213
C.5 – Código 3GL utilizado no teste da DLL amimsdll00001 214
C.6 – Código do Exemplo de Utilização da Plataforma 217

Anexo D - Especificações de Casos de Teste 219
D.1 – Casos de Teste Funcionais da DLL amimsdll00001 221
D.2 – Casos de Teste Funcionais das Sessões do módulo IMS 233

Lista de figuras

Figura 1.1 - Estrutura da Dissertação 8
Figura 2.1 – Conceito de sistema ERP 11
Figura 2.2 – Conceito do ERP II 13
Figura 2.3 – Sistemas Empresariais 14
Figura 2.4 – Evolução do ERP 15
Figura 2.5 - Evolução das versões do sistema ERP Baan 18
Figura 2.6 – Estrutura simplificada do sistema ERP Baan 21
Figura 2.7 - Estrutura do sistema ERP Baan 22
Figura 2.8 – Cenário cliente/servidor duas camadas 23
Figura 2.9 – Cenário cliente/servidor três camadas 24
Figura 2.10 – Ecrã de configuração do acesso ao sistema *ERP* Baan 29
Figura 2.11 – Arquitectura do *Application Function Server* 31
Figura 2.12 – Novo registo na sessão de ordens de venda 36
Figura 2.13 – Campos da subsessão de ordens de venda 36
Figura 2.14 – Campo desconto de ordem na subsessão ordens de venda 37
Figura 2.15 – Campo condições de pagamento na subsessão ordens de venda 37
Figura 2.16 – Opções guardar e fechar na subsessão ordens de venda 38
Figura 2.17 – Campo número de ordem na sessão ordens de venda 38
Figura 2.18 – Arquitectura da funcionalidade do Objecto XML 40
Figura 2.19 – Estrutura do objecto XML 42
Figura 2.20 – Algoritmo *Depth-First Search* (DFS) 44
Figura 2.21 – Algoritmo *Breadth-First Search* (BFS) 44
Figura 3.1 – Abordagem *Service-Oriented vs.* Abordagem *Data Centric* 51
Figura 3.2 – Estrutura do *Connection Monitor Application Block* 58
Figura 3.3 – Estrutura do *Endpoint Catalog Application Block* 65
Figura 3.4 – Estrutura do *Disconnected Service Agent Application Block* 69
Figura 3.5 – Estrutura do subsistema *RequestManager* 71
Figura 3.6 – Estrutura da classe *Request* 72
Figura 3.7 – Estrutura da classe *ConnectionMonitorAdapter* 73
Figura 4.1 – Cenário típico de uma solução de mobilidade 79

Figura 4.2 – Principais blocos da arquitectura proposta 81
Figura 4.3 – Principais blocos do *Integration Management System - IMS* 82
Figura 4.4 – Principais componentes da Camada de Acesso ao *IMS*.....84
Figura 4.5 – Esquema para o *XML* do pedido de integração.....87
Figura 4.6 – Esquema para o *XML* do resultado de processamento do pedido.....87
Figura 5.1 – Principais blocos do *Integration Management System - IMS* 104
Figura 5.2 – Arquitectura do *Integration Management System - IMS* 104
Figura 5.3 – Diagrama de Estados do Pedido de Integração.....109
Figura 5.4 – Visualização de processos com o utilitário *ottgbfprocess* 110
Figura 5.5 – Informação disponibilizada pelo comando ps 110
Figura 5.6 – Ecrã da sessão de Pedidos de Integração 114
Figura 5.7 - Ecrã da sessão de Detalhe do Pedido de Integração.....115
Figura 5.8 – Ecrã da sessão Criar Pedidos de Integração.....115
Figura 5.9 – Ecrã da sessão de Processar Pedidos de Integração.....116
Figura 5.10 – Ecrã da sessão Parâmetros *IMS* 116
Figura 5.11 – Ecrã da sessão de Histórico dos Pedido de Integração 117
Figura 5.12 – Diagrama de instalação do *Integration Management System - IMS*.....118
Figura 5.13 – Principais componentes da Camada de Acesso ao *IMS*.....119
Figura 5.14 – Subsistema de Controlo e Configuração do Acesso 122
Figura 5.15 – Subsistema Fachada do *IMS* 126
Figura 5.16 – Diagrama de Instalação da Camada de Acesso ao *IMS* 129
Figura 6.1 – Ecrã da aplicação de utilização do *Business Object* 139
Figura 6.2 – Mensagem com o número de ordem de venda criado 141
Figura 6.3 – Sessão Ordens de Venda do sistema *ERP* Baan: Ecrã Geral.....142
Figura 6.4 – Sessão Ordens de Venda do sistema *ERP* Baan: Ecrã Vender.....142
Figura 6.5 - Ecrã da aplicação preenchido com parceiro incorrecto.....143
Figura 6.6 – Mensagem de erro devolvida pela execução do *BO* 143
Figura 6.7 – Testes realizados à classe BaanIMS e respectivos resultados 149
Figura 6.8 – Teste de carga TC1 151
Figura 6.9 – Número de testes/segundo no teste de carga TC1 151
Figura 6.10 – Carga de utilização no teste de carga TC1.....151
Figura 6.11 – Teste de carga TC2 152
Figura 6.12 – Número de testes/segundo no teste carga TC2 152

Lista de tabelas

Tabela 2.1 - Pacotes do sistema Infor ERP LN 6.1 20
Tabela 2.2 – Métodos do Objecto OleAutomationServer..... 27
Tabela 2.3 – Propriedades do Objecto OleAutomationServer..... 27
Tabela 2.4 – Métodos da classe BWConfigurationClass..... 29
Tabela 2.5 – Propriedades da classe BWConfigurationClass..... 30
Tabela 2.6 – Explicação do exemplo de utilização do *AFS*..... 35
Tabela 5.1 – Funções da DLL amimsdll00001..... 106
Tabela 5.2 – Funções auxiliares da DLL amimsdll00001 106
Tabela 6.1 – Lista para verificação dos requisitos..... 144
Tabela 6.2 – Especificação do teste de sistema TS1..... 146
Tabela 6.3 – Especificação do teste de sistema TS6..... 147
Tabela 6.4 – Resultados dos testes funcionais do módulo *IMS*..... 148
Tabela 6.5 – Resultados dos testes funcionais do subsistema Fachada do *IMS* 150
Tabela 6.6 – Resumo de resultados dos testes funcionais de sistema..... 153

Abreviaturas

3GL	*Third-Generation Programming Language*
4GL	*Fourth-Generation Programming Language*
AFS	*Application Function Server*
API	*Application Programming Interface*
ATP	*Available-to-Promise*
B2B	*Business-to-Business*
B2C	*Business-to-Consumer*
BECS	*Baan Environment and Configuration Selector*
BFS	*Breadth-First Search*
BO	*Business Object*
BOI	*Business Object Interface*
COM	*Component Object Model*
CRP	*Capacity Requirements Planning*
CRM	*Customer Relationship Management*
CTP	*Capable-to-Promise*
DFS	*Depth-First Search*
DLL	*Dynamic Link Library*
DTD	*Document Type Definition*
DRP	*Distribution Requirements Planning*
ERP	*Enterprise Resource Planning*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
IMS	*Integration Management System*
MPS	*Master Production Scheduling*
MRP	*Material Resource Planning*
OLE	*Object Linking and Embedding*
REXEC	*Remote Execution Protocol*
SCM	*Supply Chain Management*
SCSF	*Smart Client Software Factory*

SGBDR	Sistema Gestor de Base de Dados Relacional
SOAP	*Simple Object Access Protocol*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
TLS	*Transport Layer Security*
TQM	*Total Quality Management*
VM	*Virtual Machine*
XML	*EXtensible Markup Language*
XSD	*XML Schema Definition*
WBS	*Work Breakdown Structure*

Capítulo 1

Introdução

1.1 Enquadramento

Assistimos hoje a um mundo cada vez mais competitivo e exigente, onde as aplicações disponibilizadas em dispositivos móveis desempenham um papel importante, face à crescente tendência da mobilidade da força de trabalho das organizações.

Os sistemas *Enterprise Resource Planning – ERP*, ou Planeamento de Recursos Empresariais, representam uma das fases mais marcantes na área dos sistemas de informação dos últimos anos. Têm origem nos sistemas *MRP (Material Resource Planning)*, que lidavam com as necessidades de informação relacionadas com o cálculo das necessidades de materiais para o processo produtivo. Porém, com o objectivo de ampliar a abrangência dos produtos, os fornecedores destes sistemas foram desenvolvendo cada vez mais módulos integrados, cujo âmbito cedo ultrapassou os limites da produção, tornando-se actualmente em sistemas transversais a toda a organização e orientados para os processos de negócio.

No contexto actual de competitividade e globalização, os sistemas de informação empresariais em geral, evoluem no sentido de se tornarem cada vez mais abrangentes e capazes de responder às necessidades suscitadas por uma crescente tendência da mobilidade da força de trabalho.

Num cenário de mobilidade da força de trabalho, as aplicações ocasionalmente conectadas surgem como sendo aquelas que oferecem maior flexibilidade ao utilizador móvel, uma maior produtividade e eficiência na utilização das aplicações empresariais. A razão prende-se com o facto de fornecerem um conjunto de funcionalidades que permitem aos utilizadores a interacção com a aplicação, mesmo quando os serviços de rede não estão disponíveis, ou não respondem num tempo adequado.

Surge então o problema - como possibilitar a execução das funcionalidades existentes nas aplicações de um sistema *ERP* a uma aplicação externa ocasionalmente conectada.

Esta questão é o mote para o trabalho a ser desenvolvido no âmbito desta dissertação levando em conta um conjunto de restrições, como por exemplo, não violar nem duplicar as regras de negócio existentes no sistema *ERP* e garantir a consistência e controlo de toda a informação.

1.2 Motivações

A escolha do tema desta dissertação está associada ao plano profissional do autor, sendo colaborador na área informática de uma organização que possui como sistema *ERP* o Baan 5.0c.

Com a concepção do protótipo pretende-se validar não só a arquitectura proposta para o sistema, mas também abrir caminho ao desenvolvimento de um conjunto de aplicações que beneficiem da infra-estrutura criada. Entre esse conjunto estão as aplicações ocasionalmente conectadas, que associadas ao acesso e a interacção móveis poderão auxiliar diversos sectores da organização.

O sector da assistência técnica pós-venda representa uma dessas áreas e poderá beneficiar de uma maior rapidez na actualização dos dados no sistema *ERP*, diminuição de recursos administrativos, melhor qualidade dos dados introduzidos, planeamento mais eficiente das ordens de serviço e aproveitamento das capacidades dos dispositivos móveis actuais que englobam câmaras digitais e informação sobre o posicionamento geográfico.

A área comercial constitui igualmente um dos cenários típicos de aplicabilidade. Neste caso, para além das vantagens intrínsecas ao uso deste tipo de aplicações, possui um benefício intangível que se concretiza na imagem de modernidade transmitida aos diversos parceiros de negócio.

1.3 Objectivos Principais

Tendo em vista a resolução do problema inicialmente formulado no enquadramento, foram definidos os seguintes objectivos principais, a serem atingidos no fim deste estudo:

- Identificação do conjunto de funcionalidades base, a serem disponibilizadas pela plataforma;
- Conceber uma arquitectura baseada em boas práticas e padrões de desenho;
- Desenvolver e implementar um protótipo da solução, demonstrando o seu bom funcionamento.

1.4 Metodologia

Tendo em conta o tema e os objectivos propostos, o trabalho desta dissertação foi dividido em três fases principais que se complementam.

A primeira teve por objectivo efectuar o estudo dos sistemas envolvidos, de forma direccionada aos aspectos mais relevantes para a persecução dos objectivos. Neste sentido, sobre o sistema *ERP* Baan foi estudada a sua arquitectura, o acesso à lógica das aplicações de forma programática, a automação do acesso e o processamento de *XML*. Apesar do seu carácter mais geral, foram também estudados os conceitos e características principais deste tipo de sistemas. Sobre as aplicações ocasionalmente conectadas, foi realizada uma breve análise de conceitos e arquitecturas possíveis, dirigindo-se o maior esforço para a análise do suporte ao modo ocasionalmente conectado disponibilizado pela *Smart Client Software Factory - SCSF*. Esta última, consiste num conjunto de práticas testadas e aceites para a resolução de problemas recorrentes, permitindo melhorar a qualidade e reduzir o risco no desenvolvimento de *software*.

Na segunda fase é elaborado um conjunto de funcionalidades base que a plataforma de suporte deve disponibilizar e proposta a sua arquitectura. Nesta fase foi analisada a norma IEEE Std 830-1998 *Recommended Practice for Software Requirements Specifications* e utilizada parte da sua estrutura na especificação dos requisitos.

A terceira fase diz respeito ao desenvolvimento, implementação e teste dos diversos componentes de software necessários à concretização da plataforma. Para uma melhor gestão, a plataforma foi segmentada em diversos subsistemas e utilizada uma abordagem iterativa e incremental no seu desenvolvimento.

Para validação do trabalho desenvolvido foram realizados testes de sistema funcionais e de carga, permitindo uma apresentação quantificável e objectiva dos resultados obtidos. Nesta validação foi adoptada uma lista de atributos dirigida à verificação dos requisitos e analisada a norma IEEE Std 829-1998 *Standard for Software Test Documentation* com o objectivo de tornar a especificação dos casos de teste realizados mais eficaz e eficiente.

1.5 Estrutura da Dissertação

Neste primeiro capítulo, é contextualizado o problema que o trabalho a ser desenvolvido por esta dissertação pretende resolver. São definidos os objectivos principais a serem atingidos e referidas as metodologias utilizadas.

O segundo capítulo está dividido em duas partes principais. Na primeira, é feita uma introdução ao tema *Enterprise Resource Planning – ERP*, abordando os aspectos e conceitos principais subjacentes aos sistemas que o suportam. A segunda parte está mais direccionada ao âmbito desta dissertação e diz respeito ao sistema concreto *ERP* Baan. Nesta parte é descrita a arquitectura e principalmente abordadas as tecnologias deste sistema *ERP*, utilizadas no âmbito do desenvolvimento da plataforma que se pretendeu conceber e implementar.

O terceiro capítulo tem dois objectivos fundamentais. O primeiro diz respeito a fornecer uma breve perspectiva sobre as aplicações ocasionalmente conectadas. O segundo relaciona-se com o estudo de boas práticas e padrões a utilizar no suporte ao modo ocasionalmente conectado. Assim, numa primeira parte, são abordados os conceitos e cenários de utilização das aplicações ocasionalmente conectadas e descritas as principais estratégias relativas ao desenho da sua arquitectura. A segunda parte visa efectuar uma análise mais detalhada do suporte ao modo ocasionalmente conectado fornecido por alguns dos blocos aplicacionais existentes na *Smart Client Software Factory – SCSF*.

O quarto capítulo diz respeito aos diversos aspectos relacionados com os requisitos do sistema que se pretende desenvolver e está dividido em três partes principais. Na primeira parte, são efectuadas breves considerações sobre conceitos associados ao próprio requisito. A segunda parte, visa descrever o âmbito, objectivos principais e realizar uma descrição geral do sistema pretendido, com particular ênfase na sua arquitectura. Na terceira parte, é criada uma lista de requisitos específicos funcionais e não funcionais, devidamente identificados e agrupados, que servirão para orientar as fases seguintes de implementação e verificação dos resultados.

O quinto capítulo descreve os aspectos mais relevantes da implementação dos requisitos documentados no capítulo anterior e está dividido três partes. Uma inicial, onde são feitas breves considerações sobre a metodologia e plataforma de implementação adoptadas. Na segunda parte é apresentada a implementação realizada

no servidor *ERP* Baan e detalhado o desenvolvimento do módulo *Integration Management System – IMS*, que constitui o subsistema principal da plataforma. Na terceira parte é descrita a implementação do subsistema Camada de Acesso ao *IMS*, desenvolvido para o servidor de mobilidade.

O sexto capítulo visa descrever as validações efectuadas sobre o trabalho realizado, com o objectivo de verificar a sua adequação ao uso e a sua conformidade com os requisitos funcionais e não funcionais especificados no quarto capítulo. Após a definição da abordagem relativa aos diversos testes a serem realizados, são apresentadas três partes principais que compõem este capítulo. Na primeira parte, é apresentado um exemplo elucidativo da utilização da plataforma que constitui o primeiro teste do sistema como um todo. Na segunda parte é verificada a própria documentação de requisitos, utilizando uma lista de verificação produzida para o efeito, seguindo-se a especificação e resultados da execução dos testes de sistema: funcionais e de carga. Na terceira parte, é apresentado o resumo e análise dos resultados obtidos.

Finamente, no sétimo capítulo são produzidas considerações sobre o trabalho realizado e sobre os resultados obtidos, apresentando-se algumas orientações relativas ao trabalho a desenvolver no futuro.

A figura 1.1 representa a estrutura adoptada para esta dissertação.

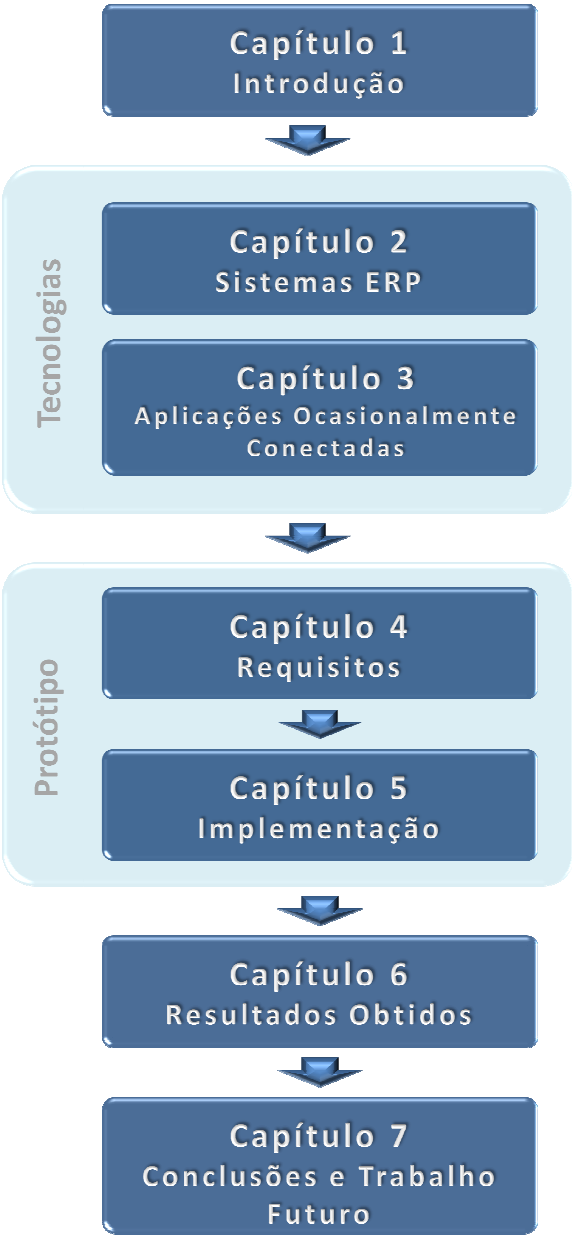


Figura 1.1 - Estrutura da Dissertação

Capítulo 2

Sistemas ERP

Este capítulo pretende, numa primeira parte, realizar a introdução ao tema *Enterprise Resource Planning – ERP*, abordando os aspectos e conceitos principais subjacentes aos sistemas que o suportam. É também feita uma breve perspectiva histórica e indicados os principais fornecedores a nível mundial de soluções *ERP*.

A segunda parte está mais direccionada ao âmbito desta dissertação e diz respeito ao sistema concreto *ERP* Baan. Nesta parte é descrita a arquitectura e principalmente abordadas as tecnologias deste sistema *ERP*, utilizadas no âmbito do desenvolvimento da plataforma que se pretendeu conceber e implementar.

2.1 Enterprise Resource Planning - ERP

2.1.1 Definição e Principais Conceitos

Os sistemas *Enterprise Resource Planning – ERP* ou Planeamento de Recursos Empresariais, representam uma das fases mais marcantes na área dos sistemas de informação nos últimos anos.

O *ERP* é um termo genérico que pretende identificar o conjunto de actividades executadas por um pacote de software modular, tendo por objectivo primário auxiliar os diversos processos de gestão de uma empresa [Adam, 2004].

Um sistema *ERP* constitui um software empresarial que tem por objectivo permitir a uma organização gerir os seus recursos de forma eficaz e eficiente, disseminar práticas comuns através da integração e automação dos processos empresariais, facilitar a partilha e acesso à informação em tempo real, devido à sua arquitectura modular e centrada num repositório de dados comum [Hossain, 2002 e Adam, 2004].

Os produtos *ERP* caracterizam-se por um conjunto de subsistemas integrados, capazes de satisfazer as necessidades de informação e automatizar os diversos processos empresariais, como por exemplo, a entrada do pedido de um cliente e todas as operações conducentes à sua expedição, incluindo o planeamento dos recursos financeiros, materiais e humanos para a sua produção. Estes produtos pretendem essencialmente eliminar a redundância de operações e burocracia, recorrendo à automatização e integração dos processos empresariais.

O conceito de sistema *ERP* pode ser representado através do esquema da figura 2.1.

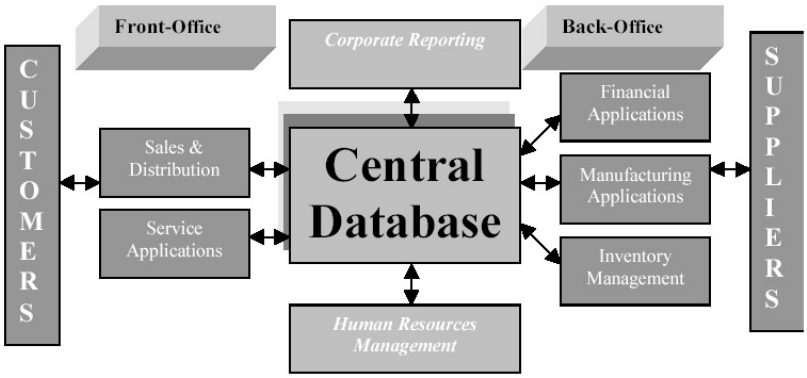


Figura 2.1 – Conceito de sistema ERP

Fonte: ERP- Global Opportunities & Challenges, Hossain L., 2002

Os módulos que compõem o *ERP* possibilitam, em tempo real, desenvolver e gerir o negócio de forma integrada. As informações são mais consistentes, dado existir uma partilha da informação e práticas comuns por todas as áreas funcionais da organização.

Resumindo, algumas das principais características dos sistemas ERP são:

- Concepção modular orientada às diversas áreas e funções de negócio;
- Alto nível de complexidade e configuração dos módulos;
- Arquitectura independente de plataformas, baseada numa filosofia cliente/servidor;
- Repositório de dados comum e centralizado;
- Disponibilidade da informação em tempo real;
- Elevados custos e duração dos processos de implementação;
- Partilha de informação e práticas comuns por toda a organização;
- Orientado para o processo de negócio e transversal a toda a organização.

O ERPII ou XRP (eXtended ERP)

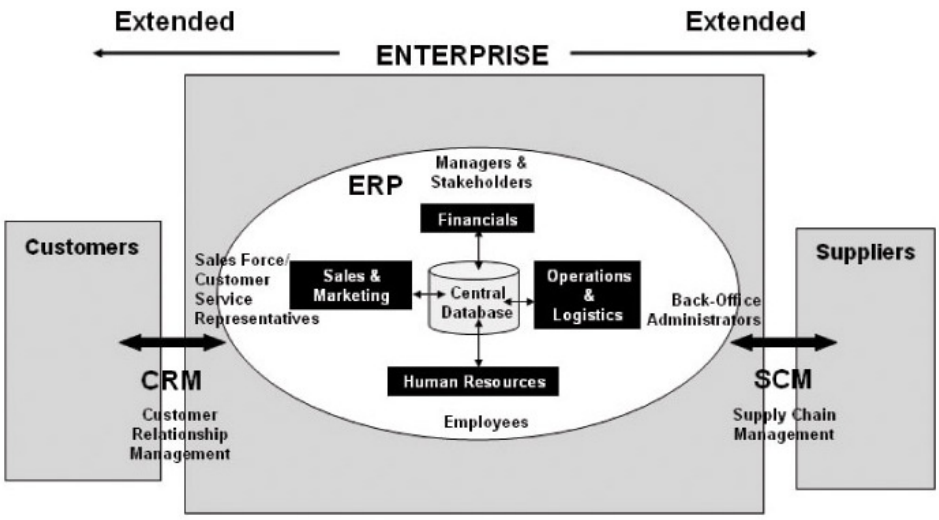


Figura 2.2 – Conceito do ERP II

Fonte: The Enterprise Resource Planning Decade, Adam, F., 2004

O conceito do *ERPII* ou *XRP (eXtended ERP)*, mostrado esquematicamente na figura 2.2, surge como uma evolução natural dos sistemas *ERP* dada a crescente inclusão de módulos adicionais e necessidade de uma arquitectura mais centrada na *Web* que permitisse dar resposta a cenários de *Business-to-Business - B2B* e *Business-to-Consumer - B2C.* São exemplos os módulos: *Customer Relationship Management - CRM* na gestão do relacionamento com os clientes; *Supply Chain Management – SCM* na gestão do relacionamento com os vários parceiros envolvidos, tendo em vista o planeamento, fornecimento, produção e entrega de produtos.

No âmbito desta evolução, os fornecedores de soluções *ERP* desenvolveram também *middleware* e ferramentas assentes em *XML* [W3C, 2009a] e *Web Services* [W3C, 2009d], facilitando a integração e interoperabilidade com outros sistemas [Laudon, 2006].

As funcionalidades de um sistema *ERP* representam uma solução genérica e reflectem uma série de considerações sobre a forma como as empresas trabalham na generalidade. Para flexibilizar a sua utilização num maior número de empresas e sectores de actividade, os sistemas *ERP* foram desenvolvidos para que a solução genérica possa ser configurada até um determinado nível, viabilizando desta forma, o cumprimento das necessidades específicas das organizações.

A configuração dos processos de negócio é um compromisso entre os requisitos da empresa e as funcionalidades disponibilizadas pelo sistema *ERP*. Na maioria das vezes, os processos de negócio carecem de uma redefinição para que os seus requisitos se aproximem das funcionalidades padronizadas do sistema, dado que reflectem boas práticas de gestão.

O diagrama que se pode observar na figura 2.3 representa precisamente essa orientação ao processo de negócio transversal às diversas áreas funcionais.

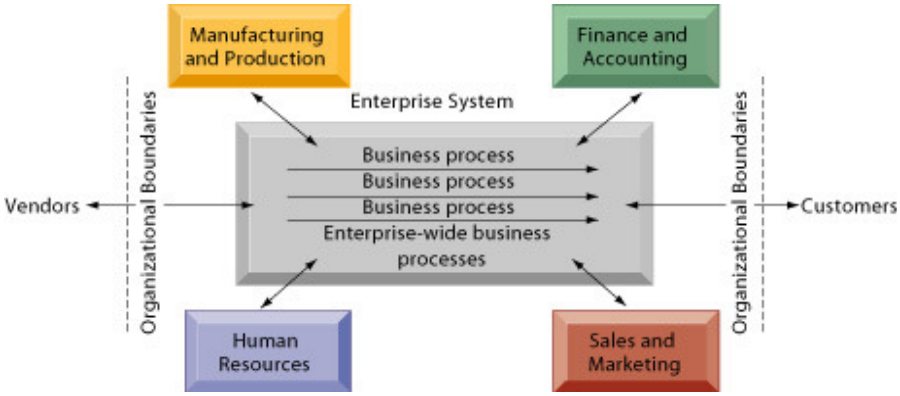


Figura 2.3 – Sistemas Empresariais

Fonte: Managing the digital firm, Laudon, K., 2006

Apesar da alta capacidade de configuração e soluções verticais, o *ERP* pode não atender a alguns requisitos essenciais e específicos do negócio. Por este motivo, a decisão de implementar um *ERP* deve levar em conta uma análise detalhada e rigorosa dos processos organizacionais e das funcionalidades do sistema.

Uma solução *ERP* engloba aspectos técnicos, organizacionais e comportamentais. Pela sua abrangência, características, e processo de implementação admite-se ser um processo de mudança organizacional. Torna-se fundamental ter em conta o impacto da redefinição dos processos e da introdução do sistema na estrutura, cultura e estratégia da empresa. A falta de compreensão desta realidade conduziu ao insucesso de inúmeras implementações destes sistemas.

2.1.2 Perspectiva Histórica e Principais Fornecedores

Perspectiva Histórica

Os sistemas *ERP* constituem uma das fases mais marcantes dos últimos anos, na área dos sistemas de informação. Têm origens nos sistemas *MRP (Material Resource Planning)*, que lidavam com as necessidades de informação relacionadas com o cálculo das necessidades de materiais para o processo produtivo. Porém e com o objectivo de ampliar a abrangência dos produtos, os fornecedores foram desenvolvendo mais módulos integrados, mas com um âmbito que cedo ultrapassou os limites da produção, como por exemplo, gestão de recursos humanos; vendas; distribuição e finanças [Adam, 2004].

Da evolução dos sistemas *MRP*, surgem os sistemas *MRPII (Manufacturing Resource Planning)*, que, para além da extensão a outras áreas funcionais, resolvem algumas das limitações dos sistemas *MRP* no que diz respeito ao planeamento finito da capacidade produtiva e gestão da execução das ordens de fabrico [Adam, 2004].

Os sistemas *ERP* constituem não só uma evolução dos sistemas *MRP* e *MRPII,* como introduzem conceitos e tecnologias existentes no mercado, como por exemplo, as linguagens de 4ª geração, sistemas de gestão de bases de dados relacionais, arquitectura cliente/servidor e independência das plataformas específicas de implementação [Adam, 2004].

O *ERP* é um conceito que surge em fins de 1980 e início dos anos 90, tendo como mercado alvo os sistemas complexos das grandes empresas [Hossain, 2002].

O conceito do *ERPII* ou *XRP (eXtended ERP)*, foi surgindo ao longo da década de noventa num panorama de proliferação da *Internet* e com o *software* empresarial a tornar-se cada vez mais flexível e capaz de se integrar com outros sistemas. Face a este cenário os fornecedores de soluções *ERP* actualizaram as suas arquitecturas no sentido de tornar os sistemas *ERP* mais habilitados para a *Web* [Laudon, 2006].

O esquema da figura 2.4 sintetiza a evolução anteriormente descrita destes sistemas.

Figura 2.4 – Evolução do ERP

Fonte: ERP- Global Opportunities & Challenges, Hossain L., 2002

Principais Fornecedores

O panorama mundial do mercado de soluções *ERP* foi dominado pelos seguintes fornecedores: SAP, Oracle, PeopleSoft, J.D. Edwards e Baan. Porém com as diversas fusões e aquisições, actualmente este domínio está concentrado em cada vez menos empresas, mas de maior dimensão:

- A SAP, que é o fornecedor líder a nível mundial;
- A Oracle é o segundo maior, após ter adquirido a PeopleSoft em 2005 (detentora do J.D. Edwards, após aquisição em 2003);
- A Infor Global Solutions, após a aquisição em 2006 da SSA Global que possuía os produtos *ERP* SSA Ln (Baan) e SSA Lx (BPCS), passou a ser igualmente um dos maiores fornecedores a nível mundial.

2.2 O Sistema ERP Baan

2.2.1 Introdução

A Baan Company foi fundada em 1978, na Holanda, por Jan Baan como uma empresa de serviços de consultoria administrativa e financeira, mas cedo começou a fornecer soluções de *software*. Iniciando com um desenvolvimento específico para um cliente na área da produção, disponibilizou em 1982, num sistema operativo proprietário, o seu primeiro pacote MRP II - Baan (BMCS). Em 1984 decidem optar por um sistema operativo mais normalizado, adoptando a plataforma *Unix*.

O lançamento da primeira geração de pacotes *ERP* (TRITON) ocorreu em 1990 e incluía os módulos Fabrico, Finanças, Distribuição e *Tools* (Administração e desenvolvimento do sistema). Em 1992 foi introduzida a arquitectura cliente/servidor e em 1995 é lançada uma das principais versões deste sistema *ERP* – o Baan IV que disponibilizava todos os módulos anteriores e adicionou o de Gestão de Projectos e Gestão de Transportes. Esta versão beneficiou das diversas parcerias estratégicas que a Baan Company estabeleceu com outros fabricantes de *software* e já estava preparada para ser suportada por sistemas operativos *Unix* e *Windows*, bem como por vários sistemas gestores de base de dados relacionais, como por exemplo, *Informix*, *Oracle*, *DB2* da *IBM* e *SQL Server* da *Microsoft*. Esta fase é também caracterizada pelo surgimento de várias soluções verticais dirigidas aos sectores da indústria automóvel, electrónica, alimentar, aeroespacial e defesa, siderurgia, papel e química.

O sistema *ERP* Baan não ficou imune à segunda vaga de sistemas *ERP* caracterizada pela exigência de uma maior flexibilidade na resposta a cenários suscitados pela proliferação da *Internet*. Neste âmbito, em 1997 o sistema *ERP* Baan apresentou soluções de integração com sistemas *Customer Relationship Management - CRM* e *Supply Chain Management – SCM, sendo* em 1998 lançada a primeira geração de aplicações *e-Business* com o claro objectivo de tornar o sistema *ERP* mais habilitado para a *Web*. Estes cenários conduziram à necessidade do desenvolvimento de soluções de integração mais abrangentes como é exemplo o Baan OpenWorld, que possibilitou em 1999 uma integração plena de vários produtos que passaram a constituir as designadas *Baan Enterprise Solutions*.

Finalmente, a versão Baan ERP 5 surge por volta do ano 2000, proporcionando um aumento bastante significativo de funcionalidade, comparativamente com a versão Baan IV, em praticamente todas as áreas.

Resumindo, os pontos-chave da história do sistema *ERP* Baan no âmbito da Baan Company são:

- 1982 - Primeiro pacote *MRP* II - Baan (BMCS)
- 1984 - Passagem do BMCS para Unix
- 1990 - Primeira geração de pacotes *ERP* (TRITON)
- 1992 - Introdução da arquitectura Cliente/Servidor
- 1994 - Modelizador Empresarial Dinâmico
- 1995 - Lançamento da do Baan IV Back Office
- 1997 - Extensão do *ERP*: *SCM*, *CRM*
- 1998 - Primeira geração de aplicações *e-Business*
- 1999 - Integração plena *Baan Enterprise Solutions*
- 2000 – Lançamento da versão Baan ERP 5

Em Agosto de 2001 a Baan Company é adquirida pela Invensys PLC e vendida à SSA Global Technologies em Julho de 2003.

Em 2005 a SSA Global Technologies lança uma nova versão do sistema *ERP* Baan, com o nome de SSA ERP LN 6.1.

Finalmente, em Maio de 2006 a SSA Global Technologies é adquirida pela Infor Global Solutions, ficando actualmente pertencente a este grande concentrador de sistemas *ERP*, com o nome mais uma vez modificado de "Infor ERP LN 6.1".

A figura 2.5 apresenta a evolução das diversas versões do sistema *ERP* Baan, sendo as principais: Baan IV; ERP 5.0c (com menor expressão) e ERP LN 6.1.

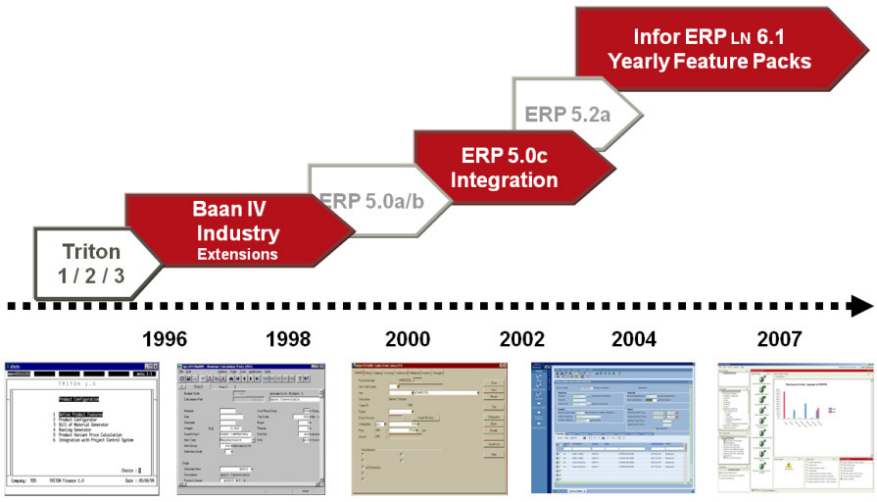


Figura 2.5 - Evolução das versões do sistema ERP Baan

Fonte: Infor Global Solutions, 2007

O sistema *ERP* Baan tem uma concepção modular e disponibiliza as funcionalidades agrupadas em vários módulos pertencentes a um dado pacote.

Na tabela 2.1 são apresentados os pacotes existentes no sistema Infor ERP LN 6.1.

Pacote	Funcionalidades
Common (tc)	São geridos diversos dados comuns que são utilizados pelos restantes módulos. São exemplos: os parceiros de negócio; artigos; calendários e dados relativos à definição da estrutura da empresa.
Manufacturing (ti)	Permite cenários de produção discreta, produção repetitiva, assemblagem e produção por projecto. Gestão de dados de engenharia e configurador de produto. Efectua a gestão das estruturas de fabrico e gamas operatórias. Gestão de ordens de produção. Gestão de ferramentas. Custeio industrial.
Order Management (td)	**Vendas**: Definição de preços e descontos; Controlo de margens; Propostas de venda; Contratos; Gestão ordem de venda; Programas de venda; Comissões e rapel; Informações de venda e marketing. **Compras:** Definição de preços e descontos; Requisições de compra; Pedidos de cotação; Contratos de compra; Gestão de ordem compra; Programas de compra; Análise e avaliação de fornecedores.
Enterprise Planning (cp)	Motor de planeamento que combina *MPS, MRP, DRP, CRP*. Várias fontes de fornecimento (produção, compra, distribuição) e estratégias de fornecimento (selecção de fornecedores para compra e armazéns para distribuição). *ATP / CTP* para artigos standard e personalizados. Agregação (multi-site) e desagregação (famílias de planeamento). Simulações de cenários e avisos por artigo, planeador e recurso. Ferramentas de análise. Indicadores de performance para comparação de cenários.
Enterprise Modeler (tg)	Permite definir os processos empresariais num interface gráfico. Modelização da organização através da gestão da biblioteca de processos e funções dos empregados numa dada versão de modelo empresarial. Definir as permissões que cada função tem num dado componente do processo de negócio e associar os empregados às funções empresariais. Gestão de todas as versões, modelos, processos, funções e empregados.
Financials (tf)	É essencialmente composto pelos módulos: Contabilidade geral; Contabilidade de clientes e de fornecedores; Gestão da tesouraria; Gestão de Activos fixos; Orçamentação e Relatórios financeiros.

People (bp)	Centraliza a gestão de empregados. Registo e aprovação de tempos e despesas de empregados.
Service (ts)	Permite responder aos cenários de manutenção interna e serviço pós-venda. Gestão de contratos. Gestão de chamadas. Manutenção condicionada. Gestão dos centros e áreas de serviço. Planeamento e controlo das ordens de serviço.
Warehouse Management (wh)	Gestão de ordens, planeamento e movimentação no armazém. *Cross-docking* dinâmico. Gestão e rastreabilidade de artigos seriados. Gestão e rastreabilidade de lotes. Gestão de zonas e localizações. Estruturas de embalagens. Relatórios e análises de stock.
Freight Management (fm)	Permite efectuar o planeamento de necessidades de expedição. Implementa os algoritmos de planeamento: entrega directa; consolidação e *pooling*. Controlo dos custos de transporte. Selecção automática de transportadores baseada em custos previstos, tempos de transporte e níveis de serviço. Facturação dos custos de transporte ao cliente.
Quality Management (qm)	Gestão das características, testes e instrumentos. Inspecções padrão, específicas e de armazenagem. Gestão de calibrações. Gestão da qualidade total (*TQM*).
Central Invoicing (ci)	Permite efectuar uma facturação centralizada das várias origens: armazém; projecto; vendas; serviços. Gestão de pedidos de facturação. Gestão de modelos e adições aos pedidos de facturação.
Project (tp)	Definição da *WBS* do projecto. Estimativa. Proposta comercial. Planeamento. Gestão de progresso dos custos e receitas. Monitorização e análise de todo o processo.
Object Data Management (dm)	Permite associar, visualizar e alterar ficheiros físicos, ao nível de qualquer objecto do sistema ERP (ao nível da transacção). Gestão documental. Gestão de repositórios. Gestão de alterações.
Tools (tt)	Administração e Desenvolvimento do Sistema.

Tabela 2.1 - Pacotes do sistema Infor ERP LN 6.1

2.2.2 Arquitectura do Sistema

A arquitectura geral do sistema *ERP* Baan é uma arquitectura cliente/servidor e está estruturada de acordo com as seguintes camadas típicas:

- **Camada de Apresentação (*Presentation Layer*)** - Um interface gráfico que corre no *Microsoft Windows* ou num *Web Browser*.
- **Camada Aplicacional (*Application Layer*)** – composta por uma máquina virtual dependente do sistema operativo, que executa objectos independentes da plataforma utilizada, uma vez que esta pode ser portável para as diversas plataformas.
- **Camada da Base de Dados (*Database Layer*)** – Esta camada é dependente da tecnologia de base de dados utilizada e representa o repositório dos dados do sistema *ERP*.

O diagrama da figura 2.6 representa a arquitectura simplificada do sistema *ERP* Baan.

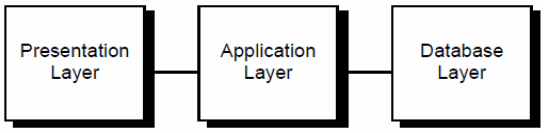


Figura 2.6 – Estrutura simplificada do sistema ERP Baan

Fonte: White Paper on Client/Server Configuration. Baan Company, 2001

Baseada nestas três camadas, é apresentada na figura 2.7 uma versão mais detalhada da arquitectura do sistema *ERP* Baan.

2.2.2 Arquitectura do Sistema (cont.)

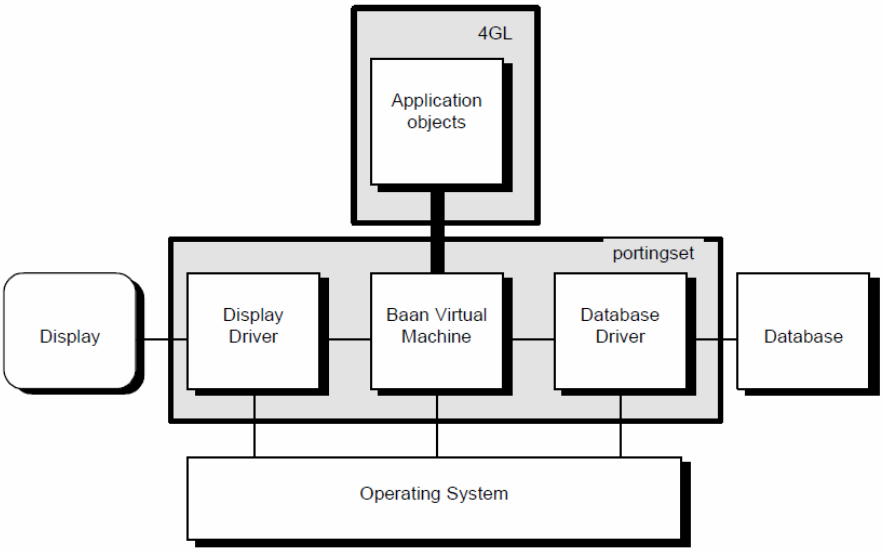


Figura 2.7 - Estrutura do sistema ERP Baan

Fonte: White Paper on Client/Server Configuration. Baan Company, 2001

Reportando à figura 2.7, no sistema *ERP* Baan cada utilizador possui os seus processos de *display driver, logic server* e *database driver*.

As aplicações são formadas pelos objectos (desenvolvidos em linguagem Baan 4GL e compilados em objectos independentes do sistema operativo e da base de dados), em conjunto com o dicionário de dados que define os respectivos modelos de dados.

A máquina virtual Baan corre os objectos 4GL sempre da mesma forma e o *kernel* de um sistema operativo corre os seus próprios objectos (processos). O sistema operativo envia informação do ecrã, para que o *display driver* possa construir os diferentes ecrãs da aplicação e recebe as respostas dadas pelos utilizadores do sistema. A máquina virtual é também responsável por enviar instruções *SQL* (Baan *SQL*) para o *database driver*, que as converte e optimiza para instruções *SQL* nativas do sistema gestor de base de dados relacional (SGBDR) da camada de base de dados. Os resultados fornecidos pelo SGBDR, são retornados às aplicações igualmente através do *database driver* e pela máquina virtual Baan.

O sistema *ERP* Baan possui uma arquitectura baseada numa estrutura multi-camada que permite a instalação do ambiente de forma flexível e escalável. Os diversos cenários possíveis de implementação devem, normalmente, levar em conta os seguintes factores:

- A gestão dos servidores aplicacionais e de base de dados;
- Escalabilidade no sentido da facilidade de aumentar o número de utilizadores e servidores aplicacionais;
- Alta disponibilidade relacionada com ambientes multi-servidor.

O sistema *ERP* Baan pode ser implementado segundo dois cenários:

- Cliente/Servidor com duas camadas – onde todas as componentes do sistema *ERP* Baan e base de dados ficam num único servidor, ficando apenas a camada de apresentação no cliente.
- Cliente/Servidor com três camadas – neste cenário as camadas principais do sistema ERP Baan – apresentação, aplicação e base de dados, estão separadas em máquinas distintas.

O esquema da figura 2.8 representa a implementação do sistema cliente/servidor *ERP* Baan com duas camadas.

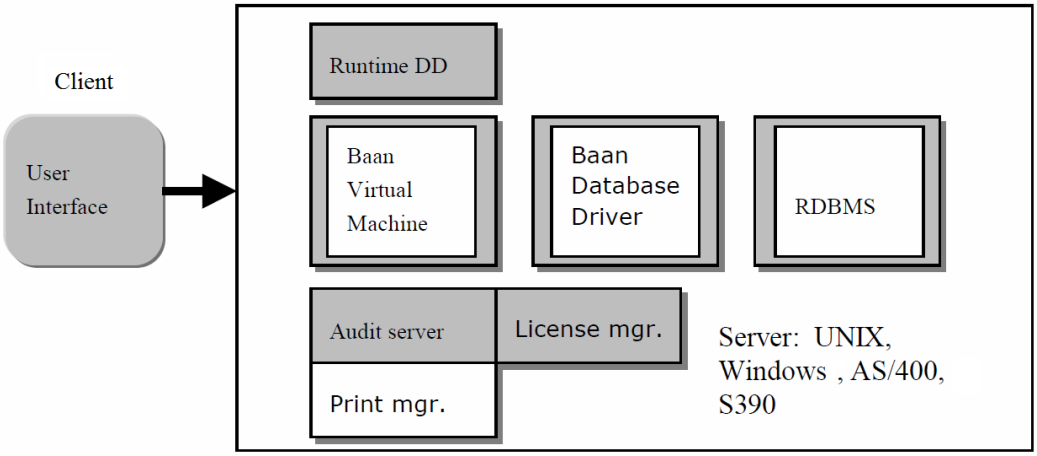


Figura 2.8 – Cenário cliente/servidor duas camadas

Fonte: White Paper on Client/Server Configuration. Baan Company, 2001

Como se pode constatar, todos os processos correm do lado servidor, excepto a camada de apresentação que fica do lado cliente. Esta configuração permite uma maior facilidade de gestão, porém o ambiente servidor não pode ser optimizado para servidor de base de dados, nem para servidor de aplicação *ERP* Baan.

2.2.2 Arquitectura do Sistema (cont.)

A implementação do sistema cliente/servidor *ERP* Baan com três camadas é representada na figura 2.9.

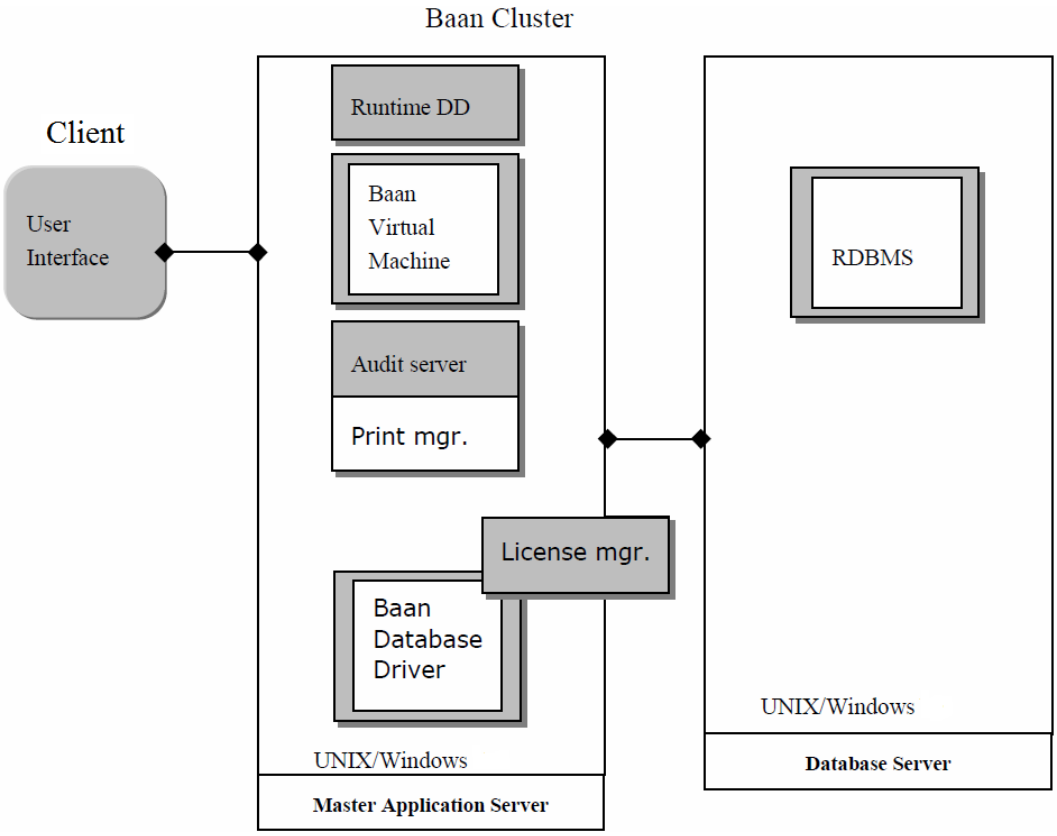


Figura 2.9 – Cenário cliente/servidor três camadas

Fonte: White Paper on Client/Server Configuration. Baan Company, 2001

Como se pode verificar, a camada aplicacional e a camada de base de dados ficam em servidores distintos. A camada de apresentação permanece do lado cliente. Esta configuração permite que os servidores de aplicação e de base de dados possam estar optimizados para a sua função, aumentando porém a complexidade da gestão destes.

2.2.3 O software Baan como Servidor de Automação OLE

Antes de abordarmos o tema específico, é conveniente efectuar um breve enquadramento sobre a tecnologia *OLE Automation*, em geral.

Automation

O *Automation*, frequentemente designando *OLE Automation,* é um mecanismo de comunicação entre processos baseado no *Component Object Model – COM* [MSFT, 1994] e foi concebido pela Microsoft, com o objectivo inicial de ser utilizado por linguagens de *scripting* (inicialmente *Visual Basic).* Actualmente é suportado por diversas linguagens de programação que corram no sistema operativo *Microsoft Windows*.

O *Automation* fornece uma infra-estrutura onde aplicações designadas controladores da automação (*automation controllers*) podem aceder e manipular objectos partilhados (*automation objects*) expostos por outras aplicações. Neste mecanismo, o controlador da automação é o cliente e a aplicação que expõe o objecto da automação é o servidor. Os servidores de objectos de automação tanto podem ser executáveis (*out-of-process*) como *Dynamic Link Libraries – DLL* (*in-process*).

Um objecto de automação é um objecto *COM* que implementa a interface pré-definida *IDispach*. Esta interface disponibiliza quatro métodos, sendo o mais relevante o método *Invoke.* Este permite chamar os métodos de uma classe pelo seu nome e com um número arbitrário de parâmetros. Nem os nomes dos métodos, nem o número e tipo dos parâmetros, necessitam de ser conhecidos na altura da compilação, (ao contrário dos objectos *COM* que não suportam *Automation*), aliás nas linguagens de *scripting* nem sequer existe compilação. Esta técnica tem o nome de *late binding*.

Dadas as vantagens inerentes, a grande maioria dos objectos *COM* são compatíveis com *Automation* e como tal fornecem, quer a forma tradicional de *early binding* na altura da compilação, como o *late binding*, fornecido pelo *automation*. Isto é conseguido através da implementação dos designados *dual interfaces*, que são interfaces derivadas de *IDispach*. Como se pode imaginar, o desenvolvimento do controlador da automação pressupõe o conhecimento do modelo empregue no objecto exportado pela aplicação servidora. Esse conhecimento pode ser transmitido através de documentação,

ou como é mais frequente e eficaz, através da disponibilização de bibliotecas de tipos (*Type Libraries*) que contêm informação acerca das classes, interfaces e outras funcionalidades expostas pelo conjunto de objectos exportados.

As bibliotecas de tipos permitem igualmente a geração de código do *Proxy/stub* necessário à interoperabilidade entre objectos *COM* e outras plataformas, como .NET e Java. Por exemplo, o .NET *Framework SDK* disponibiliza ferramentas que criam, a partir destas bibliotecas de tipos, uma .NET *DLL* para acesso a objectos compatíveis com *Automation*, utilizando quer *early binding*, quer *late binding*.

2.2.3 O software Baan como Servidor de Automação OLE (cont.)

O Baan *OleAutomationServer*

O software *ERP* Baan, em combinação com a interface de acesso (BW), pode funcionar como um servidor de *OLE Automation*. O objecto exposto por este servidor de *Automation*, permite executar funções de bibliotecas compiladas dinamicamente *DLL,* existentes remotamente no servidor *ERP* Baan. De acordo com o anteriormente exposto sobre *Automation*, este objecto pode ser utilizado quer por uma linguagem de *scripting*, como por exemplo, o *Visual Basic for Applications - VBA*, quer por uma linguagem de programação que tenha como destino o sistema operativo *Microsoft Windows*.

O objecto Baan *OleAutomationServer* fornece o conjunto de métodos e propriedades descritos respectivamente nas tabelas 2.2 e 2.3.

Métodos do Objecto Baan OleAutomationServer	
Método	Descrição
ParseExecFunction "dllname", "functioncall"	Permite chamar uma dada função com os respectivos argumentos (*functioncall*), pertencente a uma DLL (*dllname*), existente no sistema ERP Baan.
Quit	Termina a sessão de automação.

Tabela 2.2 – Métodos do Objecto OleAutomationServer

Propriedades do Objecto Baan OleAutomationServer	
Propriedade	Descrição
Timeout	Permite definir um tempo máximo para a chamada, em segundos, para impedir que o objecto fique bloqueado devido a chamadas erróneas.
FunctionCall	Utilizada para obter os valores dos argumentos que na função chamada são passados por referência.
ReturnCall	Idem da *FunctionCall*, mas se envolver dados binários deve ser esta a utilizada.
ReturnValue	Utilizada para obter o valor de retorno da função invocada. O valor retornado é sempre convertido para *string*.
Error	Contém o estado da chamada do *ParseExecFunction* e pode assumir os seguintes valores: **0** Sucesso; **-1** DLL desconhecida; **-2** função desconhecida; **-3** erro de sintaxe na função chamada; **-10** O tempo expirou; **-11** erro fatal na função.
Binary	Quando colocada a *true*, permite a recepção de dados binários na propriedade *ReturnCall*.

Tabela 2.3 – Propriedades do Objecto OleAutomationServer

2.2.3 O software Baan como Servidor de Automação OLE (cont.)

Exemplo

A título de exemplo, podemos analisar o seguinte código em *VBA* que utiliza algumas das propriedades e métodos anteriormente referidos:

```
Set BaanObj = CreateObject("Baan.Application")
BaanObj.Timeout = 5
BaanObj.ParseExecFunction "testedll", "get.nome.cliente(10518)"
If BaanObj.Error <> 0 Then
        MsgBox("Erro ocorrido:" & BaanObj.Error)
Else
        MsgBox("Valor Retornado:" & BaanObj.ReturnValue)
End If
BaanObj.Quit
Set BaanObj = Nothing
```

O objecto é instanciado e definido um *Timeout* de 5 segundos. Supondo que a *DLL* testedll e a função get.nome.cliente existem, apareceria a uma caixa de mensagem com o nome do cliente cujo código é 10518. No caso de ocorrência de algum erro apareceria uma mensagem com o código do mesmo. Se por exemplo, a execução da função demorasse mais do que 5 segundos apareceria "Erro ocorrido: -10". As instruções finais dizem respeito ao término da sessão de automação e à libertação da memória do objecto.

Restrições

Para finalizar esta secção são apresentadas algumas das limitação do uso deste servidor de *OLE Automation*:

- Não podem ser fornecidos *arrays* como argumentos de uma função chamada;
- Não são suportadas funções que tenham número variável de argumentos;
- Não são suportadas duas aplicações clientes em simultâneo para o mesmo objecto;
- O tamanho máximo da *string* nas propriedades *FunctionCall* e *ReturnValue* é de 4KB.

A *BWConfigurationClass*

A *BWConfigurationClass* constitui igualmente uma classe de *OLE Automation,* neste caso, exposta por um componente *COM* [MSFT, 1994] da aplicação *Baan Environment and Configuration Selector – BECS* que permite criar, ler e manipular a informação dos ficheiros de configuração do acesso cliente ao sistema *ERP* Baan.

Esta classe expõe o conjunto de métodos apresentados na tabela 2.4.

Métodos da classe *BWConfigurationClass*	
Método	Descrição
Load	Permite carregar a informação existente num ficheiro de configuração (.BWC) , cujo nome é passado como argumento.
Save	Permite guardar as informações de configuração.
Rename	Renomeia um ficheiro de configuração.
Delete	Elimina um ficheiro de configuração.

Tabela 2.4 – Métodos da classe BWConfigurationClass

As suas propriedades expõem toda a informação existente no interface gráfico, que se pode observar na figura 2.10, utilizado na definição das configurações.

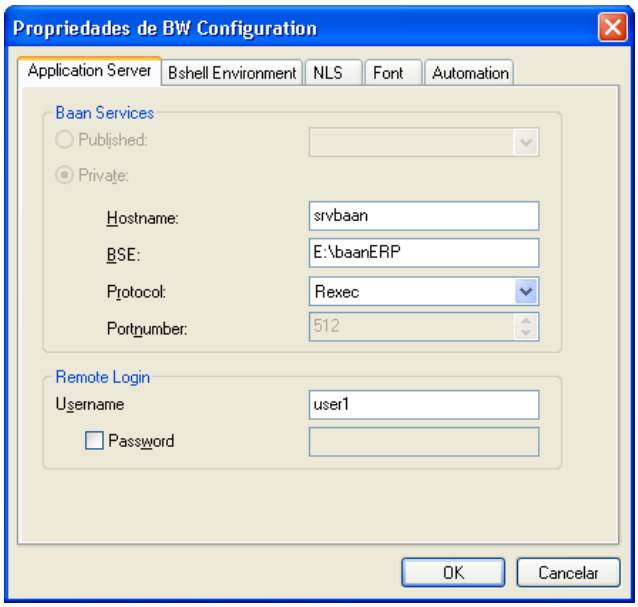


Figura 2.10 – Ecrã de configuração do acesso ao sistema *ERP* Baan

2.2.3 O software Baan como Servidor de Automação OLE (cont.)

A *BWConfigurationClass* (cont.)

A tabela 2.5 refere algumas das propriedades mais importantes.

Propriedades da classe *BWConfigurationClass*	
Propriedade	Descrição
Hostname	Permite definir o nome de rede do servidor *ERP* Baan.
Username	Define o nome de utilizador do sistema operativo associado ao utilizador do sistema *ERP* Baan.
Password	Define a palavra passe.
LoginDialog	Permite definir se as credenciais podem ser guardadas, sendo neste caso, omitido o ecrã inicial de solicitação das mesmas.
ServerBSE	Define o caminho do ambiente executável do sistema *ERP* Baan.
Protocol	Define o protocolo de comunicação utilizado.
Command	Permite a definição de comandos que definem vários aspectos do acesso cliente e do utilizador Baan. Por exemplo, permite que o utilizador entre noutra companhia que não a pré-definida.
ClassName	Permite obter o nome da classe utilizada para *OLE Automation*.

Tabela 2.5 – Propriedades da classe BWConfigurationClass

2.2.4 Execução das sessões via Application Function Server - AFS

Esta secção visa descrever a funcionalidade e arquitectura do *Application Function Server*, existente nos sistemas *ERP* Baan IV, 5.x e LN 6.1.

Nos sistemas *ERP* Baan as sessões são primariamente baseadas em interfaces gráficos para o utilizador, constituindo o principal meio através do qual a lógica de negócio é exposta e implementada. Porém adicionalmente a esta forma interactiva, a lógica de negócio de uma dada sessão pode ser acedida de forma programática por outras sessões (Baan 4GL) e por outros programas (Baan 3GL).

O acesso de forma programática às sessões Baan é efectuado, por intermédio de um protocolo de mensagens de baixo nível. Este protocolo encontra-se encapsulado num conjunto de primitivas de função designado *Application Function Server (AFS).*

Arquitectura

Os principais aspectos da arquitectura do *AFS* aparecem então esquematizados na figura 2.11.

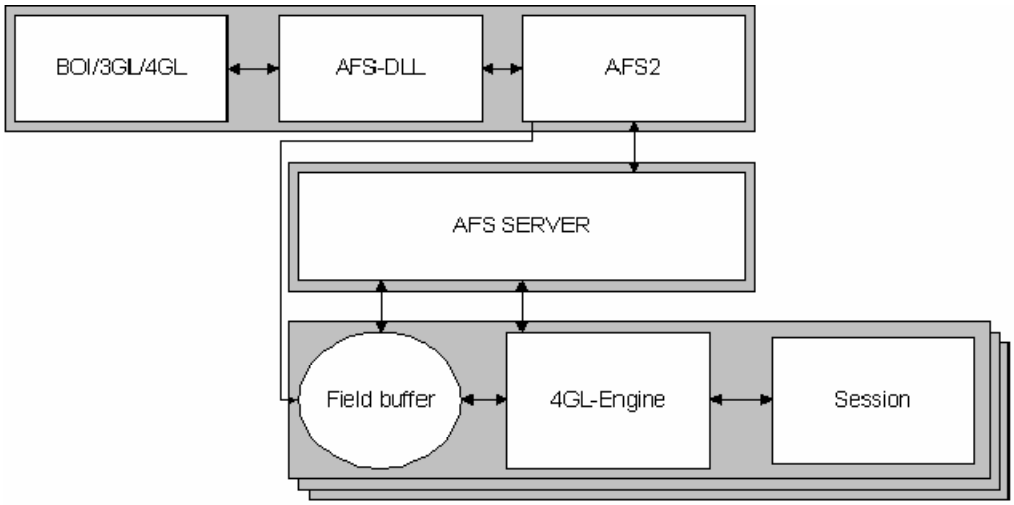


Figura 2.11 – Arquitectura do *Application Function Server*

Fonte: Developer's Guide for Application Function Server. SSA Global, 2006

2.2.4 Execução das sessões via Application Function Server – AFS (cont.)

Segue-se uma breve descrição de cada uma das componentes da arquitectura do *AFS*.

BOI/3GL/4GL - O programa que pretende aceder à lógica de uma sessão Baan ERP. Este acesso pode ser originado por um:

- *Business Object Interface – BOI* (interface para objecto de negócio);
- 3GL – programa compilado e desenvolvido numa linguagem de 3ª geração proprietária para o sistema *ERP* Baan - o *Baan Tools*;
- 4GL – programa de uma sessão Baan que pretende executar funcionalidades de outra sessão através do *AFS*.

AFS-DLL - É uma *Dynamic Link Library – DLL* composta por funções que encapsulam as chamadas às primitivas de função do *AFS*.

A utilização desta *DLL* é opcional, podendo o *BOI*, 3GL ou 4GL aceder directamente às primitivas do *AFS*. A geração desta AFS-DLL permite ter nomes intuitivos (baseados nas descrições do dicionário de dados) para as funções e campos de ecrã. A utilização desta *DLL* permite evitar erros que desta forma são detectados pelo compilador e não em tempo de execução. Porém esta *DLL* constitui um nível acrescido ao programa standard Baan ou "stpapi" e provoca, consequentemente, uma redução no desempenho.

AFS2 - Representa a *DLL* que contém a implementação das primitivas de função do *AFS*. Esta *DLL* comunica com o motor 4GL através de um protocolo dedicado BMS, para que a sessão execute a funcionalidade pedida.

AFS Server - Processo servidor que comunica com o motor 4GL através de um protocolo dedicado BMS, permitido que a sessão execute a funcionalidade pedida.

Field buffer - Contém nomes e valores dos campos de ecrã e tabela e que constituem os valores de entrada para a sessão. Se a sessão Baan necessita destes valores durante o processamento dos respectivos campos, o motor 4GL obtém esses valores do "*field buffer*" e torna-os disponíveis na sessão. Após a alteração destes valores pela sessão (originados por exemplo por uma inserção ou uma actualização) o "*field buffer*" é actualizado permitindo ao *BOI*, 3GL ou 4GL o acesso a estes novos valores.

4GL-Engine - Representa o motor responsável por todo o processamento dos ecrãs, campos, eventos e operações da tabela principal, para cada sessão de aplicação Baan 4GL. Este motor 4GL é precisamente o mesmo utilizado na execução da sessão em modo interactivo, existindo neste caso um programa de interface para o utilizador, como por exemplo, o *Baan Windows (BW)*. No modo interactivo a sessão aguarda por parte do utilizador, pelo preenchimento de campos do ecrã, accionamento de botões, opções de menu *etc*. No modo de acesso programático a sessão espera por comandos do *AFS*. A variável "*api.mode"* assume o valor verdadeiro quando a sessão é acedida programaticamente e pode ser utilizada na lógica da sessão de forma a obter comportamentos diferenciados caso esta seja chamada por *AFS* ou de forma interactiva.

Session logic - As sessões de aplicação Baan 4GL executadas com recurso ao *AFS*. Podem coexistir várias sessões numa chamada *AFS,* se esta abrir outras subsessões ou se for uma sessão sincronizada do tipo multi-ocorrência/ocorrência única (existente apenas no Baan ERP 5.x e ERP LN 6.1).

Neste tipo de execução, cada sessão pode ter apenas uma instância activa. O *AFS* efectua o rastreio das instâncias de sessão e envia as mensagens BMS para o número de processo que está internamente associado a essa instância de sessão.

2.2.4 Execução das sessões via Application Function Server – AFS (cont.)

Exemplo de utilização do *AFS*

Para uma melhor compreensão, segue-se um exemplo concreto de utilização do *AFS*:

```
 1 #pragma used dll ottstpapihand
 2
 3 function extern domain tcorno amemsdll0001.InserirOrdVendaCab(
 4     const   domain  tccom.bpid     e_ofbp,| Parceiro Cliente
 5     const   domain  tccwoc         e_cofc,| Departamento
 6     const   domain  tccotp         e_sotp,| Tipo de O. Venda
 7     const   domain  tcseri         e_seri,| Série de O. Venda
 8     const   domain  tcmcs.st30     e_corn,| Ord. Cliente
 9     const   domain  tcrefa         e_refa,| Referencia A
10     const   domain  tccdec         e_cdec,| Cond. de Entrega
11     const   domain  tcemno         e_crep,| Rep. Vendas
12     const   domain  tcdisc         e_odis,| Desconto de Ordem
13     const   domain  tccpay         e_cpay,| Condição de Pagamento
14     ref     string         e_err )
15 {
16
17     long    ret
18     string  last_err(500)
19     string  mainsession(20)
20     string  subsession(20)
21     domain  tcorno  aux.orno
22
23     mainsession = "tdsls4500m000"
24     subsession  = "tdsls4100s000"
25
26     ret = stpapi.synchronize.dialog(mainsession, "add", e_err)
27
28     if ret then
29             stpapi.put.field(subsession, "tdsls400.ofbp", e_ofbp)
30             stpapi.enum.answer(subsession, "tdsls4100.1", tcyesno.no)
31             |Inserir endereço?
32
33             stpapi.put.field(subsession, "tdsls400.cofc", e_cofc)
34             stpapi.enum.answer(subsession, "tdsls41005", tcyesno.yes)
35             |Recolher predefinições da combinação departamento/parceiro facturado?
36
37             stpapi.put.field(subsession, "tdsls400.sotp", e_sotp)
38             stpapi.put.field(subsession, "tdsls400.orno", e_seri)
39             stpapi.put.field(subsession, "tdsls400.corn", e_corn)
40             stpapi.put.field(subsession, "tdsls400.refa", e_refa)
41             stpapi.put.field(subsession, "tdsls400.cdec", e_cdec)
42             stpapi.put.field(subsession, "tdsls400.crep", e_crep)
43             stpapi.put.field(subsession, "tdsls400.odis", str$(e_odis))
44             stpapi.put.field(subsession, "tdsls400.cpay", e_cpay)
45
46             ret = stpapi.insert( mainsession, true, last_err)
47
48             if isspace(last_err) then
49                    stpapi.get.field(mainsession, "tdsls400.orno", aux.orno)
50             else
51                    e_err=last_err
52
54                     ret = stpapi.recover(subsession, last_err)
55                     ret = stpapi.recover(mainsession, last_err)
56             endif
57
58             stpapi.end.session(subsession, last_err)
59             stpapi.end.session(mainsession, last_err)
60
61             return(aux.orno)
62     endif
63
64     stpapi.end.session(mainsession, last_err)
65
66     return(aux.orno)
67 }
```

Explicação do Exemplo

O código da função `amemsdll0001.InserirOrdVendaCab`, anteriormente descrito, tem por objectivo a criação de um cabeçalho de ordem de venda, recorrendo ao *AFS*. Caso essa ordem seja inserida com sucesso o número desta é devolvido pela função. Caso contrário, o parâmetro *e_err,* passado por referência, é preenchido com a mensagem de erro devolvida pela execução da lógica da sessão, através do *AFS*.

Na tabela 2.6 apresentam-se comentários relativos ao código relevante para a explicação da utilização do *AFS*.

Linha de código	Explicação
1	Na compilação de programas que usem o *AFS* é necessária a referência à *DLL* ottstpapihand (*API* handler do *AFS*).
26	A primeira chamada da *API* para a sessão "tdsls4500m000" (Ordens de Venda) resulta na sua activação, bem como na activação da subsessão "tdsls4100s000", visto tratar-se de uma sessão sincronizada do tipo multi-ocorrência/ocorrência única.
29, 33 e 37 a 44	Para as sessões já activadas, a chamada à função stpapi.put.field permite actualizar o valor de um campo de ecrã da respectiva sessão, actualizando o "*field buffer*", conforme descrito na arquitectura do *AFS*.
30 e 34	Permite responder a questões colocadas durante a execução da sessão. Como resposta é fornecido o enumerado pretendido. Por exemplo, "tcyesno.yes" para responder com a opção "Sim" a uma dada questão identificada pelo seu código.
46	Insere um registo na respectiva sessão, utilizando os valores dos campos existentes no "*field buffer*". Durante a inserção, são executadas secções relevantes da sessão ("*choice.add.set/before.choice*",…), secções relativas aos campos ("*before.field*", "*before.checks*" e "*check.input*") e ainda o "*check.property hook*", no caso a tabela possuir *DAL* (Data Access Layer).
49	Obtém o valor corrente de um determinado campo, a partir do "*field buffer*" ou directamente da sessão. Neste caso, é retornado o número de ordem de venda atribuído após a inserção do registo.
54, 55	Quando surgem erros torna-se necessário efectuar o "recover" das sessões para estas cessem o seu modo de actualização.
58, 59	As sessões têm de ser terminadas. As subsessões devem ser terminadas antes das respectivas sessões principais.

Tabela 2.6 – Explicação do exemplo de utilização do *AFS*

2.2.4 Execução das sessões via Application Function Server – AFS (cont.)

Se num dado programa Baan utilizarmos a função descrita anteriormente da seguinte forma:

```
...

num.ord.venda = amemsdll0001.InserirOrdVendaCab("100000472", "DVDA", "VAS",
                                                 "32", "", "", "", "",
                                                 "VG0004", 5.00, "901", err_mess)

...
```

Corresponderia basicamente a efectuar as seguintes acções em modo interactivo:

1 – Abrir a sessão tdsls4500m000 (Ordens de Venda) e escolher a opção novo registo

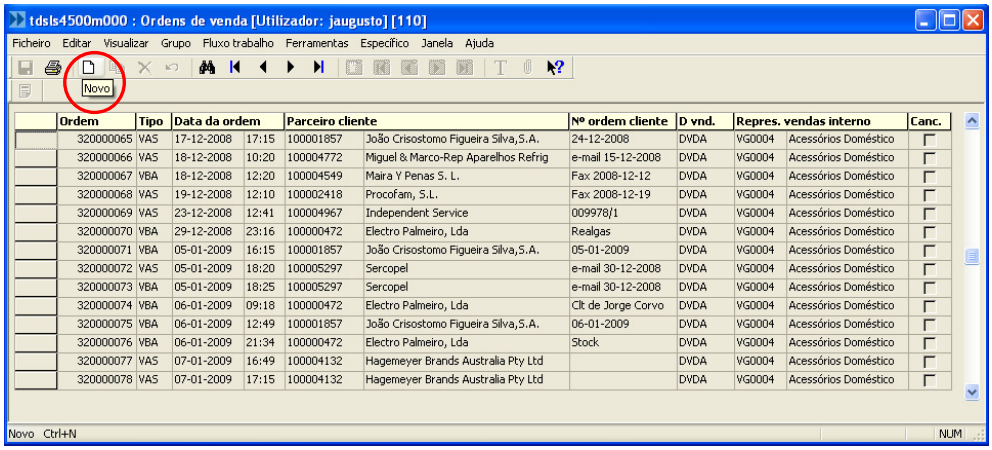


Figura 2.12 – Novo registo na sessão de ordens de venda

2 – Preencher os respectivos campos do ecrã da subsessão

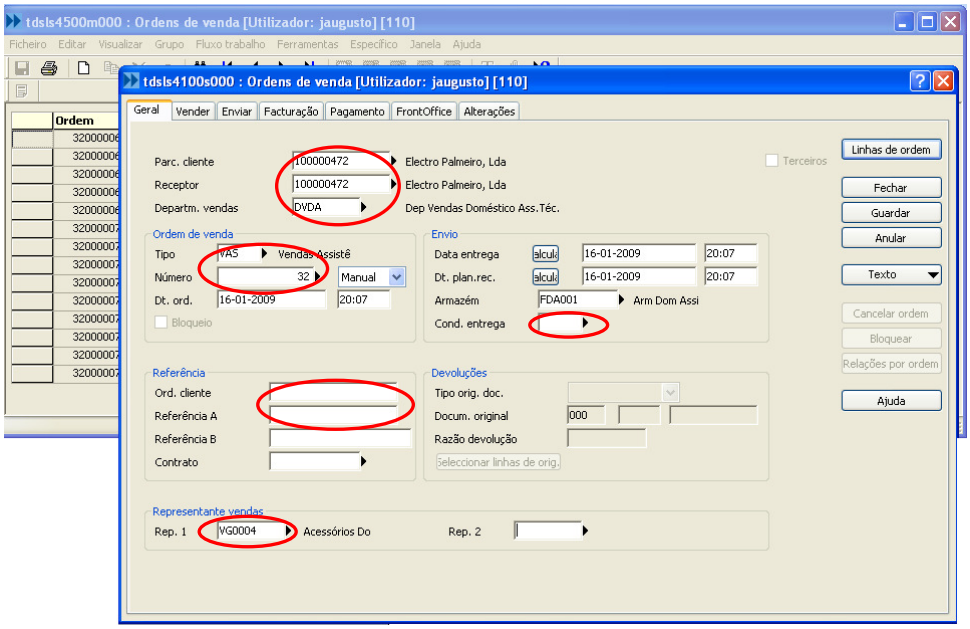


Figura 2.13 – Campos da subsessão de ordens de venda

2.2.4 Execução das sessões via Application Function Server – AFS (cont.)

3 – Preencher restantes campos do ecrã

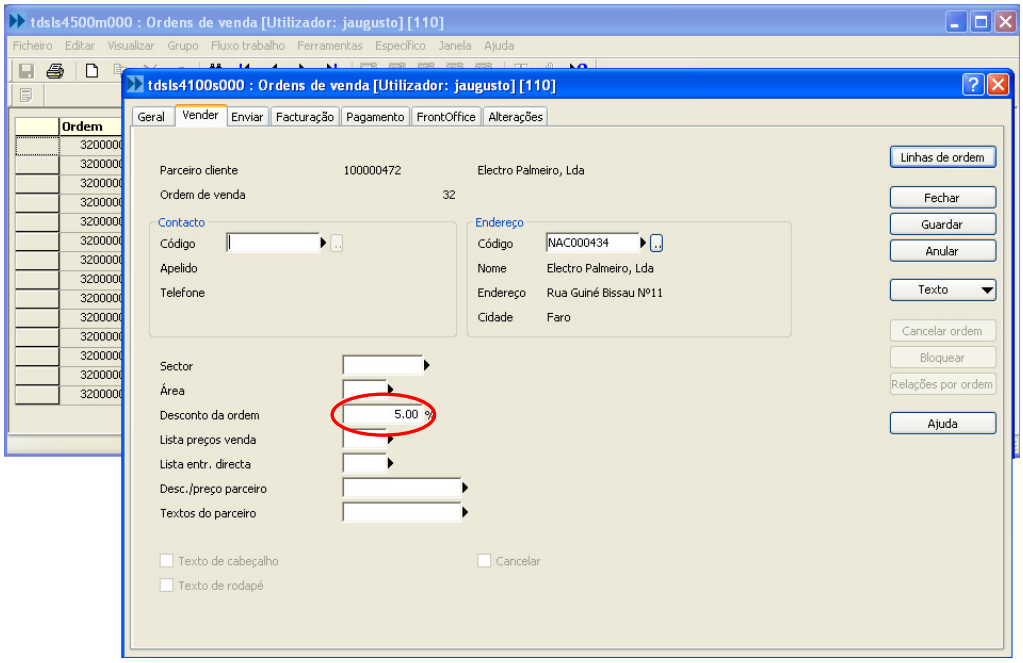


Figura 2.14 – Campo desconto de ordem na subsessão ordens de venda

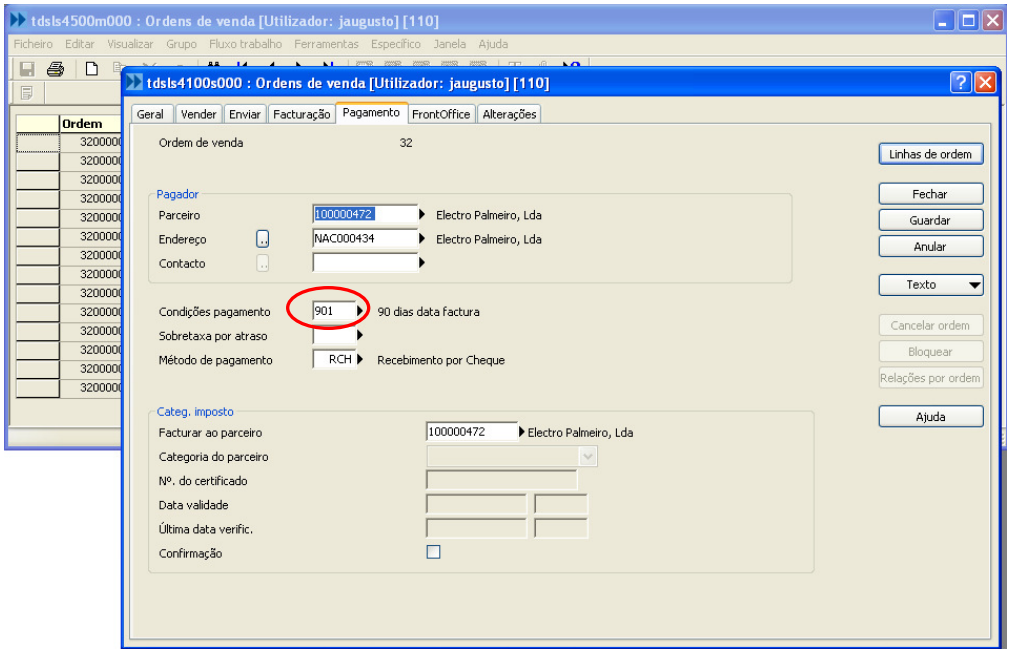


Figura 2.15 – Campo condições de pagamento na subsessão ordens de venda

2.2.4 Execução das sessões via Application Function Server – AFS (cont.)

4 – Efectuar as operações de guardar e fechar

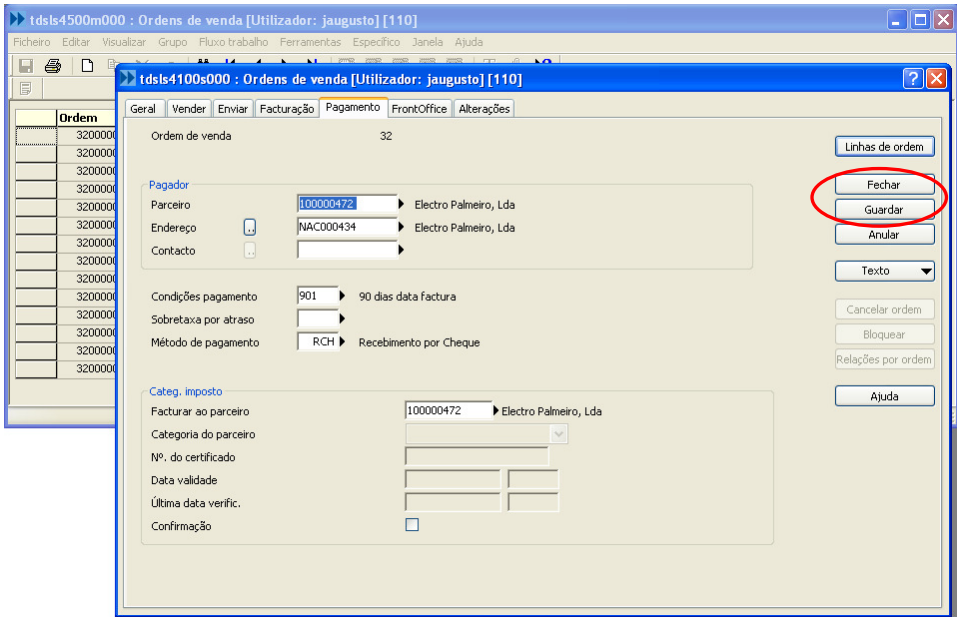
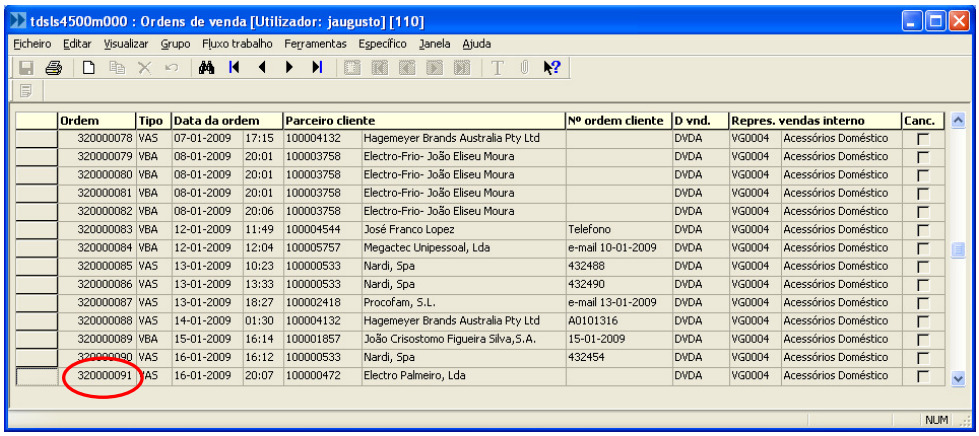


Figura 2.16 – Opções guardar e fechar na subsessão ordens de venda

5 – Obter o número de ordem de venda do registo inserido



Ordem	Tipo	Data da ordem		Parceiro cliente		Nº ordem cliente	D vnd.	Repres. vendas interno		Canc.
320000078	VAS	07-01-2009	17:15	100004132	Hagemeyer Brands Australia Pty Ltd		DVDA	VG0004	Acessórios Doméstico	
320000079	VBA	08-01-2009	20:01	100003758	Electro-Frio- João Eliseu Moura		DVDA	VG0004	Acessórios Doméstico	
320000080	VBA	08-01-2009	20:01	100003758	Electro-Frio- João Eliseu Moura		DVDA	VG0004	Acessórios Doméstico	
320000081	VBA	08-01-2009	20:01	100003758	Electro-Frio- João Eliseu Moura		DVDA	VG0004	Acessórios Doméstico	
320000082	VBA	08-01-2009	20:06	100003758	Electro-Frio- João Eliseu Moura		DVDA	VG0004	Acessórios Doméstico	
320000083	VBA	12-01-2009	11:49	100004544	José Franco Lopez	Telefono	DVDA	VG0004	Acessórios Doméstico	
320000084	VBA	12-01-2009	12:04	100005757	Megactec Unipessoal, Lda	e-mail 10-01-2009	DVDA	VG0004	Acessórios Doméstico	
320000085	VAS	13-01-2009	10:23	100000533	Nardi, Spa	432488	DVDA	VG0004	Acessórios Doméstico	
320000086	VAS	13-01-2009	13:33	100000533	Nardi, Spa	432490	DVDA	VG0004	Acessórios Doméstico	
320000087	VAS	13-01-2009	18:27	100002418	Procofam, S.L.	e-mail 13-01-2009	DVDA	VG0004	Acessórios Doméstico	
320000088	VAS	14-01-2009	01:30	100004132	Hagemeyer Brands Australia Pty Ltd	A0101316	DVDA	VG0004	Acessórios Doméstico	
320000089	VBA	15-01-2009	16:14	100001857	João Crisostomo Figueira Silva,S.A.	15-01-2009	DVDA	VG0004	Acessórios Doméstico	
320000090	VAS	16-01-2009	16:12	100000533	Nardi, Spa	432454	DVDA	VG0004	Acessórios Doméstico	
320000091	VAS	16-01-2009	20:07	100000472	Electro Palmeiro, Lda		DVDA	VG0004	Acessórios Doméstico	

Figura 2.17 – Campo número de ordem na sessão ordens de venda

2.2.4 Execução das sessões via Application Function Server – AFS (cont.)

Automatização do desenvolvimento.

Normalmente, não é necessário criar os *BOI* e as *AFS-DLL* a partir do zero. Os *BOI* podem ser gerados com uma das seguintes ferramentas: *BOI Generator; BOI Builder*.

As *AFS-DLLs* podem ser geradas pela sessão Baan "ttstpcreatdll" (*AFS-DLL generator*).

Por convenção o código atribuído à *DLL* gerada para uma dada sessão, é o mesmo com um "f" inserido depois do código do módulo. Por exemplo, a *AFS-DLL* gerada para a sessão tdsls4500m000 (Ordens de Venda), terá como o código tdslsf4500m000.

A *AFS-DLL* contém funções para:

- atribuição e obtenção de valores dos campos de ecrã da sessão;
- adicionar, modificar e eliminar registos da sessão;
- percorrer os registos da sessão;
- estabelecer respostas às questões colocadas no decorrer da lógica da sessão;
- manipulação de subsessões;
- obtenção de mensagens de erro.

Apesar do auxílio prestado por estas ferramentas, só em casos muito simples de integração é que o código gerado dará alguma resposta. O cenário normal é ser necessário desenvolver código para controlar o fluxo e o correcto desempenho, sobretudo em sessões mais complexas com chamadas a subsessões e opções específicas.

2.2.5 Suporte ao Processamento de XML

O Objecto *XML* é uma extensão da máquina virtual Baan (Baan *VM*), que permite às aplicações 3GL/4GL trocarem dados estruturados utilizando o formato *XML* [W3C, 2009a].

A funcionalidade do objecto *XML* está integrada na Baan *VM* e é acessível a estes programas utilizando um conjunto de funções 3GL que por convenção têm o prefixo "xml" (exemplo *xmlRead*, *xmlNewNode*, ...).

A arquitectura da funcionalidade do Objecto *XML* pode ser representada pelo esquema da figura 2.18.

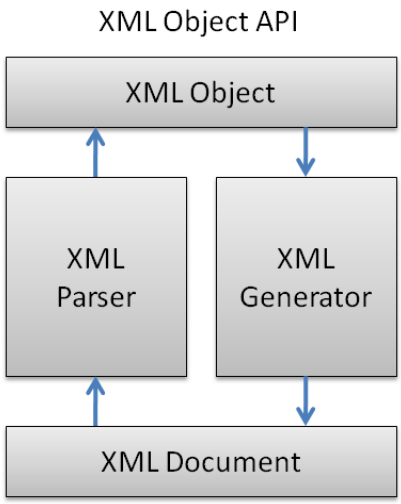


Figura 2.18 – Arquitectura da funcionalidade do Objecto XML

XML Object API – Representa um conjunto de funções 3GL disponibilizado às aplicações 3GL/4GL, para que possam aceder e manipular um objecto *XML*.

XML Object – É uma estrutura em memória, na Baan *VM*, que representa de forma aproximada um documento *XML*.

XML Parser – Componente cuja função é converter um documento *XML,* num Objecto *XML*.

XML Generator – Componente cuja função é converter um Objecto *XML*, num documento *XML*.

Estrutura do Objecto *XML*

A estrutura de um Objecto *XML* na Baan *VM* pode ser composta por quatro tipos de nós:

- **XML_ELEMENT** - nó que descreve um elemento *XML;*
- **XML_DTD** – nó que descreve um *DTD* (*Document Type Definition*) *XML*;
- **XML_DATA** – nó que descreve um item de dados *XML*;
- **XML_PI** – nó que descreve uma *Processing Instruction XML.*

Os tipos XML_ELEMENT e XML_DTD, podem ser nós internos ou folhas de uma árvore. Apenas estes nós podem conter outros nós filhos. Podem igualmente ter atributos, composto por um par: nome; valor. Os nós do tipo XML_DATA e XML_PI, são sempre folhas da árvore do objecto *XML.*

Para o seguinte exemplo de documento XML, é criada estrutura do objecto *XML* que o representa conforme a figura 2.19.

```
<?xml version="1.0"?>
 <!DOCTYPE CMF SYSTEM "cmf1.dtd">
 <CMF>
         <IDENTIFICATION
                 MESSAGE-ID="34a98u0erirori"
                 CLASS="order"
                 SUBJECT="Subject string"
         />
         <RECIPIENTS>
                 <FROM
                         NAME="Wiebe Riphagen"
                         TYPE="SMTP"
                         ADDRESS="wriphagen@baan.nl"
                 />
          </RECIPIENTS>
         <MESSAGE>The message text
 </CMF>
```

2.2.5 Suporte ao Processamento de XML (cont.)

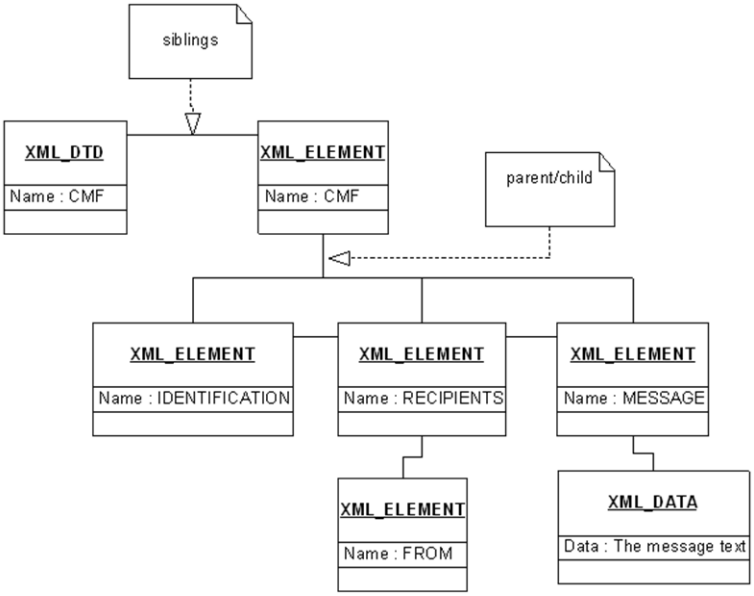


Figura 2.19 – Estrutura do objecto XML

2.2.5 Suporte ao Processamento de XML (cont.)

Objecto *XML* e Comunicação entre Processos

As funções 3GL de suporte ao objecto *XML*, acedem aos seus nós através de números identificadores. Este números são únicos no âmbito de uma máquina virtual Baan e como tal, podem ser utilizados entre processos dentro da mesma Baan *VM*, mas não podem ser transferidos para processos de outras máquinas virtuais Baan.

Ao transferir nós entre processos aplicam-se as seguintes restrições:

- Existe sempre um processo dono do objecto *XML*;
- Apenas ao processo dono é permitido alterar ou eliminar partes da árvore. Os restantes processos têm apenas acesso de leitura;
- Quando o processo dono termina, a árvore é implicitamente eliminada, o que significa que os restantes processos deixam de poder utilizar a referência a esta.

Um processo torna-se o dono de uma árvore *XML* ao chamar uma das seguintes funções:
xmlRead; xmlReadFromString; xmlNewNode ; xmlNewDataElement; xmlDuplicate; xmlRewriteDataElement; xmlDuplicateAndInsert; xmlDuplicateAndAppend; xmlDuplicateAndInsertInChilds; xmlDuplicateAndAppendtoChilds.

Um processo pode transferir a posse de uma cópia da árvore *XML* para outro processo, recorrendo à função *xmlDuplicateToProcess* e indicando o *PID* desse processo.

2.2.5 Suporte ao Processamento de XML (cont.)

Ordem de processamento da árvore *XML*

São várias as funções 3GL de suporte ao objecto *XML* que possuem como argumentos um intervalo de nós definido por um nó inicial e um nó final. Quando só é especificado o nó inicial, percorrem-se os diversos nós sempre segundo o algoritmo *Depth-first search (DFS)* representado na figura 2.20. Noutros casos é utilizado o algoritmo *Breadth-First Search (BFS)* representado na figura 2.21.

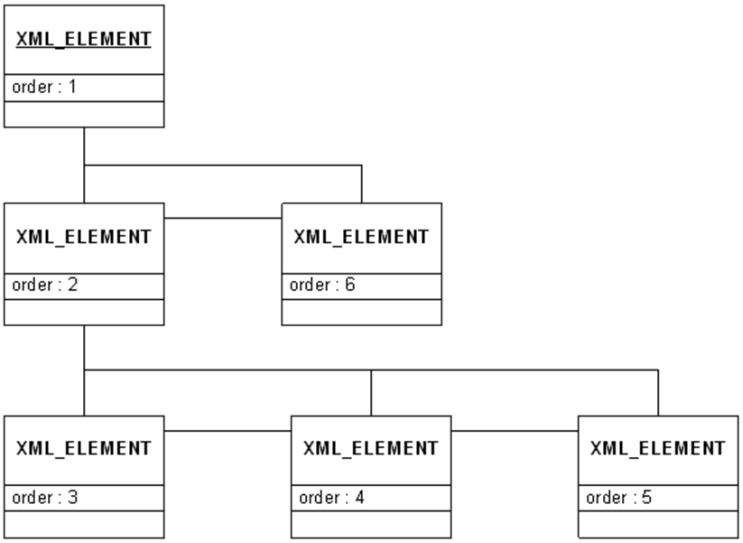


Figura 2.20 – Algoritmo Depth-First Search (DFS)

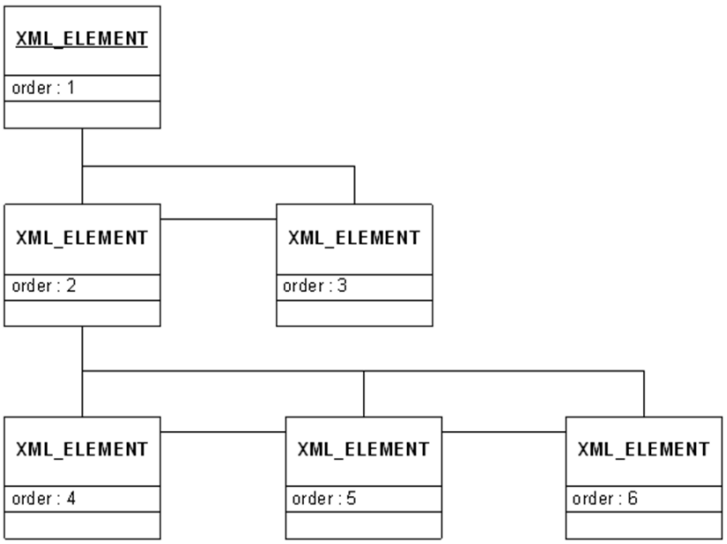


Figura 2.21 – Algoritmo Breadth-First Search (BFS)

Restrições do Objecto *XML*

O objectivo principal do objecto *XML*, integrado na máquina virtual Baan, não é a concepção de aplicações avançadas de geração/interpretação de documentos *XML*, mas sim permitir às aplicações 3GL/4GL trocarem dados estruturados utilizando o formato *XML*. Como tal as capacidades de geração e interpretação de documentos *XML* é algo limitada. Como exemplos dessas limitações podemos citar as seguintes:

Na interpretação

- Apenas verifica a boa formação do documento *XML*. Não efectua a validação usando um *DTD* (*Document Type Definition*) [W3C, 2009], ou um *XSD (XML Schema Definition)* [W3C, 2009c];
- Referências a entidades externas ao documento *XML* são ignoradas;
- O número máximo de níveis de profundidade de um elemento é de 256;
- A interpretação do documento termina se for encontrado um erro;
- É suportada apenas a interpretação completa do documento, resultando num objecto XML em memória da Baan *VM*;

Na geração

- É sempre colocada a seguinte declaração fixa *<?xml version="1.0"?>* ;
- O *encoding* de saída é sempre UTF-8;
- O objecto *XML* tem de estar completamente residente previamente na memória da Baan *VM*, para que possa ser gerado algum documento *XML*.

Capítulo 3

Aplicações Ocasionalmente Conectadas

Este capítulo tem dois objectivos fundamentais. O primeiro diz respeito a fornecer uma breve perspectiva sobre as aplicações ocasionalmente conectadas. O segundo relaciona-se com o estudo de boas práticas e padrões a utilizar no suporte ao modo ocasionalmente conectado. Dados os objectivos, numa primeira parte, são abordados os conceitos e cenários de utilização das aplicações ocasionalmente conectadas e descritas as principais estratégias relativas ao desenho da sua arquitectura. A segunda parte visa efectuar uma análise mais detalhada do suporte ao modo ocasionalmente conectado fornecido por alguns dos blocos aplicacionais existentes na *Smart Client Software Factory – SCSF*.

3.1 Conceitos e Cenários de Utilização

3.1.1 Aplicação Ocasionalmente Conectada

Apesar de assistimos nos dias de hoje a um mundo cada vez mais conectado, existem várias situações em que não é possível garantir que uma dada aplicação tenha conectividade à rede, ou que os serviços aos quais está a tentar aceder estejam disponíveis.

Uma aplicação diz-se ocasionalmente conectada se por vezes não consegue interagir com os serviços através da rede em tempo útil, ou nem sequer consegue estabelecer uma ligação, mas fornecendo nesta situação um conjunto de funcionalidades que permitem aos utilizadores a interacção com a aplicação da forma, mais parecida possível da situação conectada [Boulter, 2004].

As aplicações ocasionalmente conectadas permitem a realização de tarefas, mesmo quando não estão conectadas aos recursos de rede e posteriormente realizar as actualizações dos dados através de uma operação de sincronização, de forma transparente para o utilizador. Estas actualizações podem ocorrer quase imediatamente, ou podem até demorar dias a serem concluídas. Este cenário permite uma maior produtividade e eficiência aos utilizadores, proporcionando-lhes uma melhor usabilidade da aplicação.

Para que a aplicação forneça todas as potencialidades do cenário ocasionalmente conectado existe normalmente um conjunto de dados locais necessários à satisfação da lógica de negócio da aplicação. Os dados que reflectem o trabalho realizado pelo utilizador no modo desconectado, são utilizados nas operações de sincronização. Estas operações actualizam igualmente um conjunto de dados base necessários ao funcionamento da aplicação.

O conjunto exacto de funcionalidades e capacidades que a aplicação deve suportar de forma ocasionalmente conectada, depende de vários factores de ordem operacional e técnica. Mas a ideia geral é que por muito limitado que seja, tem de existir sempre um conjunto mínimo de funcionalidades disponibilizadas pela aplicação enquanto desconectada [Boulter, 2004].

3.1.2 Cenários Comuns de Utilização

As aplicações ocasionalmente conectadas são extremamente úteis em várias situações. Uma envolve o conhecimento explícito de trabalhar de forma desconectada, como as seguintes:

- Um técnico de assistência pós-venda, que possui no seu dispositivo móvel as ordens de serviço a executar para um dia de trabalho, os dados de *stock* de artigos existentes na sua viatura, preços dos artigos e outros dados base. Durante a execução das ordens vai declarando as peças e horas gastas;
- Um vendedor que necessita de registar uma encomenda, tendo já no seu dispositivo móvel os dados dos artigos, lista de preços e níveis de desconto para o cliente visitado;
- Um agente de seguros que necessita de criar uma apólice, enquanto está fora do escritório e a aplicação possui os dados relevantes para o cálculo do prémio e fornecimento de detalhes da apólice.

A outra diz respeito a situações provocadas por intermitência ou latência dos recursos de rede, em que a aplicação deve gerir a situação (conectada/desconectada) preferencialmente de forma transparente para o utilizador. São normalmente exemplos as situações associadas à utilização da aplicação em que o utilizador está em movimento, por exemplo, enquanto viaja. São igualmente exemplos as situações em que existe uma indisponibilidade temporária do serviço ao qual a aplicação tenta aceder.

3.2 Desenho da Arquitectura

Existem duas abordagens principais relativamente ao desenho da arquitectura das aplicações ocasionalmente conectadas, nomeadamente: a abordagem centrada nos dados (*Data-Centric*) e a abordagem orientada aos serviços (*Service-Oriented*).

A figura 3.1 esquematiza os principais aspectos de cada uma destas abordagens.

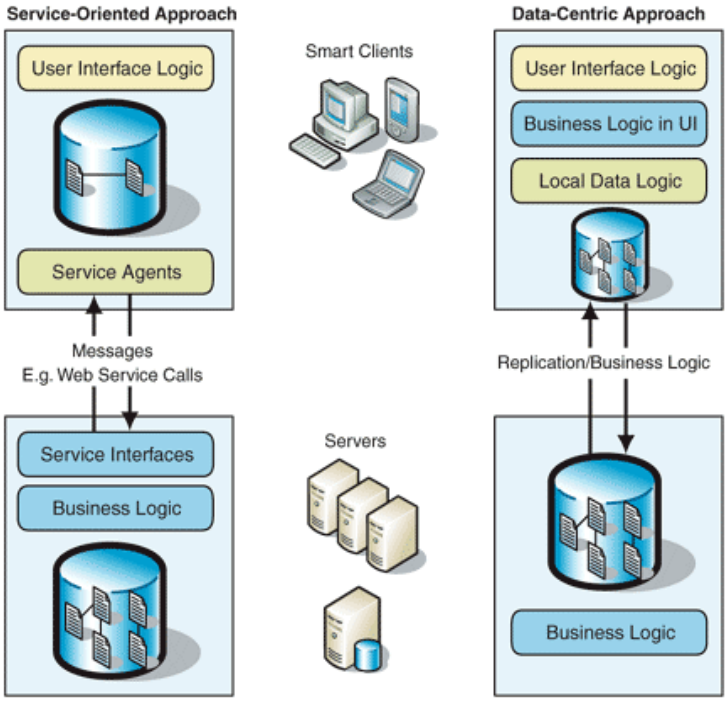


Figura 3.1 – Abordagem *Service-Oriented vs*. Abordagem *Data Centric*

Fonte: Smart Client Architecture and Design Guide. Microsoft, 2004

As aplicações que utilizam a estratégia centrada nos dados possuem um sistema gestor de base de dados relacional (SGBDR) instalado localmente no cliente tirando partido das capacidades fornecidas por este, para propagar as actualizações dos dados para o servidor, gerir o processo de sincronização e detectar e resolver os conflitos dessa actualização.

As aplicações que utilizam a abordagem orientada aos serviços, armazenam a informação em mensagens, organizando estas em filas, enquanto a aplicação está desconectada. Após o restabelecimento da conexão, as filas são enviadas para o servidor, efectuando o respectivo processamento.

3.2.1 Abordagem Centrada nos Dados (Data-Centric)

Tipicamente, quando se utiliza a abordagem centrada nos dados, o servidor publica a informação e o cliente cria uma subscrição dos dados que necessita, para que os possa copiar para o seu repositório local, durante os períodos em que está conectado. Enquanto desconectada a utilização da aplicação, provoca actualizações nesses dados locais. Quando a aplicação está novamente conectada, o repositório local propaga as alterações efectuadas para o servidor. Neste momento, as alterações ocorridas no servidor enquanto a aplicação esteve desconectada podem também ser propagadas para o cliente. Quaisquer conflitos ocorridos nesta fase de replicação (*merge-replication*), são tratados por regras de resolução de conflito especificadas no servidor ou no cliente e de acordo com a lógica de negócio que se pretendeu implementar.

Como as alterações dos dados ocorrem de forma autónoma no servidor e cliente, não são utilizadas transacções *ACID* (Atomicidade, Consistência, Isolamento, Durabilidade). Quando a replicação das alterações é feita, os subscritores passam a utilizar os dados existentes no publicador.

A principal vantagem da abordagem centrada nos dados reside no facto de todo o código necessário ao rastreamento das alterações e resolução de conflitos já estar contido no SGBDR. Normalmente inclui a detecção de conflitos tanto a nível da coluna como a nível do registo, validação dos dados e restrições. No modelo centrado nos dados, o sistema o sistema gestor de base de dados gere a sincronização. Os utilizadores definem quais as tabelas que requerem sincronização dos dados e o SGBD fornece a infra-estrutura necessária para detectar alterações e detectar e resolver os conflitos que surjam entre as operações de replicação. É também frequente que a essa infra-estrutura possam ser acrescentadas nas aplicações, funcionalidades personalizadas de detecção e resolução de conflitos.

Como neste modelo existe um único repositório de dados para todo o sistema, fica garantida a convergência dos dados entre os clientes e o servidor, após as operações de sincronização.

Existe porém a necessidade de disponibilizar uma forma de gerir a distribuição dos diversos conjuntos de dados de forma eficiente e escalável, principalmente quando existe um grande número de clientes. Mais uma vez o SGBD oferece, normalmente, as funcionalidades necessárias para se particionar estes conjuntos de dados de forma horizontal e vertical e até com regras dinâmicas.

3.2.1 Abordagem Centrada nos Dados (Data-Centric)

Existem algumas desvantagens em relação à abordagem centrada nos dados, que impõe a necessidade de um repositório local de dados, o qual pode não ser adequado em algumas situações, como por exemplo, se o dispositivo móvel for limitado (ex. *smartphone* ou telemóvel) ou se o mecanismo de distribuição da aplicação tiver de ser extremamente rápido. Outra desvantagem deste modelo centrado nos dados, diz respeito à forte ligação entre o repositório local e as tabelas da base de dados do servidor, significando que qualquer alteração no seu esquema tenha impacto directo na aplicação cliente.

Existem também vários benefícios, decorrentes da utilização de um repositório local, onde a replicação das actualizações fazem convergir os dados entre o cliente e o servidor, delegando no SGBD as tarefas de sincronização, detecção e correcção de conflitos. Esta abordagem centrada nos dados pode ser utilizada quando:

- O cliente suporta a criação da instância local de base de dados;
- Possa existir uma ligação forte entre o cliente e o servidor através de definições do modelo de dados e protocolos de comunicação;
- Pretende-se que já exista um mecanismo integrado de sincronização e detecção/resolução de conflitos dos dados;
- Os mecanismos de resolução de conflitos na reconciliação de dados, oferecidos pelo SGBD, minimizem de forma significativa a necessidade de desenvolvimento dessa funcionalidade;
- Não é requerido interagir com múltiplos serviços díspares.

3.2.2 Abordagem Orientada aos Serviços (Service-Oriented)

Com a abordagem orientada aos serviços (*Service-Oriented*) a aplicação cliente pode interagir com quaisquer serviços dispersos que necessite. O foco da aplicação está mais centrado nos pedidos a esses serviços do que nas alterações a um conjunto de dados locais para posterior propagação ao servidor. Como é óbvio, esses pedidos podem provocar alterações nos dados locais e nos dados do servidor, mas essas alterações devem ser vistas como meros subprodutos das chamadas aos serviços [Boulter, 2004].

Esta abordagem é baseada na troca de mensagens entre o cliente e o serviço remoto, com um esquema previamente estabelecido. Nesta comunicação o cliente não tem de conhecer os aspectos da implementação do serviço remoto e vice-versa, bastando que cumpram o esquema estabelecido.

Ao contrário da abordagem centrada nos dados, a estratégia orientada aos serviços, por adoptar mecanismos de comunicação *loosely-coupled*, permite reduzir a dependência entre o cliente e o serviço remoto.

A abordagem orientada aos serviços é particularmente útil para as aplicações que operam em ambientes de Internet e extranet, uma vez que é implementada recorrendo à utilização de *Web Services*, que habitualmente são disponibilizados através do protocolo *HyperText Transfer Protocol (HTTP)* [W3C, 1999].

Outra vantagem da estratégia orientada aos serviços é a não obrigatoriedade da existência de uma base de dados do lado da aplicação cliente, permitindo uma maior abrangência de dispositivos móveis.

A grande desvantagem desta abordagem é a necessidade de desenvolvimento de toda a infra-estrutura de armazenamento e reencaminhamento (*store-and-forwarding*) das mensagens relativas aos pedidos dos serviços, bem como a detecção do estado das ligações aos recursos de rede e consequente alteração do estado da aplicação ocasionalmente conectada.

A abordagem orientada aos serviços deve ser utilizada no desenho de aplicações ocasionalmente conectadas quando:

- É necessário maior controlo e flexibilidade sobre o processo de reconciliação de dados;
- A aplicação opera em ambientes de Internet e extranet;
- A aplicação cliente interage com múltiplos serviços díspares;
- É pretendido desacoplar o desenvolvimento da aplicação cliente e do serviço;
- A aplicação necessita de um esquema de segurança personalizado;
- O dispositivo cliente é muito limitado em termos de recursos.

3.3 Suporte da Smart Client Software Factory - SCSF

A *Smart Client Software Factory – SCSF* [MSFT, 2008] constitui um dos vários projectos disponibilizados pela equipa de *Patterns & Practices* [MSFT, 2009a] da Microsoft e representa um esforço em difundir recomendações que reflectem boas práticas no desenho de aplicações, utilizando a plataforma Microsoft. Este tipo de projectos, fornece um conjunto de documentação, código reutilizável e normalmente uma implementação de referência.

A *Smart Client Software Factory – SCSF*, não é excepção, disponibilizando os itens referidos, no sentido de proporcionar aos arquitectos e programadores de software a inclusão de práticas testadas e aceites, expostas através de padrões que aceleram o desenvolvimento e reduzem o risco dos projectos relativos à criação de C*omposite Smart Client Applications* [MSFT, 2009].

A *SCSF* disponibiliza um conjunto de blocos aplicacionais direccionados à resolução de vários problemas comuns nas aplicações, implementando os padrões e boas práticas nessa resolução.

No âmbito das aplicações ocasionalmente conectadas a *SCSF* fornece um suporte particularmente útil ao modo ocasionalmente conectado através de três blocos aplicacionais:

- *Connection Monitor Application Block*;
- *Endpoint Catalog Application Block*;
- *Disconnected Service Agent Application Block*.

3.3.1 O Bloco Aplicacional Connection Monitor

Este bloco aplicacional pode ser utilizado para detectar alterações do estado da conectividade às redes e fornece uma interface entre as aplicações e as funcionalidades de conexão existentes no computador que as executa. Vários blocos aplicacionais e funcionalidades da *SCSF* dependem dos serviços disponibilizados por este bloco, como é o caso do *Disconnected Service Agent Application Block.*

Para a maioria das aplicações ocasionalmente conectadas o benefício primário é a detecção de alterações nas redes lógicas. A rede lógica é definida pelo programador e representa um conjunto de recursos, como por exemplo *Web Services,* necessários ao funcionamento da aplicação. Com este bloco aplicacional é também possível monitorizar alterações na conectividade dos adaptadores físicos de rede.

No contexto da utilização deste bloco aplicacional, as redes lógicas não representam a rede definida ao nível do sistema operativo cliente, mas sim uma colecção de endereços para um conjunto de recursos necessários à aplicação. Cada rede lógica é definida por um nome e por um endereço representativo (*representative address*). O nome é utilizado para identificar a rede, ao passo que o endereço representativo é utilizado para determinar o estado da conectividade de todos os endereços da colecção.

A título de exemplo consideremos uma aplicação que consome um conjunto de *Web Services* existentes no domínio www.meireles.pt. O programador pode definir uma rede lógica chamada "Meireles" para o conjunto de *Web Services* e um endereço representativo http://www.meireles.pt que o bloco aplicacional utiliza para determinar o estado da conectividade dessa rede lógica e respectiva colecção de endereços.

Na definição das redes lógicas, para além da localização de serviços remotos, devem ser considerados os estados possíveis de conectividade do dispositivo cliente. Por exemplo, se uma dada aplicação acede a serviços remotos existentes quer na Internet, quer na rede local da empresa, os estados possíveis no dispositivo cliente podem ser os seguintes: conectado apenas à Internet; conectado apenas à rede local; conectado a ambas Internet e rede local; desconectado de ambas.

Neste caso, para tratar as diferentes situações, será necessário definir duas redes: uma para a rede local e outra para a Internet para que a aplicação possa interagir com os serviços num cenário em que apenas uma das redes esteja disponível.

Estrutura do Connection Monitor Application Block

O componente chave deste bloco aplicacional é a classe ***ConnectionMonitor*** que expõe colecções de instâncias das classes ***Connection*** e ***Network*** que representam as conexões e redes correntemente configuradas. O diagrama da figura 3.2 representa as principais classes deste bloco aplicacional.

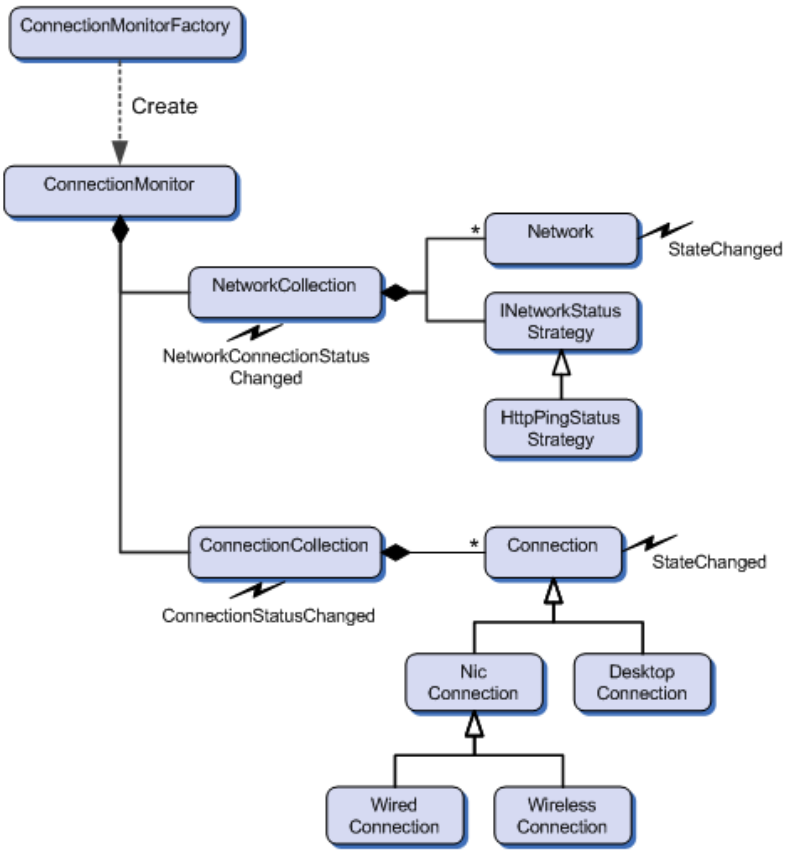


Figura 3.2 – Estrutura do *Connection Monitor Application Block*

Fonte: Smart Client Software Factory - April 2008. Microsoft Corporation, 2008

Monitorização das redes lógicas

A classe ***ConnectionMonitor*** expõe uma instância da ***NetworkCollection*** através da sua propriedade ***Networks***. Por sua vez, a classe ***NetworkCollection*** expõe uma lista genérica de instâncias de ***Network*** cuja chave é o nome dessa rede.

Para determinar o estado das redes, a ***NetworkCollection*** possui uma implementação da interface ***INetworkStatusStrategy***, que declara um método designado ***IsAlive***. O bloco aplicacional inclui uma implementação pré-definida da interface ***INetworkStatusStrategy*** chamada ***HttpPingStatusStrategy***. Esta estratégia recorre ao

envio de um comando "HTTP GET" para um hospedeiro especificado e analisa a resposta para determinar o estado da conectividade.

A classe ***Network*** representa a rede lógica e expõe o nome da rede e endereço. Expõe ainda a propriedade ***Connected*** que pode ser utilizada para verificar o estado de conectividade de uma rede específica. O evento ***StateChanged*** pode ser utilizado para detectar transições de estado na conectividade da rede.

Monitorização das conexões

A classe base abstracta ***Connection*** representa um adaptador físico de rede do computador local. O bloco aplicacional inclui implementações concretas para os seguintes tipos de conexões:

- ***NicConnection*** – representa qualquer interface de rede, por exemplo, uma placa *Ethernet* ou um adaptador *wireless*. Esta classe fornece a capacidade de detectar se a aplicação tem ou não conectividade com uma rede, independentemente do seu tipo. O bloco aplicacional possui ainda duas classes que derivam de *NicConnection*, para poder determinar o estado de tipos de conexões mais específicos, nomeadamente:
    - ***WiredConnection*** – representa uma ligação com cabo e adaptador *Ethernet*.
    - ***WirelessConnection*** - representa uma ligação sem fios.
- ***DesktopConnection*** – representa uma ligação a serviços locais e cujo estado pode ser actualizado programaticamente.

A classe ***ConnectionMonitor*** expõe uma instância da ***ConnectionCollection*** através da sua propriedade ***Connections***. Por sua vez, a classe ***ConnectionCollection*** expõe uma lista genérica de instâncias de ***Connection*** cuja chave é o nome dessa conexão.

Todas as classes de conexão específicas que derivam da classe abstracta ***Connection*** expõem as propriedades, métodos e eventos desta, que permitem determinar o tipo de conexão e preço (especificados no ficheiro de configuração da aplicação), bem como testar o estado da conexão utilizando a propriedade ***IsConnected***. Existem igualmente métodos para obter e actualizar informação sobre o estado da conexão. O evento ***StateChanged*** pode ser subscrito para monitorizar transições no estado da conexão.

Monitorização das conexões (cont.)

O bloco aplicacional permite ainda conhecer o custo relativo de cada conexão, conforme definido no ficheiro de configuração da aplicação, por tipo de ligação.

A classe ***Connection*** expõe uma propriedade designada ***Price***, para obter e especificar o custo de cada conexão, permitindo à aplicação e aos utilizadores o controlo dos pedidos com o conhecimento e implicações desta informação. Se o custo da conexão não for relevante para a aplicação, as conexões podem ser definidas com o "*Price*" igual a zero.

Configuração

Para utilizar o *Connection Manager Application Block* é necessário adicionar informação de configuração (ao ficheiro de configuração da aplicação), sobre as redes lógicas e adaptadores físicos de rede que pretendemos monitorizar.

Com o seguinte exemplo de configuração, podemos observar que o bloco aplicacional irá monitorizar duas redes lógicas e um adaptador físico de rede do tipo *Ethernet*.

```
<ConnectionMonitor>
  <Networks>
    <add Name="OfficeLan" Address="http://intranet"/>
    <add Name="Internet" Address="http://www.meireles.pt"/>
  </Networks>
  <Connections>
    <add Type="WiredConnection" Price="1"/>
  </Connections>
</ConnectionMonitor>
```

O atributo ***Type*** relativo ao adaptador físico de rede pode ter os valores: *NicConnection*; *WiredConnection*; *WirelessConnection* e *DesktopConnection*. Adicionalmente pode ser especificado o nome qualificado de outra classe que implemente de forma personalizada as funcionalidades requeridas. De forma semelhante, o atributo ***StrategyType*** do nó ***<Networks>*,** pode referenciar o nome qualificado de uma classe que implemente a interface ***INetworkStatusStrategy*** para definir uma estratégia personalizada na monitorização do estado de uma rede lógica. Se este atributo for omitido, será utilizada a estratégia pré-definida ***HttpPingStatusStrategy***.

Utilização

Para simplificar a criação de instâncias de ***ConnectionMonitor***, este bloco aplicacional inclui a classe ***ConnectionMonitorFactory*** que expõe o método estático ***CreateFromConfiguration.*** Este método recebe como argumento um valor do tipo *string* especificando o nome da secção (do ficheiro de configuração da aplicação) que contém a lista de conexões e custo das mesmas a partir da qual é populada a instância de ***ConnectionMonitor***. Se este argumento for omitido, é utilizado o nome de secção pré-definido "*ConnectionMonitor*". Segue-se um exemplo em C# das duas formas referidas:

```
ConnectionMonitor sampleMonitor  =
   ConnectionMonitorFactory.CreateFromConfiguration();
```

```
String configurationName = "SampleConnectionSection";
ConnectionMonitor sampleMonitor  =
ConnectionMonitorFactory.CreateFromConfiguration(configurationName);
```

Para obter uma lista das conexões configuradas a partir da propriedade ***Connections*** do objecto ***ConnectionMonitor*** pode usar-se:

```
ConnectionCollection connList = sampleMonitor.Connections;
```

Dado que a classe ***ConnectionCollection*** deriva de ***KeyedCollection*** que por sua vez implementa a interface ***IEnumerable*** é possível iterar a colecção com a instrução *foreach*:

```
StringBuilder builder = new StringBuilder();
foreach (Connection conn in connList)
{
  builder.Append(String.Format("Type: {0}, Price: {1}, "
                             + "IsConnected: {2}",
                                conn.ConnectionTypeName,
                                conn.Price.ToString(),
                                conn.IsConnected.ToString()));
  builder.Append("\n");
}
```

De forma semelhante às conexões, é possível obter uma lista das redes lógicas configuradas a partir da propriedade ***Networks*** do objecto ***ConnectionMonitor***:

```
NetworkCollection netList = sampleMonitor.Networks;
```

Utilização (cont.)

Dado que a classe ***NetworkCollection*** deriva de ***KeyedCollection*** que por sua vez implementa o interface ***IEnumerable*** é possível iterar a colecção com a instrução *foreach*:

```
StringBuilder builder = new StringBuilder();
foreach (Network netwk in netList)
{
  builder.Append(String.Format("Name: {0}, Connected: {1}",
                 netwk.Name, netwk.Connected.ToString()));
  builder.Append("\n");
}
```

Para obter uma lista apenas das redes que estão conectadas, pode-se utilizar a propriedade ***ActiveNetworks*** da classe ***NetworkCollection***:

```
ReadOnlyCollection<Network> activeNetworks =
   sampleMonitor.Networks.ActiveNetworks;
```

Ambas as classes ***Network*** e ***Connection*** desencadeiam o evento ***StateChanged*** quando o seu estado muda. Este evento pode ser subscrito e criado um *handler* que aceite uma instância da classe ***StateChangedEventArgs***, que expõe a propriedade ***IsConnected***:

```
conn.StateChanged += StateChangeHandler;
netwk.StateChanged += StateChangeHandler;
```

```
public void StateChangeHandler(object sender, StateChangedEventArgs e)
{
  if (e.IsConnected)
      MessageBox.Show("Now connected");
  else
      MessageBox.Show("Now disconnected");
}
```

A classe ***Connection*** expõe o método ***UpdateStatus*** que pode ser utilizado para desencadear o evento ***StateChanged*** e obter informação através do *handler* especificado na subscrição do evento:

```
conn.UpdateStatus();
```

Utilização (cont.)

Para criar uma nova instância de ***Connection***, é invocado o construtor da classe concreta (neste exemplo uma *NicConnection*) e especificar o nome e o custo dessa conexão:

```
NicConnection conn = new NicConnection("NicConnection", 6);
```

A conexão criada pode então ser adicionada ao objecto ***ConnectionMonitor*** com a propriedade ***Connections*** através do método ***Add*** exposto pela classe ***ConnectionCollection***:

```
sampleMonitor.Connections.Add(conn);
```

Podem ainda ser utilizados métodos da classe ***KeyedCollection*** (da qual a classe ***ConnectionCollection*** deriva) para determinar a presença de uma conexão específica, obter uma referência para ela e removê-la da ***ConnectionCollection***:

```
if (sampleMonitor.Connections.Contains("Internet"))
{
  Connection conn = sampleMonitor.Connections["Internet"];
  if (!conn.IsConnected)
  {
    MessageBox.Show("Removing connection with price " +
conn.Price.ToString());
    sampleMonitor.Connections.Remove(conn);
  }
}
```

Os mesmos princípios e mecanismos aplicam-se às classes ***Network*** e ***NetworkCollection***.

Para criar uma instância:

```
Network network = new Network("Intranet", "http://intranet");
```

Adicionar à colecção:

```
sampleMonitor.Networks.Add(network);
```

Manipular a colecção:

```
if (sampleMonitor.Networks.Contains("Intranet"))
{
    Network netwk = sampleMonitor.Networks["Intranet"];
    if (!netwk.Connected)
      sampleMonitor.Networks.Remove(netwk);
}
```

3.3.2 O Bloco Aplicacional Endpoint Catalog

Este bloco aplicacional representa um serviço simples que utiliza um dicionário para armazenar individualmente "*endpoints*" e respectivas credenciais. Um *endpoint* consiste num nome, opcionalmente um endereço e credencial pré-definidos, e uma série itens de rede. Cada item de rede é composto por um nome, endereço e credencial necessária para ligar o *endpoint* através dessa rede. A credencial é composta por dois ou três valores *string*: nome de utilizador; palavra passe; e opcionalmente o nome de domínio de autenticação.

O seguinte exemplo representa a secção de configuração ***<Endpoints>***, do ficheiro de configuração da aplicação:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  ...
  <Endpoints>
    <EndpointItems>

      <add Name="my-host-name" Address="http://default-host.com/path"
           UserName="user-name" Password="password">
        <NetworkItems>
          <add Name="Internet" Address="http://internet-host.com/path"
               UserName="user-name" Password="password" />
          <add Name="Work" Address="http://work-host.com/path"
               UserName="user-name" Password="password" Domain="domain-name" />
        </NetworkItems>
      </add>

      <add Name="host-name-no-default">
        <NetworkItems>
          <add Name="Internet" Address="http://internet-host.com/path"
               UserName="user-name" Password="password" />
          <add Name="Work" Address="http://work-host.com/path"
               UserName="user-name" Password="password" Domain="domain-name" />
        </NetworkItems>
      </add>

    </EndpointItems>
  </Endpoints>

</configuration>
```

Neste caso temos representada a configuração de dois *endpoints* em que o primeiro, para além do nome, tem um endereço e credenciais pré-definidas.

Estrutura do *Endpoint Catalog Application Block*

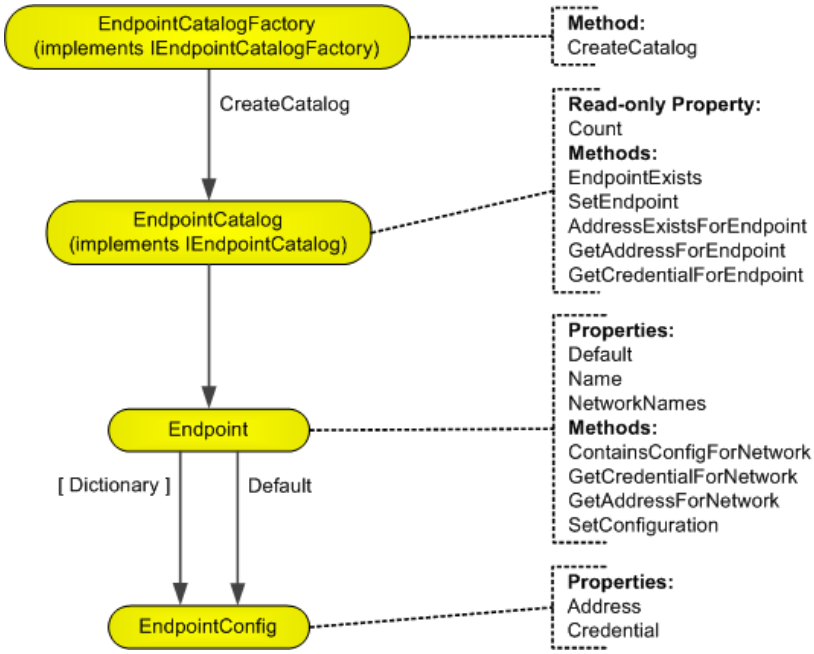


Figura 3.3 – Estrutura do *Endpoint Catalog Application Block*

Fonte: Smart Client Software Factory - April 2008. Microsoft Corporation, 2008

A classe ***EndpointCatalog*** (classe principal do bloco aplicacional ***EndpointCatalog***) utiliza a classe ***Dictionary*** para armazenar e expor individualmente uma série de instâncias da classe ***Endpoint*** através de um valor chave que é o respectivo nome.

Para preencher a ***Dictionary*** com a informação armazenada no ficheiro de configuração da aplicação, é utilizado o método ***CreateCatalog*** da classe ***EndpointCatalogFactory***. Este método que simplifica a criação de instâncias de ***EndpointCatalog***, utiliza as classes de configuração existentes na subpasta "*Configuration*" deste bloco aplicacional, para aceder à informação armazenada na secção de configuração.

A classe ***EndpointCatalog*** expõe a propriedade ***Count*** e um conjunto de métodos que permitem: a verificação da existência de um *endpoint*; se um dado *endpoint* tem endereço; obter o endereço e credencial pré-definidos para o *endpoint*.

A classe ***Endpoint*** expõe o nome do *endpoint* e uma lista contendo os nomes de todas as redes para esse *endpoint*. Fornece também um conjunto de métodos que permitem verificar a existência de uma rede específica para o *endpoint* e obter o endereço e credencial para uma dada rede. Esta classe pode ter opcionalmente um endereço e credencial pré-definidos, utilizados no caso do item de rede não especificar

essa informação. Para o efeito, cada instância de ***Endpoint*** expõe, através da sua propriedade ***Default***, uma instância da classe ***EndpointConfig***, expondo a informação através de duas propriedades: ***Address*** e ***Credential***.

Configuração

Para utilizar o *Endpoint Catalog Application Block* é necessário adicionar uma secção de configuração no ficheiro de configuração da aplicação:

```
<configSections>
  <section name="Endpoints"
type="Microsoft.Practices.SmartClient.EndpointCatalog.Configuration.EndpointSection,
Microsoft.Practices.SmartClient.EndpointCatalog" />
</configSections>
```

Com o seguinte exemplo de configuração, podemos observar que estão configurados dois *endpoints* identificados pelos nomes "ws1" e "ws2", ambos com informação de endereço e credencial pré-definidas. O primeiro pode ser utilizado através de duas redes (identificadas pelos nomes "Internet" e "Work") e o segundo apenas por uma ("Internet"). Cada uma destas redes tem configurado o endereço e a credencial necessária para o acesso:

```
<Endpoints>
  <EndpointItems>
    <add Name="ws1" Address="http://default-host.com/WebService1.asmx"
UserName="default-user-name" Password="default-password" Domain="default-
domain">
      <NetworkItems>
        <add Name="Internet" Address="http://internet-
host.com/WebService1.asmx"
               UserName="internet-user-name" Password="internet-password" />
        <add Name="Work" Address="http://work-host.com/WebService1.asmx"
               UserName="work-user-name" Password="work-password"  />
      </NetworkItems>
    </add>
    <add Name="ws2" Address="http://default-host.com/WebService2.asmx"
UserName="default-user-name" Password="default-password" Domain="default-
domain">
      <NetworkItems>
        <add Name="Internet" Address="http://internet-
host.com/WebService2.asmx"
               UserName="internet-user-name" Password="internet-password" />
      </NetworkItems>
    </add>
  </EndpointItems>
</Endpoints>
```

Como é óbvio, informação sensível como a das credenciais deverá ser protegida, recorrendo à encriptação da secção de configuração. Para o efeito podem ser usadas as classes de configuração protegida, existentes no espaço de nomes "*System.Configuration*", como por exemplo a classe ***SectionInformation*** que expõe o método ***ProtectSection***.

Utilização

Para a utilização deste bloco aplicacional começa-se por criar uma nova instância da classe ***EndpointCatalogFactory*,** especificando no seu construtor o nome da secção (do ficheiro de configuração da aplicação) que contém a informação sobre os *endpoints*. A instância pode ser guardada como uma interface ***IEndpointCatalogFactory***, no caso de desejarmos estender o bloco aplicacional através da criação de uma classe personalizada que implemente essa interface:

```
String configName = "Endpoints";
IEndpointCatalogFactory catalogFactory
                         = new EndpointCatalogFactory(configName);
```

Com o método ***CreateCatalog*** da ***EndpointCatalogFactory*** é criada uma instância da classe ***EndpointCatalog***, que contém os detalhes de todos os *endpoints* configurados para a aplicação:

```
IEndpointCatalog catalog = catalogFactory.CreateCatalog();
```

Pode verificar-se se um dado *endpoint* existe:

```
String epName = "MyWebServiceEndpoint";
if (catalog.EndpointExists(epName))
{
  // Endpoint existe.
}
```

E pode obter-se o número de *endpoints* existentes:

```
int endpointCount = catalog.Count;
```

Neste exemplo, verifica-se se o endereço “MyWebServiceEndpoint” existe na rede identificada com o nome “MyHost”, e no caso de existir, podem-se obter os valores da respectiva credencial:

```
String epName = "MyWebServiceEndpoint";
String epNetworkName = "MyHost";
if (catalog.AddressExistsForEndpoint(epName, epNetworkName))
{
  String epAddress
       = catalog.GetAddressForEndpoint(epName, epNetworkName);
  NetworkCredential epCredentials
        = catalog.GetCredentialForEndpoint(epName, epNetworkName);
  String epUsername = epCredentials.UserName;
  String epPassword = epCredentials.Password;
  String epDomain = epCredentials.Domain;
}
```

3.3.3 O Bloco Aplicacional Disconnected Service Agent

Este bloco aplicacional tira partido dos dois blocos anteriormente descritos, (*Connection Monitor Application Block* e *Endpoint Catalog Application Block*) e fornece um conjunto de funcionalidades particularmente direccionadas para a execução de Web *Services* em aplicações ocasionalmente conectadas. Com este agente de serviço desconectado é possível à aplicação manter uma fila dos pedidos de execução de *Web Services* efectuados enquanto o dispositivo está desconectado, para posteriormente os executar se uma dada rede para o recurso ficar disponível.

Existem duas funcionalidades distintas neste bloco aplicacional:

- Um subsistema que permite invocar *Web Services* utilizando padrões semelhantes para quem está familiarizado com a utilização de *Web Services proxies*. Porém, o agente de serviço desconectado cria filas dos pedidos e mantém a rastreabilidade necessária para as respostas das chamadas assíncronas que a aplicação espera receber aquando da resposta do *Web Service*.
- Um *dispacher* que determina exactamente quando é que deve processar cada chamada a um *Web Service* levando em conta as condições de conectividade e custo das ligações. Quando a condição correcta ocorre, o *dispacher* efectua a chamada ao *Web Service*, utilizando os endereços e credenciais apropriados para o *endpoint*.

Estrutura do *Disconnected Service Agent Application Block*

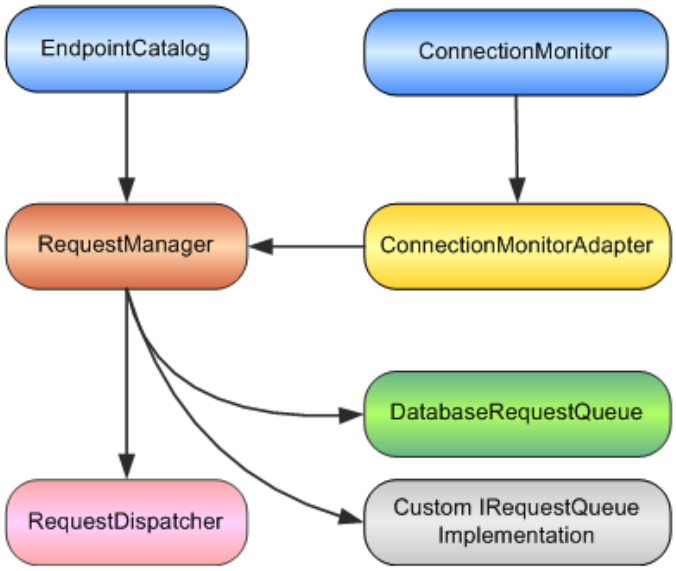


Figura 3.4 – Estrutura do *Disconnected Service Agent Application Block*

Fonte: Smart Client Software Factory - April 2008. Microsoft Corporation, 2008

O *Disconnected Service Agent Application Block* é composto por um conjunto de classes principais, necessárias à utilização dos serviços fornecidos por este bloco aplicacional.

A classe ***RequestManager*** gere a fila de pedidos e utiliza os serviços disponibilizados pela classe ***RequestDispatcher*** para enviar os mesmos. A fila de pedidos pendentes e de pedidos falhados é armazenada do lado cliente, numa base de dados *SQL Server Compact Edition*, recorrendo à utilização da classe ***DatabaseRequestQueue***, ou através de uma implementação personalizada da interface ***IRequestQueue***.

A classe ***Request*** constitui um encapsulamento do pedido ao *Web Service*, incluindo os argumentos necessários ao método do *Web Service*, o *endpoint* para entrega do pedido e propriedades que especificam como o *dispatcher* deve tratar e entregar o pedido. A classe ***EndpointCatalog*** (do *Endpoint Catalog Application Block*) expõe a informação dos *endpoints* existente no ficheiro de configuração da aplicação.

Estrutura do *Disconnected Service Agent Application Block* (cont.)

A classe ***ConnectionMonitorAdapter*** fornece informação acerca da ligação utilizada pelo pedido (como por exemplo, o custo e detalhes das redes) e desencadeia eventos quando o estado da ligação muda. Como a própria palavra "*Adapter*" sugere, esta classe serve de ligação e conversão necessárias entre as classes ***RequestManager*** e ***ConnectionMonitor*** (do Connection *Monitor Application Block*), sendo a última responsável por monitorizar e expor as conexões e redes lógicas existentes.

A classe ***RequestManager*** controla o envio dos pedidos efectuados aos *Web services*, retirando as mensagens da fila e enviando-as quando a aplicação está conectada. Para manter a integridade dos pedidos e assegurar o correcto desempenho quando a aplicação não tem uma conexão activa, os pedidos são mantidos na base de dados até serem submetidos no servidor remoto ou até estes expirarem (devido à especificação de um tempo ou número máximo de tentativas). Desta forma a fila de pedidos sobrevive a eventuais perdas de conectividade.

Esta classe fornece funcionalidades que permitem o controlo do serviço de envio, expondo métodos para começar e parar o envio automático de pedidos, enviar todos os pedidos pendentes ou enviar individualmente cada um destes.

A classe ***RequestManager*** é um *singleton* e não possui um construtor pré-definido. O acesso à instância desta classe é efectuado através da sua propriedade ***Instance*** e chamando o método ***Initialize*** com uma série de parâmetros, nomeadamente: a fila para pedidos pendentes e falhados; a referência para uma classe que implemente a interface ***IConnectionMonitor*** (como é o caso da classe ***ConnectionMonitorAdapter***); uma referência para uma classe que implemente a interface ***IRequestDispatcher*** (como é o caso da classe ***RequestDispatcher***); uma instância da classe ***EndpointCatalog*** que contém a informação dos *endpoints* a utilizar pelos pedidos.

As filas que contêm os pedidos são instâncias de uma classe que implemente a interface ***IRequestQueue***. O *Disconnected Service Agent Application Block* contém as classes ***DatabaseRequestQueue*** que armazena os pedidos numa base de dados local e ***MemoryRequestQueue*** que os armazena em memória.

Estrutura do *Disconnected Service Agent Application Block* (cont.)

O diagrama da figura 3.5 representa o subsistema ***RequestManager*** indicando as propriedades, métodos e eventos expostos pelas diversas classes.

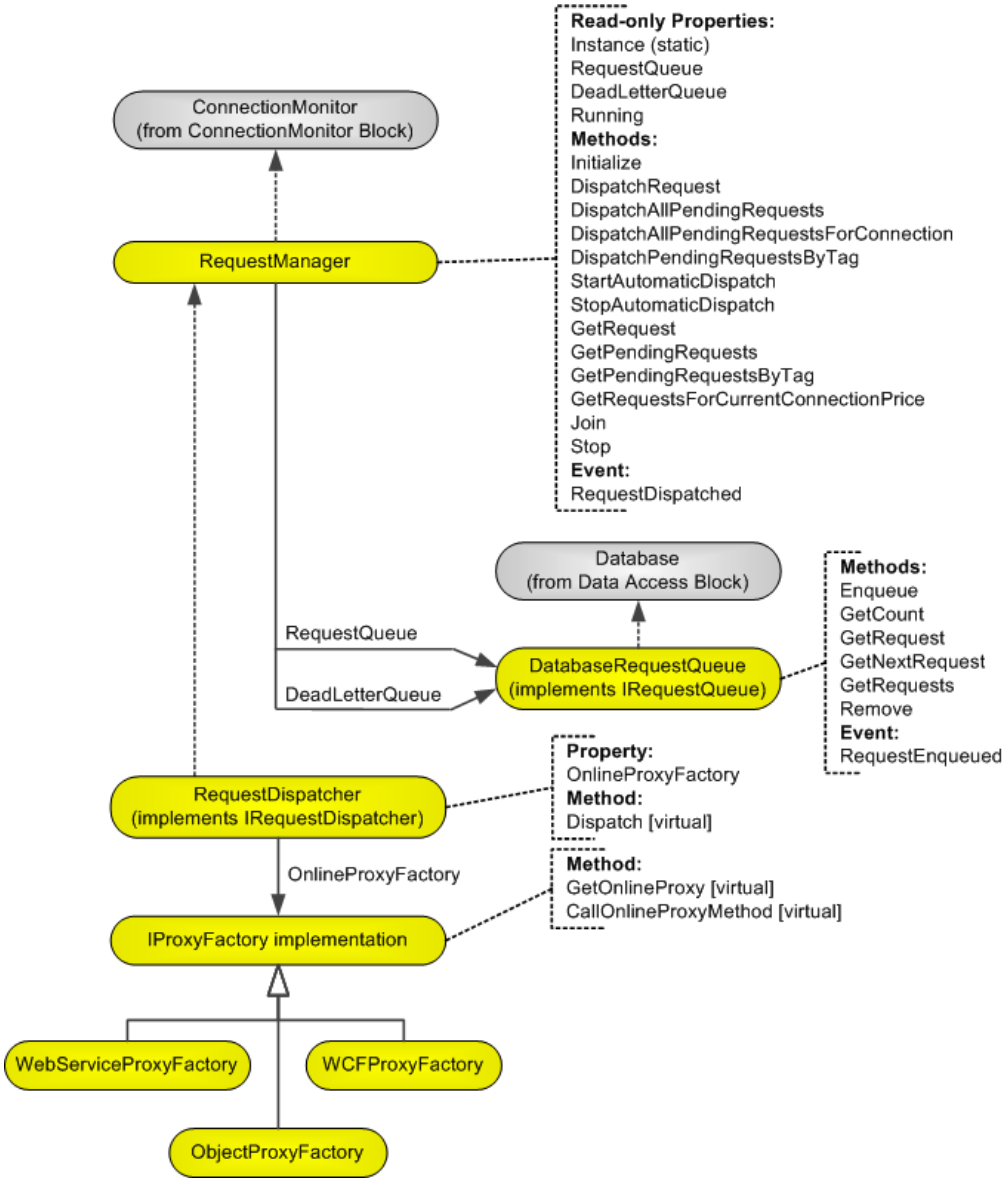


Figura 3.5 – Estrutura do subsistema *RequestManager*

Fonte: Smart Client Software Factory - April 2008. Microsoft Corporation, 2008

Estrutura do *Disconnected Service Agent Application Block* (cont.)

A classe ***Request*** e outras associadas fornecem as funcionalidades necessárias para criar pedidos a enviar a um *Web Service* remoto. A classe ***Request*** expõe propriedades que definem o comportamento, parâmetros (argumentos do *Web Service*), *endpoints* e outros detalhes de um pedido. A maioria dos pedidos a *Web Services* necessitam que os argumentos sejam enviados como parte integrante do pedido. Para o efeito esta classe expõe a propriedade ***CallParameters*** à qual se pode atribuir um *array* de objectos contendo os argumentos para o método do Web *service*. As classes ***CallParameters*** e ***CallParametersSerializer*** possuem métodos que a ***DatabaseRequestQueue*** utiliza para automaticamente serializar e deserializar o *array* de argumentos.

Os pedidos para um serviço remoto necessitam de detalhes do *endpoint*, como por exemplo a rede, endereço e credenciais. A propriedade ***Endpoint*** da classe ***Request*** expõe o nome de uma instância de ***Endpoint*** exposta pelo *Endpoint Catalog application block*.

O diagrama da figura 3.6 representa uma visão esquemática da classe ***Request*** e outras associadas, indicando as propriedades, métodos e eventos expostos.

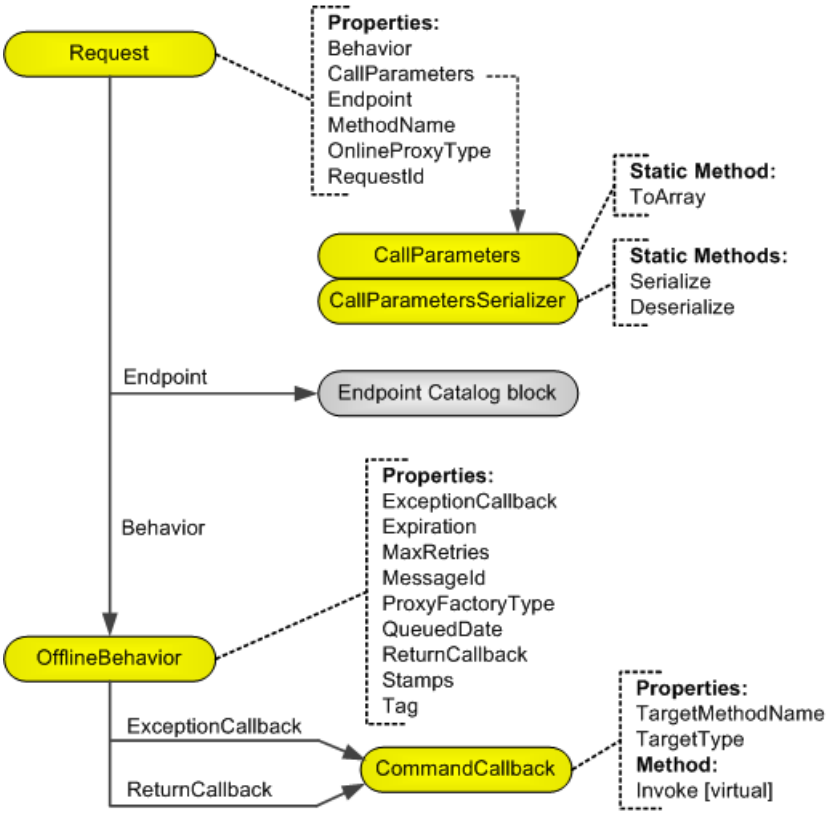


Figura 3.6 – Estrutura da classe *Request*

Fonte: Smart Client Software Factory - April 2008. Microsoft Corporation, 2008

Estrutura do *Disconnected Service Agent Application Block* (cont.)

Por fim, a classe ***ConnectionMonitorAdapter*** fornece um adaptador para o *Connection Monitor Application Block* e expõe informação acerca da ligação, permitindo testar se a aplicação está conectada, obter o nome da rede e o custo da conexão. Esta classe recebe uma instância de ***ConnectionMonitor*** no seu construtor.

A classe ***ConnectionMonitorAdapter*** expõe ainda eventos que permitem tratar situações relacionadas com a mudança de estado da conexão, como por exemplo, o evento ***ConnectionStatusChange.*** Outro exemplo é o evento ***ActiveNetworkChangedEvent*** que ocorre quando o dispositivo muda a sua ligação para outra rede.

No diagrama da figura 3.7 representa-se uma visão esquemática da classe ***ConnectionMonitorAdapter***, indicando as propriedades, métodos e eventos expostos.

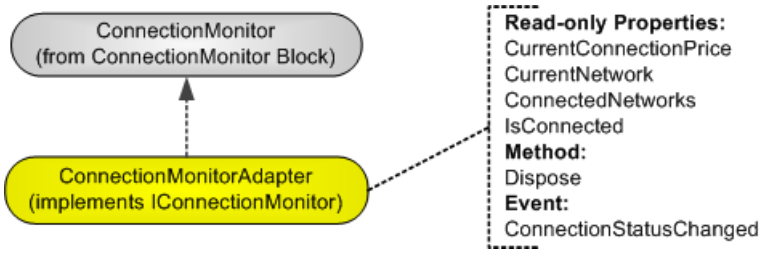


Figura 3.7 – Estrutura da classe *ConnectionMonitorAdapter*

Fonte: Smart Client Software Factory - April 2008. Microsoft Corporation, 2008

Capítulo 4

Requisitos

Este capítulo tem por objectivo principal descrever os diversos aspectos relacionados com os requisitos do sistema que se pretende conceber e está dividido em três partes principais. Na primeira, são efectuadas breves considerações sobre conceitos associados ao próprio requisito. A segunda parte, que possui uma estrutura inspirada na norma IEEE Std 830-1998 [IEEE, 1998b] para documentação de especificação de requisitos, visa descrever o âmbito e objectivos principais e realizar uma descrição geral do sistema pretendido, com particular ênfase na sua arquitectura. Finalmente, na terceira parte, é criada uma lista de requisitos específicos funcionais e não funcionais, devidamente identificados e agrupados, que servirão para orientar as fases seguintes de implementação e verificação dos resultados.

4.1 O Requisito – definição, processo e especificação

Numa definição muito simples, um requisito pode ser definido como "algo que um cliente necessita", ou na perspectiva de quem constrói como "algo que necessita ser desenhado" [Macaulay, 1996].

Um requisito do sistema define uma funcionalidade ou característica considerada relevante na óptica do utilizador. Representa o comportamento esperado do sistema, que na prática consiste num serviço que deve ser disponibilizado ao utilizador. [Jacobson, Booch e Rumbaugh, 1999]

Segundo Bennet, McRobb e Farmer (1999), os requisitos podem ser classificados em três categorias:

- Requisitos funcionais – que descrevem o que um sistema faz ou é esperado que faça. Estes requisitos são inicialmente levantados, abrangendo a descrição de processamentos a efectuar pelo sistema, entradas e saídas de informação que resultam da interacção com utilizadores ou outros sistemas.
- Requisitos não funcionais – estão relacionados com características qualitativas do sistema, descrevendo a qualidade com que o sistema deve fornecer os requisitos funcionais. Abrangem, por exemplo, medidas de desempenho, volume de dados e considerações de segurança.
- Requisitos de facilidade de utilização – tentam garantir que existirá uma boa ligação entre o sistema desenvolvido, utilizadores e tarefas que desempenham apoiadas no sistema.

O processo de engenharia de requisitos comporta um conjunto estruturado de actividades levadas a cabo para criar e manter o documento de requisitos de um sistema. Um modelo de alto nível destas actividades inclui: a identificação; análise e negociação; validação e documentação de requisitos [Kotonya e Sommerville, 1998].

Sobre o aspecto formal da produção do documento de especificação de requisitos de um sistema, importa referir como boa prática a norma IEEE Std 830-1998 *Recommended Practice for Software Requirements Specifications* [IEEE, 1998b].

Os próximos pontos deste capítulo irão ser baseados em parte da estrutura apresentada nessa norma, nomeadamente:

- Âmbito
- Descrição geral que engloba: Perspectiva do produto / Arquitectura; Interfaces; Funcionalidades do produto; Características dos utilizadores; Restrições; Pressupostos e dependências
- Requisitos específicos

4.2 Âmbito

4.2.1 Descrição

O âmbito deste projecto diz respeito à concepção, desenvolvimento e implementação de uma plataforma de suporte à execução de funcionalidades existentes nas aplicações de sistemas *ERP* Baan, por aplicações móveis externas a este. Dada a natureza das aplicações e dos dispositivos móveis é esperado que a plataforma forneça um conjunto de funcionalidades que possibilitem às aplicações móveis a resposta aos seus diversos desafios: funcionamento das aplicações com o dispositivo conectado ou desconectado da rede; assegurar a consistência e durabilidade das transacções, após as operações de sincronização entre os dispositivos móveis e o sistema *ERP*; garantir as questões de segurança, fiabilidade e desempenho envolvidas em todo o processo.

Este projecto não contempla o desenvolvimento das aplicações móveis. Porém, para termos uma ideia de um cenário possível da interligação das aplicações móveis com o sistema *ERP* Baan, apresenta-se o seguinte esquema da figura 4.1.

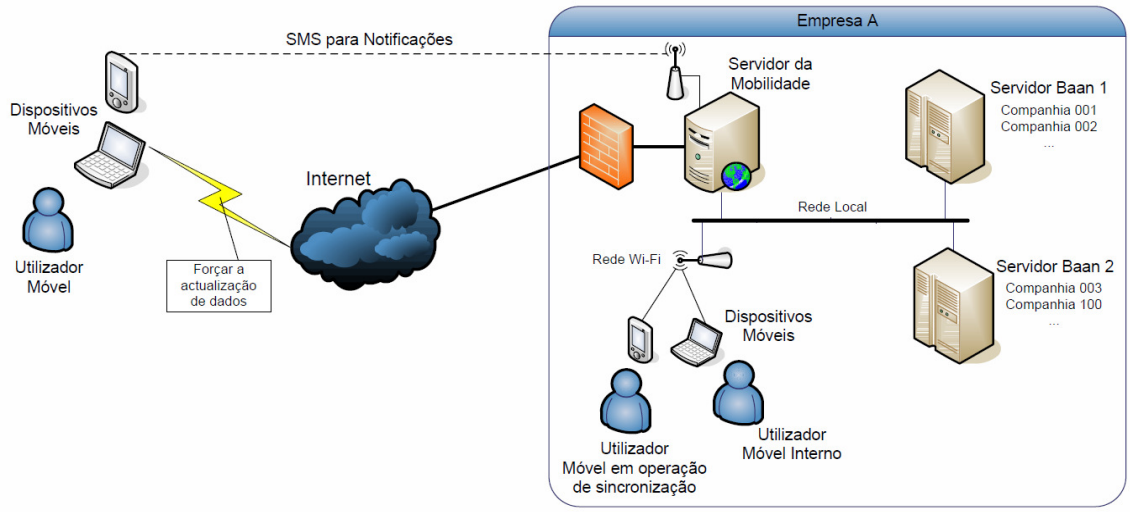


Figura 4.1 – Cenário típico de uma solução de mobilidade

Dado este cenário típico, o âmbito deste projecto centra-se apenas no que será necessário existir no servidor da mobilidade e no servidor do sistema *ERP* Baan para que uma aplicação móvel, em modo ocasionalmente conectado, possa executar uma funcionalidade existente no sistema *ERP*, como por exemplo, inserir uma ordem de venda ou declarar materiais e horas gastas numa ordem de serviço.

4.2.2 Objectivos

O desenvolvimento e colocação em serviço desta plataforma tem os seguintes objectivos:

- Permitir que aplicações externas aos sistemas *ERP* Baan IV, 5.0c e LN 6.1, possam executar funcionalidades destes, privilegiando o modo ocasionalmente conectado;
- Aumentar a eficiência e eficácia dos processos de negócio dos sectores que possam beneficiar da posterior implementação de aplicações móveis, como por exemplo, o comercial e a assistência técnica;
- Uniformização e automatização de processos e aplicações proporcionada pela maior cobertura do sistema *ERP*;
- Permitir uma maior facilidade de integração, no futuro desenvolvimento de aplicações externas ao sistema *ERP*.

4.2.3 Benefícios Esperados

- Tangíveis
    - Redução de custos inerente à uniformização e automatização de processos.
    - Redução de custos com o suporte do fornecedor *ERP* Baan.
    - Optimização da utilização do sistema *ERP*.

- Intangíveis
    - Desenvolvimento do conhecimento (*know-how*) interno.
    - Melhoria na qualidade dos dados.
    - Maior satisfação dos utilizadores.

4.3 Descrição Geral

4.3.1 Perspectiva do produto / Arquitectura

Apresentamos neste ponto a arquitectura concebida para permitir que aplicações móveis, externas aos sistemas *ERP* Baan, possam executar funcionalidades das aplicações destes, privilegiando o modo ocasionalmente conectado.

O diagrama que se apresenta na figura 4.2 esquematiza os principais blocos da arquitectura proposta.

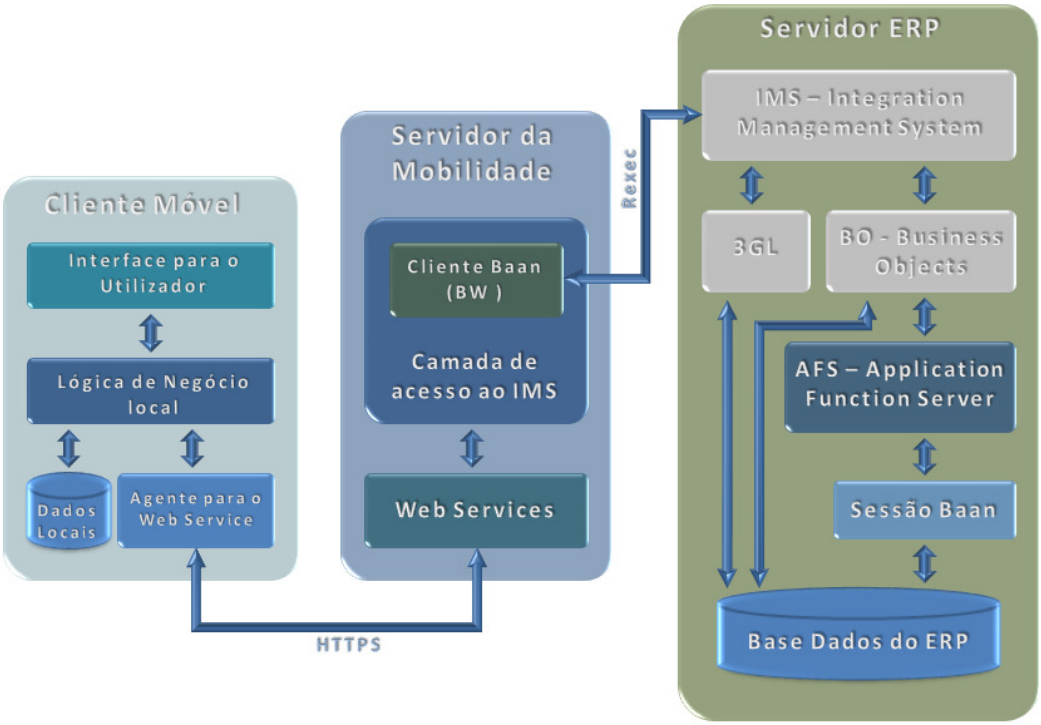


Figura 4.2 – Principais blocos da arquitectura proposta

Dado o objectivo, foi concebida uma arquitectura consistente com sistemas do tipo *loosely-coupled*[1], ou seja, a comunicação entre a parte cliente e a parte remota é baseada na troca de mensagens com um esquema previamente estabelecido. Neste tipo de sistemas o cliente não tem de conhecer os aspectos da implementação do serviço remoto e vice-versa, basta que ambos cumpram o esquema de mensagem definido para poderem comunicar.

1 – *Coupled* – termo que define o grau de dependência entre dois componentes num sistema distribuído [Boulter, 2004].

4.3.1 Perspectiva do produto / Arquitectura (cont.)

Do lado do servidor *ERP* Baan, a plataforma é composta por módulos a serem desenvolvidos neste trabalho, como é o caso do principal *Integration Management System - IMS* e por módulos existentes como é o caso do *Application Function Server – AFS* (já descrito na secção 2.2.4).

O bloco *IMS - Integration Management System* representa o módulo principal desta plataforma e constitui um dos principais desenvolvimentos a ser efectuado. Este módulo deverá ser responsável pela gestão dos pedidos de execução, controlo da execução dos pedidos, monitorização e gestão de parâmetros do sistema.

O conceito principal em torno deste módulo é o de Pedido de Integração. É através deste que a plataforma recebe a informação sobre qual o objecto de negócio e função que se pretende executar e respectivos parâmetros de entrada e saída.

Como resultado da execução de um pedido de integração poderá ser gerado um documento de resultado, contendo os parâmetros de saída e respectivos valores preenchidos e/ou uma secção com a descrição dos erros ocorridos.

Na figura 4.3 mostra-se o diagrama que esquematiza alguns dos aspectos principais deste bloco.

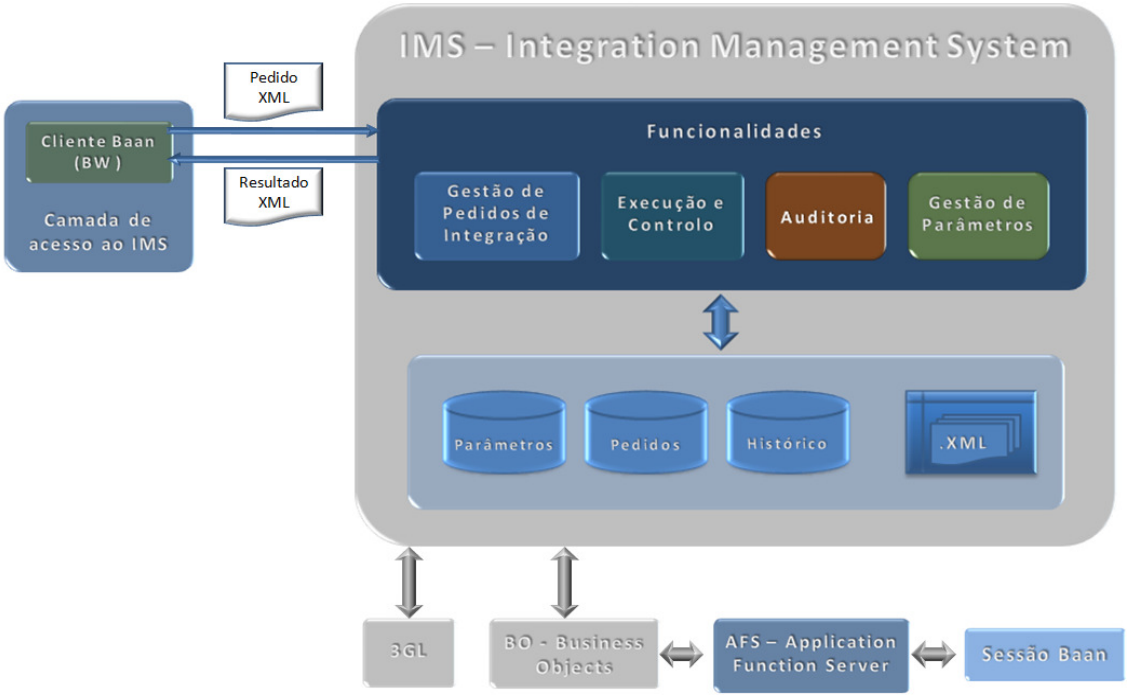


Figura 4.3 – Principais blocos do *Integration Management System - IMS*

4.3.1 Perspectiva do produto / Arquitectura (cont.)

O bloco *BO – Business Objects* representa um conjunto de programas compilados a serem desenvolvidos em *Baan tools*, com o objectivo de exporem funções de negócio para consumo das aplicações externas através do módulo *IMS*. Os *BO* podem envolver a simples utilização da lógica de negócio de uma sessão Baan através do *AFS,* até à utilização de várias sessões, regras próprias, funções de outras *DLLs – Dynamic Link Library*, acessos à base de dados do *ERP*, ou até mesmo outros *BO*.

O bloco 3GL – representa um conjunto de programas compilados e desenvolvidos em *Baan Tools* 3GL, ficando reservado o seu uso para operações específicas que não envolvam a utilização da lógica de negócio existente.

O bloco AFS - *Application Function Server* constitui um módulo já existente no sistema *ERP* Baan e representa a forma de aceder de forma programática às sessões Baan, através de um protocolo de mensagens de baixo nível. O detalhe da arquitectura e funcionamento deste módulo encontra-se descrito na secção 2.2.4 desta dissertação.

O bloco Sessão Baan representa as aplicações do sistema *ERP* Baan cujas funcionalidades se pretendem executar. As sessões Baan englobam normalmente: ecrãs; listagens; programas e tabelas. As sessões constituem o principal meio através do qual a lógica de negócio é exposta e implementada através da interacção com os utilizadores do sistema *ERP*. Apesar de nesta arquitectura essa interacção ser substituída pelo acesso programático do *AFS*, a lógica de negócio da sessão não é adulterada.

A base de dados do sistema *ERP* constitui o repositório de toda a informação manipulada pelas sessões Baan.

4.3.1 Perspectiva do produto / Arquitectura (cont.)

Do lado do Servidor da Mobilidade, a plataforma comporta essencialmente a camada de acesso ao *Integration Management System – IMS*. Esta camada deverá ser responsável por comunicar com o servidor *ERP* e expor as funcionalidades do *IMS* no ambiente do servidor da mobilidade, para que o *Web Service* [W3C, 2009d] as possa utilizar e expor às aplicações móveis.

No diagrama da figura 4.4 esquematizam-se os principais componentes previstos para esta camada.

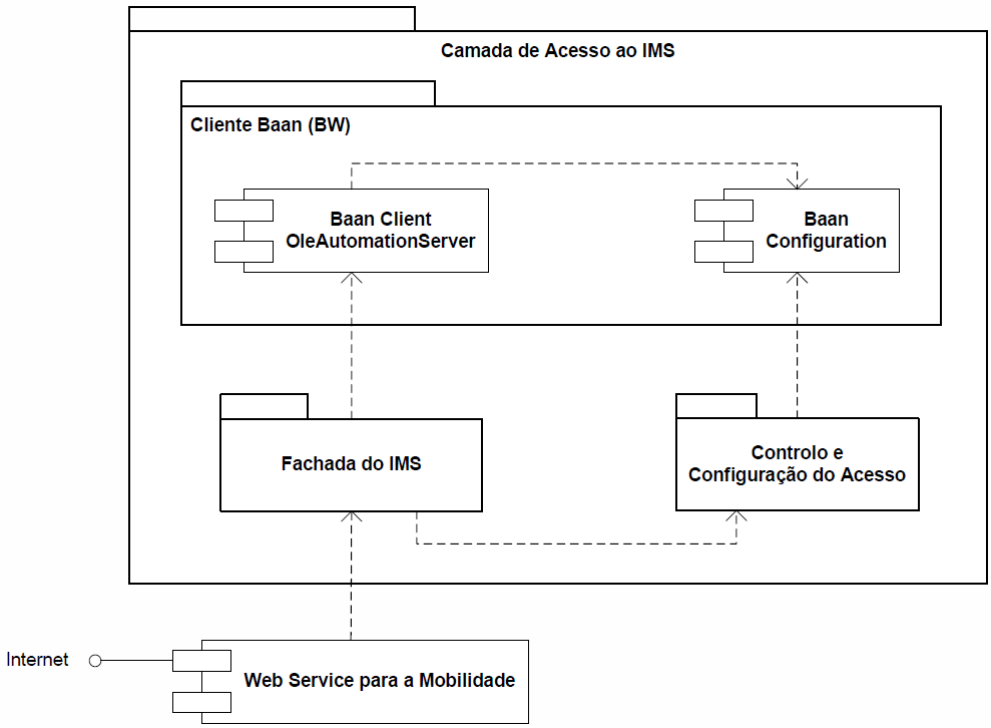


Figura 4.4 – Principais componentes da Camada de Acesso ao *IMS*

Esta camada deverá utilizar os dois componentes *COM (Component Object Model)* fornecidos pelo *software* cliente de acesso ao sistema *ERP* Baan: *OleAutomationServer* e *Baan Configuration*.

O Baan Client *OleAutomationServer* é um componente *COM* compatível com *Automation* e permite chamar funções de bibliotecas compiladas dinamicamente (*DLL*) existentes remotamente no servidor *ERP* Baan. Nesta plataforma será responsável por chamar as funções disponibilizadas pelo módulo *IMS*, anteriormente descrito.

4.3.1 Perspectiva do produto / Arquitectura (cont.)

Uma das vantagens da utilização deste componente é a reutilização da forma de comunicar com o servidor *ERP*, evitando a necessidade de protocolos e componentes de comunicação adicionais.

O *Baan Configuration* representa o componente *COM* que expõe, através da classe *BwConfigurationClass*, um conjunto de propriedades e métodos necessários ao carregamento e manipulação de ficheiros de configuração do acesso cliente ao sistema *ERP* Baan. Estes ficheiros definem um conjunto de parâmetros particularmente relevantes para esta plataforma, como por exemplo, o nome do servidor *ERP*, o utilizador e a companhia nesse servidor.

A Fachada *IMS* expõe as funcionalidades do módulo remoto *IMS* e encapsula todos os detalhes da utilização do componente *OleAutomationServer* na lógica de acesso ao sistema *ERP* Baan.

O componente Controlo e Configuração do Acesso permite gerir os acessos à plataforma, criando e carregando dinamicamente informação nos ficheiros de configuração do acesso cliente ao sistema ERP Baan. Desta forma, o mecanismo de autenticação ao servidor *ERP* é mantido e reutilizado por esta plataforma.

O *Web Service* para a Mobilidade representam nesta plataforma a forma de expor as funcionalidades disponibilizadas pelo componente Fachada do *IMS* às aplicações móveis através da *Internet/Intranet*.

O alto nível de interoperabilidade dos *Web Services* permite a comunicação com um vasto conjunto de aplicações e constituem normalmente a melhor abordagem para a maioria das aplicações móveis. Os *Web Services* utilizam protocolos normalizados, amplamente implementados, e como tal, passam normalmente pelos mecanismos de *firewall* sem grandes configurações adicionais (ao contrário de outras tecnologias que requerem a abertura de portos específicos). Permitem, pela mesma razão, que sistemas de diferentes plataformas tecnológicas e diferentes linguagens possam utilizar serviços comuns.

4.3.1 Perspectiva do produto / Arquitectura (cont.)

Finalmente, do lado do cliente móvel, a aplicação deve ser estruturada em três camadas típicas: interface com o utilizador; lógica de negócio local e camada de dados.

A camada de dados locais é responsável por conter os dados obtidos entre operações de sincronização da informação e contém normalmente dados base e operacionais. Estes dados são necessários ao funcionamento da aplicação, uma vez que o modo ocasionalmente conectado pressupõe momentos em que a aplicação está desconectada de qualquer infra-estrutura de rede.

A camada lógica de negócio local, para além da sua habitual missão, é responsável nesta arquitectura por chamar as funções dos *Business Objects – BO*. Esta chamada é efectuada recorrendo às funcionalidades do *IMS* expostas pelos *Web Services* do servidor da mobilidade. Os métodos dos *Web Services* são consumidos recorrendo ao "Agente para Web Service". As posteriores respostas terão como impacto a actualização os dados locais da aplicação móvel, sendo desta forma processada a sincronização dos mesmos.

A camada de interface para o utilizador visa fornecer toda a interacção do utilizador com a aplicação. Neste âmbito, esta camada deverá estar preparada para lidar com diversas situações proporcionadas por uma panóplia de dispositivos móveis que oferecem, entre outros aspectos, diferentes ecrãs e orientações, teclados físicos e virtuais, e interfaces multimodais.

Porém, a aplicação móvel não constitui o objecto de estudo principal, mas sim a plataforma que permite a estas executarem funcionalidades existentes nas aplicações dos sistemas *ERP* Baan. Neste contexto, em relação à aplicação móvel importa salientar a existência do bloco "Agente para Web Service" que representa a infra-estrutura especialmente vocacionada para o suporte à execução de *Web Services* em aplicações ocasionalmente conectadas. Este agente deverá seguir o padrão do *Disconnected Service Agent Application Block* já exposto na secção 3.3.3 desta dissertação.

4.3.2 Interfaces

Para a troca de mensagens com a plataforma deve ser escolhido um formato que permita uma boa interoperabilidade entre sistemas. Como tal, o pedido de integração bem como o seu resultado deverão ser descritos em *eXtensible Markup Language (XML)* [W3C, 2009a]. São propostos de seguida os respectivos esquemas para o pedido de integração e resultado, que se podem observar nas figuras 4.5 e 4.6 respectivamente.

Pedido de integração

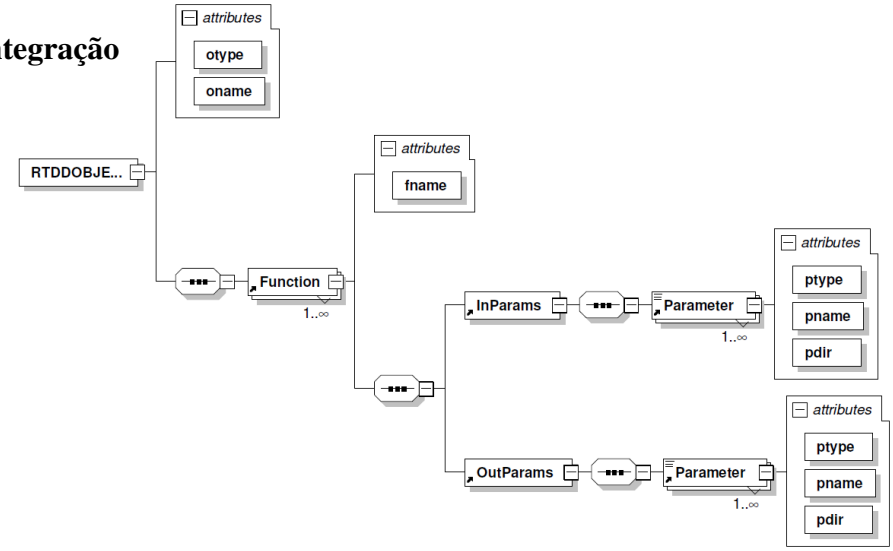


Figura 4.5 – Esquema para o *XML* do pedido de integração

Resultado

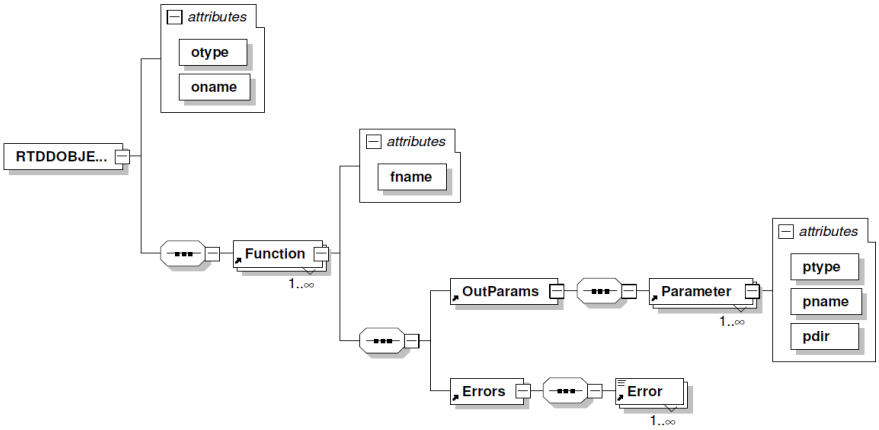


Figura 4.6 – Esquema para o *XML* do resultado de processamento do pedido

Como se pode verificar, em ambos os esquemas propostos deve existir a indicação do *Business Object* (*RTDDOBJECT*) e da função (*Function*) utilizada. No pedido serão indicados os parâmetros de entrada (*InParameter*) e de saída (*OutParams*). No resultado, são indicados os parâmetros de saída, com os valores preenchidos resultantes do processamento e uma secção de erros ocorridos (*Errors*).

4.3.2 Interfaces (cont.)

Interfaces de Comunicação

A comunicação entre o servidor *ERP* Baan e o servidor da mobilidade deverá ser efectuada utilizando o protocolo *Remote Execution Protocol (REXEC)* sobre *Transmission Control Protocol/Internet Protocol (TCP/IP)* [IETF, 1991]. O REXEC é normalmente o protocolo utilizado na comunicação entre o servidor *ERP* Baan e os diversos clientes quando estes utilizam o BW (programa cliente de acesso).

Entre o servidor da mobilidade e os clientes da plataforma serão utilizados protocolos normalizados, nomeadamente os envolvidos na disponibilização dos *Web Services*.

Apesar dos *Web Services* serem agnósticos relativamente ao protocolo que os disponibilizam, normalmente é utilizado o *HyperText Transfer Protocol (HTTP)* [W3C, 1999].

Dada a natureza da informação empresarial, é de considerar nesta plataforma a adopção de medidas de segurança adicionais tais como o fornecimento de transporte seguro para o protocolo *HTTP* recorrendo ao *Transport Layer Security (TLS)* [IETF, 2000]. Com a implementação do *TLS* a comunicação entre a aplicação cliente e o *Web Service* é encriptada, passando as mensagens *Simple Object Access Protocol (SOAP)* [W3C, 2009b] a utilizar o protocolo *HyperText Transfer Protocol Secure (HTTPS)*.

4.3.3 Funcionalidades do produto

As funcionalidades do produto enquadram-se nos seguintes grupos de processos funcionais:

- Gestão dos pedidos de integração – engloba todo um conjunto de funcionalidades necessárias para criar e realizar acções sobre os pedidos de integração;
- Execução e controlo dos pedidos – envolve as tarefas necessárias para que um pedido execute num tempo e número de tentativas pré-definidos;
- Auditoria – engloba as funcionalidades que permitem responder às perguntas básicas: quem executou, o quê, com que parâmetros e quando;
- Gestão de parâmetros – engloba a simples manutenção de parâmetros necessários ao funcionamento de todo o sistema, como por exemplo, directorias de trabalho, tempo padrão máximo de execução de processos;
- Geração de classes facilitadoras – envolve as funcionalidades necessárias à disponibilização de classes na linguagem de programação das aplicações externas. Estas classes visam auxiliar a interacção da aplicação externa com a funcionalidade do objecto de negócio que se pretende executar através desta plataforma.

4.3.4 Características dos utilizadores

Os utilizadores deste produto enquadram-se nos seguintes perfis:

- Programador de objectos de negócio – com conhecimento de programação em Baan *tools*, responsável por desenvolver os *Business Objects* (*BO*) anteriormente descritos no ponto referente à arquitectura;
- Administrador de sistema – responsável por instalar os pacotes, módulos e tabelas do produto no sistema *ERP* Baan e efectuar a gestão do servidor *ERP*;
- Programador consumidor das classes facilitadoras – responsável por desenvolver as aplicações externas ao sistema *ERP,* utilizando as classes de auxílio da utilização da plataforma desenvolvida.

4.3.5 Restrições

O consumo das necessárias licenças de acesso ao sistema *ERP* Baan, resultantes da utilização da plataforma deve ser optimizado para o mínimo possível.

Os diversos níveis de permissões de acesso que um dado utilizador tem no sistema *ERP* (como por exemplo, sessões, opções de menu, tabelas e campos), não devem ser alterados pelo facto de, eventualmente, executar alguma funcionalidade através da plataforma.

4.3.6 Pressupostos e dependências

É assumido que os utilizadores deste produto já estão familiarizados com o sistema *ERP* Baan.

A plataforma a ser desenvolvida depende da existência do sistema *ERP* Baan instalado numa das seguintes versões: Baan IV; Baan 5.0c; Baan LN 6.1.

4.4 Requisitos Específicos

Com o objectivo de sistematizar e rastrear o desenvolvimento e testes, os requisitos específicos funcionais e não funcionais estão devidamente identificados e agrupados.

Os funcionais encontram-se agrupados de forma a permitir um fácil relacionamento com os principais grupos de processos funcionais a disponibilizar pelo produto, nomeadamente: gestão de pedidos de integração; execução e controlo dos pedidos; auditoria e gestão de parâmetros.

Dentro de cada grupo, encontram-se ordenados pela sua prioridade.

4.4.1 Requisitos Funcionais

ID	Descrição	Grupo	Prioridade
RF01	Criar pedido de integração	Gestão de Pedidos	Alta
O sistema deverá permitir a recepção de um documento *XML* que descreve o pedido de integração, persistir o mesmo no local pré-definido, persistir a informação adicional e de controlo na tabela de pedidos de integração do módulo *IMS* e com o estado igual a "Criado". A resposta a esta chamada deve indicar o número único atribuído ao pedido, ou zero em caso de insucesso.			

ID	Descrição	Grupo	Prioridade
RF02	Executar pedido de integração	Gestão de Pedidos	Alta
Chamada para executar um pedido de integração que deverá referir o número do pedido e a escolha do modo síncrono ou assíncrono. No modo síncrono o processo que chama fica bloqueado até que o pedido seja processado. No assíncrono não existe o referido bloqueio. A resposta a esta chamada deve retornar verdadeiro no caso do pedido ser colocado em execução, ou falso em caso de insucesso.			

4.4.1 – Requisitos Funcionais (cont.)

ID	Descrição	Grupo	Prioridade
RF03	Colocar pedido de integração no estado "Espera"	Gestão de Pedidos	Alta
Uma chamada referindo o número do pedido, deverá colocar este no estado "Espera", actualizando a tabela de pedidos de integração. A resposta a esta chamada deve retornar verdadeiro no caso do pedido ser colocado no estado Espera, ou falso em caso de insucesso.			

ID	Descrição	Grupo	Prioridade
RF04	Consultar estado de um pedido de integração	Gestão de Pedidos	Alta
Uma chamada referindo o número do pedido de integração, deve obter o estado deste, consultando a tabela de pedidos de integração. A resposta a esta chamada deve retornar o valor que corresponde ao estado, ou zero no caso de insucesso.			

ID	Descrição	Grupo	Prioridade
RF05	Obter resultado de um pedido de integração	Gestão de Pedidos	Alta
Para pedidos com o estado "Executado" ou "Com Erros", uma chamada referindo o número do pedido deve obter um documento *XML* de resultado contendo os valores resultantes do processamento desse pedido. A resposta a esta chamada deve retornar um documento *XML* de resultado em caso de sucesso ou zero no caso de insucesso.			

ID	Descrição	Grupo	Prioridade
RF06	Sessão para tratamento de pedidos de integração pendentes	Gestão de Pedidos	Alta
A sessão Baan deverá colocar em execução, os pedidos que estejam no estado "Espera". O escalonamento da execução destes pedidos deve ser do tipo *FIFO (First In, First Out)*. Esta sessão deverá ser agendada numa tarefa periódica do gestor de tarefas do *ERP* Baan.			

4.4.1 – Requisitos Funcionais (cont.)

ID	Descrição	Grupo	Prioridade
RF07	Criar e executar pedido de integração numa única chamada	Gestão de Pedidos	Média
Para optimização dos acessos ao servidor *ERP*, criar uma única chamada que combine as operações descritas em RF01 com as do RF02. A resposta a esta chamada deve retornar nos seus parâmetros de saída os resultados das duas operações de forma individualizada.			

ID	Descrição	Grupo	Prioridade
RF08	Criar e colocar o pedido em Espera numa única chamada	Gestão de Pedidos	Média
Para optimização dos acessos ao servidor *ERP*, criar uma única chamada que combine as operações descritas em RF01 com as do RF03. A resposta a esta chamada deve retornar nos seus parâmetros de saída os resultados das duas operações de forma individualizada.			

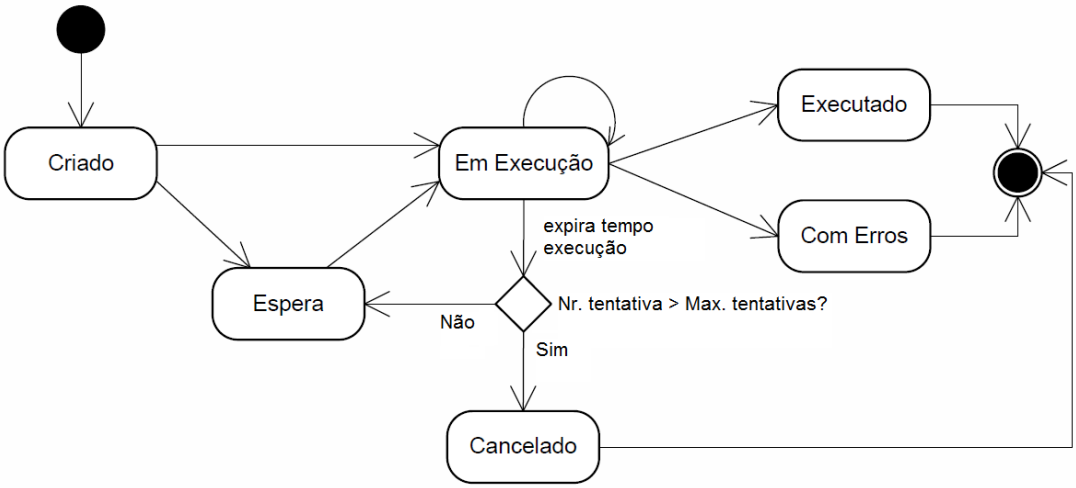
ID	Descrição	Grupo	Prioridade
RF09	Criar pedido de integração, executar e obter resultado numa única chamada	Gestão de Pedidos	Média
Para optimização dos acessos ao servidor *ERP*, criar uma única chamada que combine as operações descritas em RF01, RF02 e RF05. A resposta a esta chamada deve retornar nos seus parâmetros de saída os resultados das três operações de forma individualizada.			

ID	Descrição	Grupo	Prioridade
RF10	Sessão que permita a gestão dos pedidos de integração de forma visual	Gestão de Pedidos	Baixa
A sessão Baan deverá ser capaz de criar, colocar em espera, executar, obter resultado, localizar e filtrar os diversos pedidos de integração.			

4.4.1 Requisitos Funcionais (cont.)

ID	Descrição	Grupo	Prioridade
RF11	Validador de transições de estado do pedido de integração	Execução e Controlo	Alta

O sistema deve garantir que todas as transições de estado dos pedidos de integração cumpram a validade descrita na seguinte máquina de estados:



A resposta a esta chamada deve retornar verdadeiro se a transição for possível, ou falso caso seja inválida.

ID	Descrição	Grupo	Prioridade
RF12	Lançamento e monitorização do processo de execução	Execução e Controlo	Alta

Este programa deverá ser responsável pela criação de um novo grupo de processos, no qual irá lançar o programa de processar o pedido de integração.
Após este lançamento, fica responsável por monitorizar o tempo de execução do grupo de processos, eliminando-o caso este ultrapasse o tempo máximo de execução definido no pedido de integração.
Caso não seja especificado um tempo limite de execução no pedido, deverá ser aplicado o tempo padrão definido nos parâmetros do módulo *IMS*.

4.4.1 – Requisitos Funcionais (cont.)

ID	Descrição	Grupo	Prioridade
RF13	Processar pedido de integração	Execução e Controlo	Alta
O programa deverá ser capaz de: 1 abrir o documento *XML* do pedido de integração e verificar a sua conformidade; 2 interpretar cada uma das suas secções; 3 lançar a execução do objecto de negócio ou programa 3GL referidos no pedido e com os respectivos parâmetros; 4 gerar documento de resultado			

ID	Descrição	Grupo	Prioridade
RF14	Histórico das transições de estado dos pedidos de integração	Auditoria	Alta
A menos que o pedido especifique que não deve guardar histórico, qualquer transição do seu estado deverá ficar registada numa tabela de histórico dos pedidos. Para além do estado deve ser guardada a data hora minuto e segundo, o utilizador, e no caso de erro a descrição do mesmo.			

ID	Descrição	Grupo	Prioridade
RF15	Sessão de consulta de histórico dos pedidos de integração	Auditoria	Média
Deverá existir uma sessão Baan que permita consultar todo o histórico de um pedido de integração. A sessão deverá permitir a pesquisa e aplicação de filtros sobre os pedidos com histórico.			

4.4.1 Requisitos Funcionais (cont.)

ID	Descrição	Grupo	Prioridade
RF16	Parametrização do tempo máximo pré-definido para a execução dos pedidos de integração	Gestão de Parâmetros	Alta
Deverá ser possível estabelecer um tempo máximo de execução a ser aplicado sempre que um pedido de integração não o especifique.			

ID	Descrição	Grupo	Prioridade
RF17	Parametrização da localização do repositório *XML* de pedidos e resultados de integração.	Gestão de Parâmetros	Alta
Deverá ser possível definir o local de armazenamento dos ficheiros *XML* referentes à persistência dos pedidos de integração e respectivos resultados.			

ID	Descrição	Grupo	Prioridade
RF18	Sessão para manutenção de parâmetros do módulo *IMS*	Gestão de Parâmetros	Média
Deverá existir uma sessão Baan que permita a manutenção dos diversos parâmetros do módulo *IMS*, necessários ao funcionamento de toda a plataforma.			

4.4.2 Requisitos Não Funcionais

ID	Descrição
RNF01	Preservação dos níveis de acesso do sistema *ERP* Baan
Os diversos níveis de permissões de acesso que um dado utilizador tem no sistema *ERP,* não devem ser alterados pelo facto de, eventualmente, executar alguma funcionalidade através da plataforma.	

ID	Descrição
RNF02	Optimização do uso das licenças de acesso ao sistema ERP Baan
O consumo das necessárias licenças de acesso ao sistema *ERP* Baan, resultantes da utilização da plataforma, deve ser optimizado para 1 licença por companhia/servidor e apenas durante o tempo necessário à satisfação dos pedidos de integração.	

ID	Descrição
RNF03	Disponibilidade da plataforma
Deverá ser equivalente à disponibilidade do sistema *ERP* Baan que a suporta.	

ID	Descrição
RNF04	Número de utilizadores concorrentes suportados
A plataforma deverá suportar um mínimo de 25 utilizadores concorrentes, para cada combinação companhia/servidor do sistema *ERP* Baan.	

ID	Descrição
RNF05	Tempo máximo para criação do pedido de integração
95% das transacções deverão ser concluídas em menos de 2 segundos em horário normal de trabalho. Os restantes 5% não deverão exceder os 10 segundos. Como é óbvio, este tempo só se aplica às chamadas feitas à plataforma que não envolvam a espera pelo processamento do pedido de integração. Um pedido de integração até poderia demorar dias a ser processado.	

4.4.2 Requisitos Não Funcionais (cont.)

ID	Descrição
RNF06	Linguagem de programação da camada de acesso ao *IMS* existente no servidor de mobilidade
A linguagem de programação utilizada para o desenvolvimento da camada de acesso ao *IMS* deve ser C#.	

Capítulo 5

Implementação

No capítulo anterior foi dada uma visão geral da plataforma que se pretende desenvolver, focando vários aspectos fundamentais para esta fase de implementação, nomeadamente: o âmbito; a arquitectura; interfaces; funcionalidades; restrições; dependências e sobretudo a definição de um conjunto de requisitos específicos funcionais e não funcionais devidamente agrupados, prioritizados e identificados de forma a sistematizar e rastrear as fases seguintes.

Este capítulo visa precisamente descrever os aspectos mais relevantes da implementação desses requisitos e está dividido em três partes. Uma inicial, onde são feitas breves considerações sobre a metodologia e plataforma de implementação adoptadas. Na segunda parte é apresentada a implementação realizada no servidor *ERP* Baan e detalhado o desenvolvimento do módulo *Integration Management System – IMS*, que constitui o principal componente da plataforma. Na terceira parte é descrita a implementação da Camada de Acesso ao *IMS*, desenvolvida para o servidor da mobilidade.

5.1 Metodologia

O processo de desenvolvimento do software pode ser definido, de forma muito simples, como um conjunto de actividades requeridas para transformar os requisitos num sistema de software que representa o produto final [Jacobson, Booch e Rumbaugh, 1999].

Como podemos verificar no capítulo anterior, a arquitectura da plataforma que se pretende desenvolver está dividida em dois subsistemas principais: o módulo *Integration Management System – IMS* no servidor *ERP* Baan e a camada de acesso ao *IMS* no servidor de mobilidade.

Para o desenvolvimento destes subsistemas foi adoptada uma abordagem segundo o modelo iterativo e incremental. Este modelo caracteriza-se basicamente pelo desenvolvimento do produto em incrementos, sobre os quais ocorrem iterações de refinamento que se sucedem ao longo do tempo. Por sua vez, as iterações podem envolver a totalidade ou parte do seguinte conjunto de actividades: requisitos; análise; desenho; implementação e testes. Como tal, em cada iteração podem ocorrer alterações de desenho e incremento de funcionalidades.

A escolha é justificada por um conjunto de vantagens que este modelo oferece [Schach, 2004] e que são particularmente úteis neste desenvolvimento, nomeadamente:

- a robustez da arquitectura é testada relativamente cedo;
- oferece múltiplos pontos de verificação da adequação do produto (cada iteração que incorpore a fase de testes é uma oportunidade de validação dos artefactos desenvolvidos até ao momento);
- é centrado nas funcionalidades e na arquitectura do sistema;
- permite mitigar os riscos precocemente.

Nesta perspectiva, o desenvolvimento foi orientado pela prioridade dos requisitos e divide-se segundo uma lógica de incremento de funcionalidades, começando por um conjunto básico de funcionalidades e passando por sucessivas iterações até à obtenção do produto final.

O desenvolvimento do subsistema *Integration Management System – IMS* dividiu-se em vários grupos de funcionalidades:

- gestão de pedidos de integração;
- execução e controlo dos pedidos de integração;
- auditoria;
- gestão de parâmetros.

O desenvolvimento do subsistema Camada de acesso ao *IMS* dividiu-se em:

- codificação do subsistema de controlo e configuração do acesso;
- codificação do subsistema Fachada do *IMS*;
- definição e codificação do *Web Service* [W3C, 2009d] que expõe para a Internet/Intranet as funcionalidades da Fachada do *IMS*.

Para o desenvolvimento do subsistema *IMS* foi criado um módulo específico - o *IMS* com a descrição "*Integration Management System*". O referido módulo foi criado no pacote "am" (acrónimo de António Meireles S.A.) e visou encapsular todos os artefactos (*DLL*, programas, tabelas, sessões, ecrãs) desenvolvidos para este subsistema.

Importa ainda referir que no desenvolvimento de algumas sessões Baan pertencentes ao módulo *IMS*, foram utilizadas técnicas de quarta geração. Estas técnicas não constituem um método, mas sim um termo que abrange um conjunto de ferramentas que auxiliam e automatizam o desenvolvimento, a partir de algumas definições de alto nível. No caso do desenvolvimento das sessões Baan, a sessão é gerada a partir das definições da tabela principal, indicação do tipo de sessão, tipo de ecrã, entre outras. Como é óbvio o código, ecrãs e relatórios gerados são apenas um esqueleto para o "verdadeiro" desenvolvimento, porém reduzem significativamente o tempo necessário à criação dessa base de partida.

5.2 Plataforma de Implementação

Dada a natureza dos dois subsistemas que compõem o produto, as plataformas de implementação são necessariamente distintas.

O subsistema *IMS*, está assente no *software ERP* existente na António Meireles S.A. que é o sistema *ERP* Baan 5.0c. No desenvolvimento deste subsistema foi utilizada uma linguagem proprietária deste *software ERP* – o Baan *Tools*. O código desenvolvido nesta linguagem é essencialmente de dois tipos: 3GL e 4GL. O 3GL está associado a uma lógica procedimental, baseada numa sequência de instruções. A linguagem 4GL é de mais alto nível, visando objectivos específicos, sendo o código desenvolvido numa lógica orientada ao evento. Por exemplo, o código associado a uma sessão Baan é do tipo 4GL.

Para a implementação do subsistema Camada de acesso ao *IMS*, foi escolhida a plataforma .NET [MSFT, 2009b] da Microsoft e a linguagem de programação C# [MSFT, 2009c]. Esta escolha deveu-se ao facto de, por um lado, a plataforma .NET ter sido projectada desde o início para incluir suporte aos *Web Services*, por outro, permite uma fácil integração entre o código *managed* das aplicações .NET e código *unmanaged* dos componentes *COM* [MSFT, 1994].

Para uma plena utilização da plataforma .NET e da linguagem de programação C#, foi escolhido um ambiente de desenvolvimento integrado - o Visual Studio .NET [MSFT, 2009d], que contém um conjunto de ferramentas particularmente úteis para o desenvolvimento deste subsistema, nomeadamente as relacionadas com os *Web Services* e as relacionadas com importação de componentes *COM*.

5.3 O Integration Management System – IMS

O módulo *IMS* é composto por um conjunto de programas, sessões, tabelas e bibliotecas compiladas dinamicamente (*DLL*), que criam as funcionalidades necessárias do lado do servidor *ERP* Baan, para que o subsistema "Camada de Acesso ao *IMS*" as possa expor às aplicações externas, utilizando um esquema de mensagens pré-estabelecido e descrito em *XML* [W3C, 2009a]. Relembrando o esquema apresentado no capítulo anterior, apresenta-se de novo na figura 5.1.

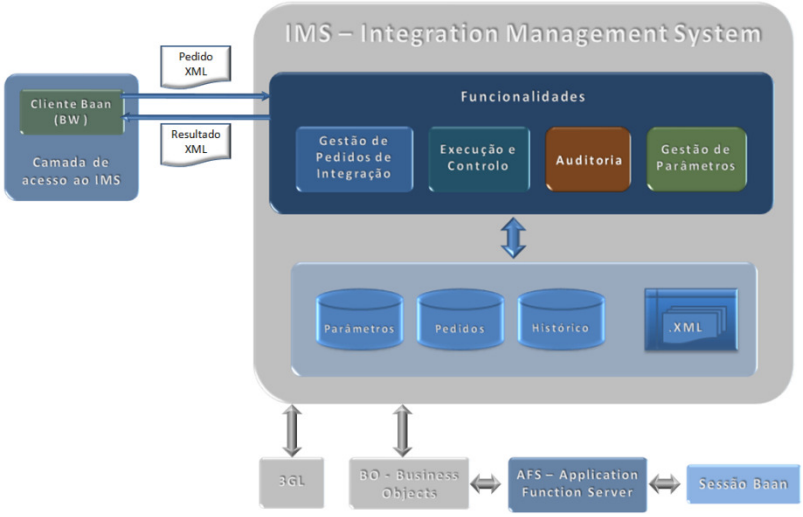


Figura 5.1 – Principais blocos do *Integration Management System - IMS*

Apresentamos neste capítulo, na figura 5.2, um esquema da implementação deste módulo referindo os nomes dos artefactos desenvolvidos e mantendo um esquema de cores relacionado com os grupos de funcionalidades descritos no esquema anterior.

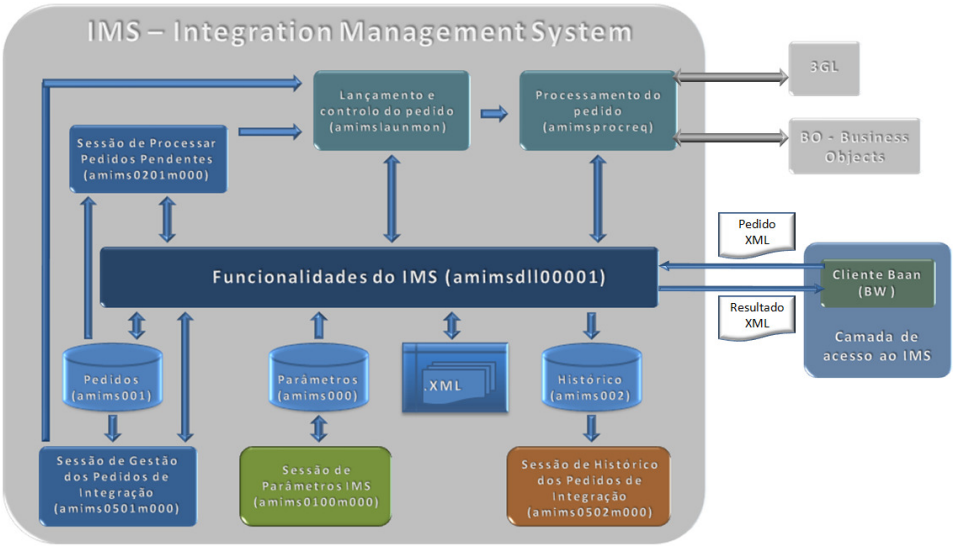


Figura 5.2 – Arquitectura do *Integration Management System - IMS*

5.3.1 A DLL amimsdll00001

Como se pode verificar no esquema da figura 5.2, um dos componentes centrais deste módulo é a biblioteca compilada dinamicamente (*DLL*) amimsdll00001. Esta biblioteca é responsável por expor as funcionalidades necessárias à utilização da plataforma através do subsistema "Camada de Acesso ao *IMS*".

A tabela 5.1 sintetiza as principais funcionalidades fornecidas por esta *DLL*, fazendo referência ao requisito funcional subjacente ao seu desenvolvimento.

Nome da Função	Descrição	ID do Requisito
create.request	Cria um pedido de integração. Recebe os atributos do pedido e devolve o seu número único, ou zero em caso de insucesso.	RF01
put.request.waiting	Coloca um pedido de integração no estado "Espera", para ser posteriormente executado. Recebe o número de pedido e devolve verdadeiro (1) em caso de sucesso, ou falso (0) no caso de insucesso .	RF03
execute.request	Coloca um pedido em execução. Recebe o número de pedido e o modo de execução (0 – assíncrono, 1 – síncrono) e devolve verdadeiro (1) em caso de sucesso, ou falso (0) no caso de insucesso.	RF02
get.request.stat.const	Consulta o estado de um pedido de integração. Recebe o número de pedido e devolve o valor da constante associada ao enumerado que representa o estado do pedido (1 – Criado, 2 - Espera, 3 – Em Execução, 4 – Processado, 5 – Com Erros, 6 - Cancelado), ou zero em caso de insucesso.	RF04
get.request.proc.result	Obtém o resultado de um pedido de integração. Recebe o número de pedido e devolve um documento *XML* [W3C, 2009a] de resultado, contendo os valores do processamento desse pedido, ou zero em caso de insucesso.	RF05
create.and.put.waiting.request	Cria um pedido e coloca-o em espera. Recebe os atributos do pedido e retorna nos parâmetros de saída, o número único atribuído ao pedido e o resultado (1 – Sucesso, 0 – Insucesso) da segunda operação.	RF08
create.and.execute.request	Cria e coloca em execução, um pedido. Recebe os atributos do pedido e retorna nos parâmetros de saída, o número único atribuído ao pedido e o resultado (1 – Sucesso, 0 – Insucesso) da segunda operação.	RF07

5.3.1 A *DLL* amimsdll00001 (cont.)

Nome da Função	Descrição	ID do Requisito
create.exec.and.get.result	Cria pedido de integração, executa e obtém o resultado. Recebe os atributos do pedido e retorna nos parâmetros de saída, o número único atribuído ao pedido e os resultados (1 – Sucesso, 0 – Insucesso) da segunda e terceira operação. No caso de sucesso devolve ainda um documento *XML* de resultado, contendo os valores do processamento desse pedido.	RF09
update.request.status	Centraliza qualquer alteração do estado de um pedido e cria o histórico da alteração efectuada. Recebe o número de pedido e o estado pretendido, devolvendo verdadeiro (1) no caso de sucesso, ou falso (0) em caso de insucesso.	RF14
check.stat.trans	Valida uma dada transição de estado. Recebe o estado de origem e o estado de destino, devolvendo verdadeiro (1) quando a transição é válida, ou falso (0) caso seja inválida.	RF11

Tabela 5.1 – Funções da DLL amimsdll00001

A tabela 5.2 refere ainda outras funções fornecidas por esta *DLL*, que apesar de não terem directamente associado um requisito, são necessárias a vários programas e sessões deste módulo.

Nome da Função	Descrição
read.parameters	Lê os parâmetros do módulo *IMS*.
get.free.num	Obtém um número único de pedido de integração.
get.in.model.fname	Obtém o modelo de ficheiro *XML* para criação de um pedido de integração de forma interactiva.
get.in.fname.from.req.num	Obtém o ficheiro *XML* persistido de um pedido de integração através da indicação do seu número identificador.
get.out.fname.from.req.num	Obtém o ficheiro *XML* persistido com o resultado do processamento de um pedido através da indicação do seu número.

Tabela 5.2 – Funções auxiliares da DLL amimsdll00001

5.3.1 A DLL amimsdll00001 (cont.)

A título de exemplo do código desenvolvido nesta *DLL* apresentamos o realizado para a função amimsdll00001.update.request.status:

```
...
 10          table   tamims000                        |Parâmetros do IMS
 11          table   tamims001                        |Pedidos de Integração
 12          table   tamims002                        |Histórico dos Pedidos de Integração
 13
 14          #pragma used dll ottdllbw
...
 308   function extern long amimsdll00001.update.request.status (domain tcmcs.long p.reqn,
 309                                                             domain amims.stat p.stat,
 310                                                             domain tcmcs.str215m p.lmes)
 311   {
 312           domain  tcdate          aux.date
 313           long                    ret
 314
 315           ret = 0
 316
 317           db.retry.point()
 318
 319           select amims001.*
 320           from amims001 for update
 321           where amims001._index1 = {:p.reqn}
 322           selectdo
 323                   if amimsdll00001.check.stat.trans(amims001.stat, p.stat) then
 324
 325                           if p.stat = amims.stat.in.proc then
 326                                   amims001.rnum = amims001.rnum + 1
 327                           endif
 328
 329                           if p.stat <> amims.stat.in.proc or
 330                              ((p.stat = amims.stat.in.proc and amims001.rnum <= amims001.mxrn)
 331                              or amims001.mxrn = 0) then
 332
 333                                   aux.date = utc.num()
 334
 335                                   if amims001.hist = tcyesno.yes then
 336
 337                                           select amims002.*
 338                                           from amims002 for update
 339                                           where amims002._index1 = {:p.reqn}
 340                                           order by amims002._index1 desc
 341                                           as set with 1 rows
 342                                           selectdo
 343                                                   amims002.sqnb = amims002.sqnb + 1
 344                                                   amims002.modt = aux.date
 345                                                   amims002.logn = logname$
 346                                                   amims002.stat = p.stat
 347                                                   amims002.lmes = p.lmes
 348
 349                                                   db.insert(tamims002, DB.RETRY)
 350                                           endselect
 351                                   endif
 352
 353                                   amims001.stat = p.stat
 354                                   amims001.lmdt = aux.date
 355                                   amims001.lmes = p.lmes
 356
 357                                   db.update(tamims001, DB.RETRY)
 358
 359                                   ret = commit.transaction() = 0 ? 1 : 0
 360                           endif
 361                   endif
 362           endselect
 363
 364           return(ret)
 365   }
...
```

O código das restantes funções desta *DLL* pode ser consultado no anexo A.1.

5.3.2 Execução e Controlo

A execução e controlo dos pedidos de integração é implementada por dois programas 3GL que satisfazem os requisitos funcionais RF12 e RF13. O programa **amimslaunmon** é responsável por criar um novo grupo de processos, no qual lança a execução do programa de processamento do pedido de integração **amimsprocreq**. Após este lançamento, fica responsável por monitorizar o tempo de execução desse grupo de processos, eliminando-o caso este ultrapasse o tempo máximo de execução definido no pedido de integração. Se o pedido não definir um tempo máximo no qual deve executar é aplicado o tempo máximo definido nos parâmetros do módulo *IMS*. A ideia subjacente a este controlo é a de evitar que um pedido fique em execução por tempo indeterminado, causado por exemplo, pelo bloqueio provocado por uma situação não tratada, durante a execução da lógica de uma sessão através do *Application Function Server - AFS*.

Complementarmente a este controlo, pode ser definido o número máximo de tentativas que um dado pedido tem para conseguir executar. Se este número for atingido, o pedido é colocado num estado cancelado e não volta a ser processado. Caso não tenha sido atingido, ou não esteja definido esse limite, o pedido será reprocessado ciclicamente pela sessão de processar pedidos pendentes (amims0201m000), existente numa tarefa periódica do gestor de tarefas do *ERP* Baan.

O programa **amimsprocreq** é responsável pelo processamento do pedido de integração e comporta as seguintes tarefas: abrir o documento *XML* do pedido de integração; verificar a sua conformidade; interpretar cada uma das suas secções; lançar a execução do objecto de negócio ou programa 3GL referidos no pedido com os respectivos parâmetros; gerar o documento *XML* de resultado.

5.3.2 Execução e Controlo (cont.)

Os programas **amimslaunmon** e **amimsprocreq** são responsáveis por realizar uma série de alterações ao estado de um pedido de integração. A sua máquina de estados pode ser observada no diagrama da figura 5.3.

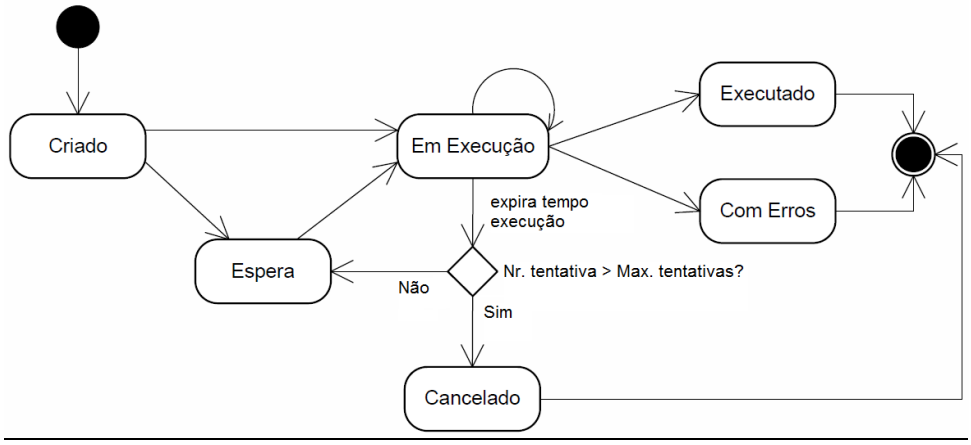


Figura 5.3 – Diagrama de Estados do Pedido de Integração

Neste âmbito, o programa **amimslaunmon** é responsável pela passagem do estado de um pedido "Em Execução" para "Cancelado" ou para "Espera", consoante o número máximo de tentativas tenha sido ou não ultrapassado. O programa **amimsprocreq** está associado à passagem do estado de um pedido "Em Execução" para "Executado", após o seu processamento, ou para "Com Erros" se ocorrerem situações que inviabilizem o processamento do pedido de integração.

5.3.2 Execução e Controlo (cont.)

No desenvolvimento destes programas relativos à execução e controlo dos pedidos de integração foi importante o contributo dos utilitários *ottgbfprocess* e o comando "ps" em *ottstpshell*, fornecidos pelo sistema *ERP* Baan. Estes utilitários permitiram obter várias informações sobre processos lançados pelo cliente e respectiva execução no servidor, possibilitando uma forma simples de efectuar algumas verificações durante a fase de desenvolvimento e testes.

Os ecrãs destes utilitários podem ser observados nas figuras 5.4 e 5.5.

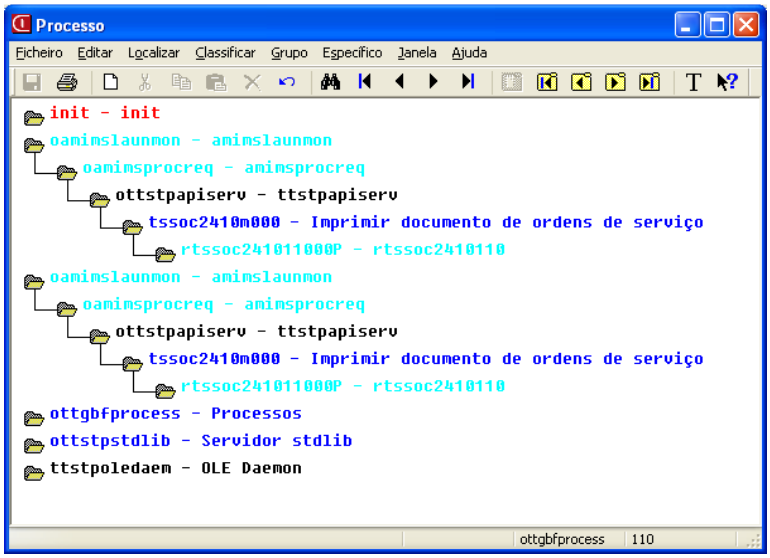


Figura 5.4 – Visualização de processos com o utilitário *ottgbfprocess*

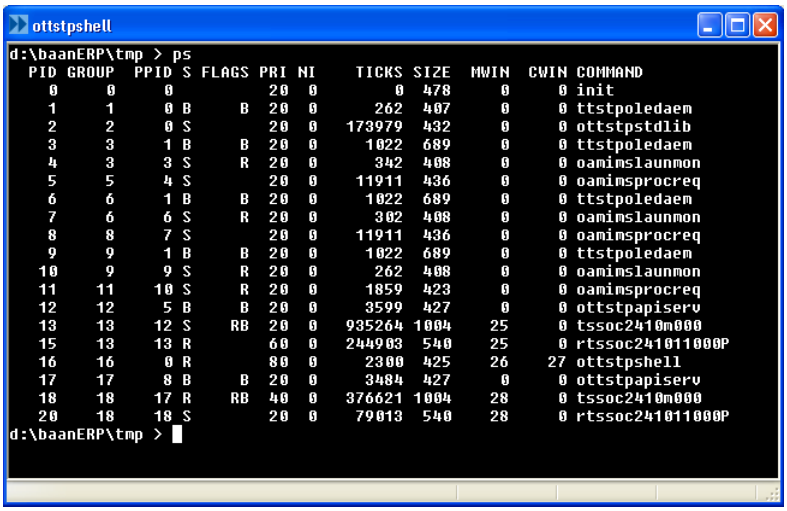


Figura 5.5 – Informação disponibilizada pelo comando ps

5.3.2 Execução e Controlo (cont.)

A título de exemplo do código 3GL desenvolvido no programa **amimsprocreq** apresentamos uma das suas partes – a execução do *Business Object* indicado no documento *XML* do pedido e com os respectivos parâmetros de entrada e saída:

```
1   |*****************************************************************************
2   |* amimsprocreq  0  VRC B500 c  expl
3   |* Processamento de pedidos de Integração
4   |* jaugusto
5   |* 10-02-09 [16:54]
6   |*****************************************************************************
7   |* Script Type: 0
8   |*****************************************************************************
9
10          #pragma used dll oamimsdll00001
11
12          long                    arg.reqn
13          long                    ret.xml
14          string                  lmess(10)
15
16          #define PTYPE_BASIC (ptype = "string" or ptype = "long" or ptype = "double" or ptype = "date")
17          #define PTYPE_XMLST (ptype = "xmlst")
18
...
```

Função que processa o caso do pedido ser do tipo ImsBO (*Business Object*), carregando-o em memória, bem como a função deste a ser executada:

```
...
97   function long exec.ImsBO.request(domain tcmcs.str15 p.bprogram,
98                                    domain tcmcs.str215 p.funcname,
99                                    const long p.inp, const long p.onp)
100  {
101         long                    ret, np, np.out, ap1, ap2, error.sec
102         long                    param.ok, has.out.param, outsect, dllid, funcid
103         string                  imsvar(500)
104         string                  auxdata(1024)
105         string                  ptype(6)
106
107         ret = 0
108
109         dllid = load_dll(strip$(shiftl$(p.bprogram)))
110         funcid = get_function(dllid, p.funcname)
111
112         if funcid > 0 then
...
```

Percorre os parâmetros de entrada e atribui os valores às variáveis respectivas do *BO*:

```
...
117                 np = xmlGetFirstChild(p.inp)
118
119                 while np <> 0
120
121                         ap1 = xmlGetAttribute(np, "pname", imsvar)
122                         ap2 = xmlGetAttribute(np, "ptype", ptype)
123
...
```

5.3.2 – Execução e Controlo (cont.)

Continuação do exemplo do código 3GL desenvolvido no programa **amimsprocreq**: Percorre os parâmetros de entrada e atribui os valores às variáveis respectivas do *BO*:

```
124                     if ap1 <> 0 and ap2 <> 0 then
125
126                             if PTYPE_BASIC then
127                                     if xmlGetData(np, auxdata) > 0 then
128                                             if not put.var(pid, imsvar, strip$(shiftl$(auxdata)) ) then
129                                                     param.ok = false
130                                                     break
131                                             endif
132                                     endif
133                             else
134                                     if PTYPE_XMLST then
135                                             if not put.var(pid, imsvar, np) then
136                                                     param.ok = false
137                                                     break
138                                             endif
139                                     else
140                                             param.ok = false
141                                             break
142                                     endif
143                             endif
...
```

Se os parâmetros de entrada estiverem todos correctos, então é executada a função do *BO*:

```
...
199                 if param.ok then
200
201                         exec_function(dllid, funcid)
202
203                         ret = 1
...
```

5.3.2 Execução e Controlo (cont.)

Se existirem parâmetros de saída, os seus valores são colocados na estrutura de resultado:

```
...
207             outsect = xmlFindFirst("OutParams", ret.xml)
208
209             if outsect <> 0 then
210
211                     np = xmlGetFirstChild(p.onp)
212
213                     while np <> 0
214
215                             ap1 = xmlGetAttribute(np, "pname", imsvar)
216                             ap2 = xmlGetAttribute(np, "ptype", ptype)
217
218                             if ap1 <> 0 and ap2 <> 0 then
219
220                                     if PTYPE_BASIC then
221
222                                             np.out = xmlDuplicateAndAppendToChilds(outsect, np)
223
224                                             if np.out <> 0 then
225
226                                                     if get.var(pid, imsvar, auxdata) then
227
228                                                             if xmlNewNode(strip$(shiftl$(auxdata)),
229                                                                           XML_DATA,
230                                                                           np.out) = 0 then
231                                                                     lmess = "amims00015"
232                                                             endif
233                                                     else
234                                                             lmess = "amims00015"
235                                                     endif
236                                             else
237                                                     lmess = "amims00015"
238                                             endif
239                                     else
240                                             if get.var(pid, imsvar, np.out) then
241
242                                                     if xmlDuplicateAndAppendToChilds(outsect, np.out) = 0 then
243                                                             lmess = "amims00015"
244                                                     endif

...
```

Finalmente, é obtida a estrutura de erros:

```
...
256                     error.sec = xmlFindFirst("Errors", ret.xml)
257
258                     if error.sec <> 0 then
259                             if get.var(pid, "ims.Errors", np.out) then
260                                     error.sec = np.out
261                             else
262                                     lmess = "amims00016"
263                             endif
...
```

O restante código deste programa, bem como o do **amimslaunmon**, podem ser consultados respectivamente nos anexos A.3 e A.2.

5.3.3 Sessões Desenvolvidas

Apesar do acesso a este módulo ser essencialmente programático, foram desenvolvidas várias sessões Baan com o objectivo de possibilitar a manutenção e consulta dos dados existentes de forma interactiva e visual. A criação destas sessões facilitou a consulta e rastreio da utilização do sistema *ERP* Baan efectuada através da plataforma desenvolvida. Foram criadas as seguintes sessões:

- Pedidos de Integração (amims0501m000)
- Detalhe do Pedido de Integração (amims0101s000)
- Criar Pedidos de Integração (amims0101m000)
- Processar Pedidos de Integração Pendentes (amims0201m000)
- Parâmetros *IMS* (amims0100m000)
- Histórico dos Pedidos de Integração (amims0502m000)

As sessões, Pedidos de Integração (amims0501m000), Detalhe do Pedido de Integração (amims0101s000) e Criar Pedidos de Integração (amims0101m000) visam a satisfação do requisito funcional RF10, permitindo criar, executar, colocar em espera, obter resultado, localizar e filtrar os diversos pedidos de integração. As seguintes figuras representam os ecrãs destas sessões:

- Pedidos de Integração (amims0501m000) – Nesta sessão é possível consultar, filtrar e localizar os diversos pedidos de integração. Permite abrir a sessão de detalhe de um dado pedido e realizar operações sobre este.

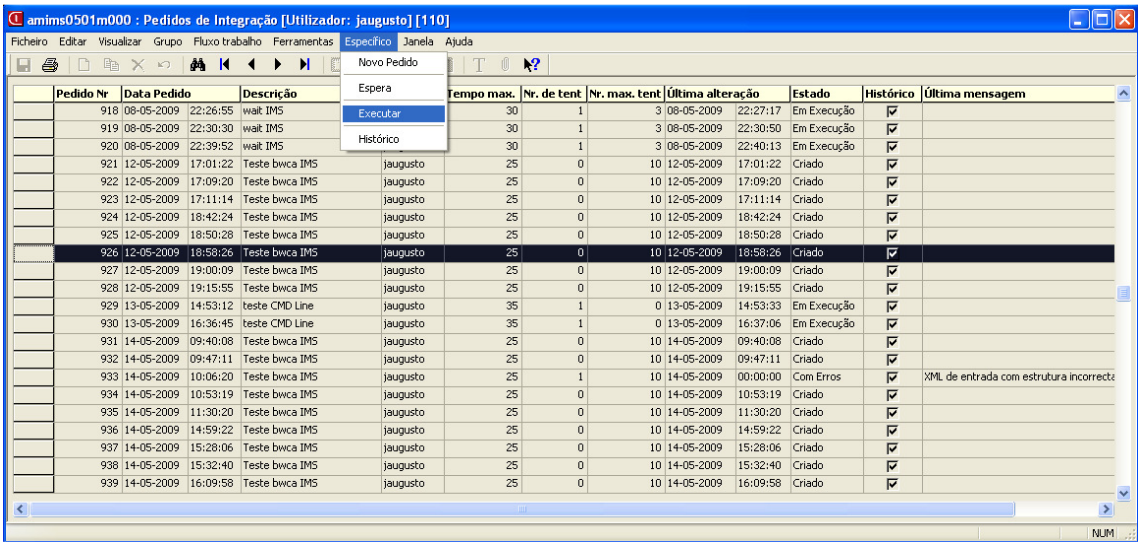


Figura 5.6 – Ecrã da sessão de Pedidos de Integração

5.3.3 Sessões Desenvolvidas (cont.)

- Detalhe do Pedido de Integração (amims0101s000) – É uma sessão secundária da sessão anterior (amims0501m000) que permite visualizar os atributos de um dado pedido, realizar operações sobre este e consultar os ficheiros *XML* que contêm toda a informação do pedido e resultado do seu processamento.

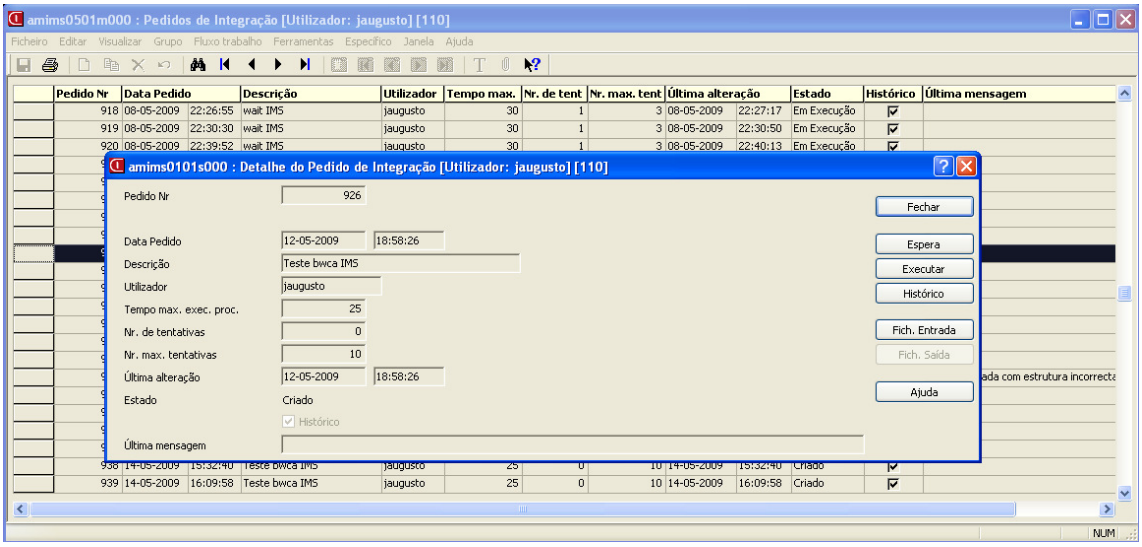


Figura 5.7 - Ecrã da sessão de Detalhe do Pedido de Integração

- Criar Pedidos de Integração (amims0101m000) – Nesta sessão é possível criar um pedido de integração, preenchendo os seus atributos e fornecendo um ficheiro *XML* que descreva todos os detalhes desse pedido.

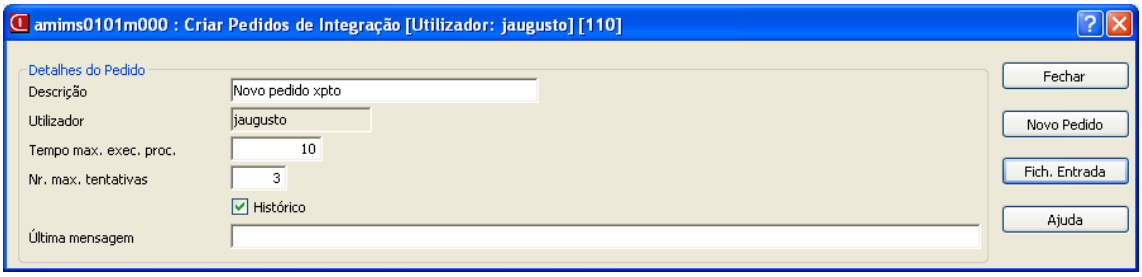


Figura 5.8 – Ecrã da sessão Criar Pedidos de Integração

5.3.3 Sessões Desenvolvidas (cont.)

A sessão Processar Pedidos de Integração Pendentes (amims0201m000) visa responder ao requisito funcional RF06. Coloca em execução os pedidos que estejam no estado "Espera", começando pelos mais antigos. Esta sessão foi agendada numa tarefa periódica do gestor de tarefas do *ERP* Baan.

- Processar Pedidos de Integração Pendentes (amims0201m000)

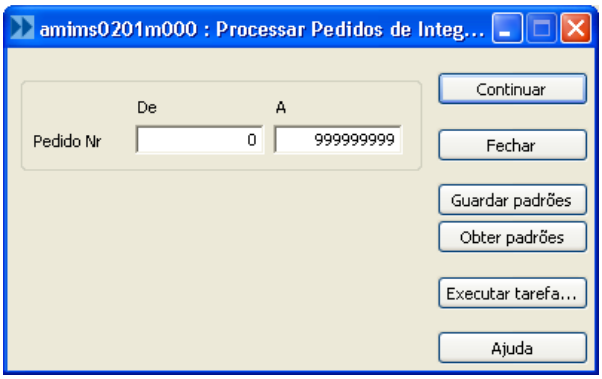


Figura 5.9 – Ecrã da sessão de Processar Pedidos de Integração

A sessão Parâmetros *IMS* (amims0100m000) visa responder aos requisitos funcionais RF16, RF17, RF18 e permite consultar e efectuar a manutenção dos parâmetros necessários ao funcionamento do módulo *IMS*.

- Parâmetros *IMS* (amims0100m000)

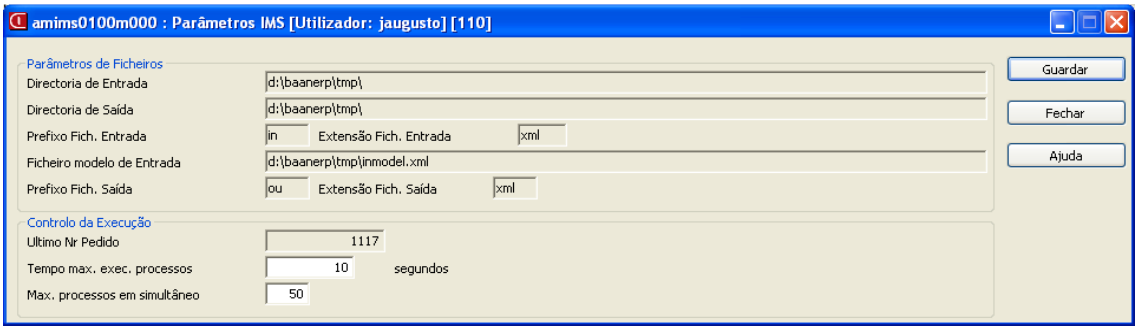


Figura 5.10 – Ecrã da sessão Parâmetros *IMS*

5.3.3 Sessões Desenvolvidas (cont.)

A sessão Histórico dos Pedidos de Integração (amims0502m000) visa responder ao requisito funcional RF15 e permite consultar a informação sobre todas as transições de estado de um pedido, o momento em que ocorreram, com que utilizador e última mensagem (no caso de erros) retornada pelo processamento do pedido.

- Histórico dos Pedidos de Integração (amims0502m000)

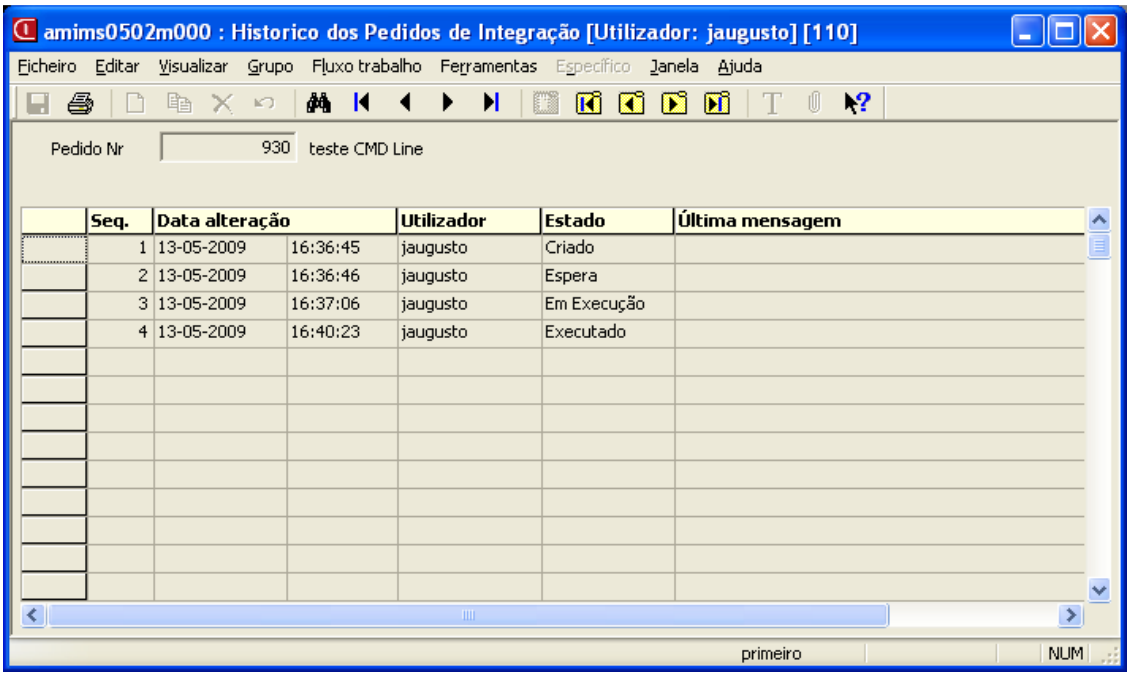


Figura 5.11 – Ecrã da sessão de Histórico dos Pedido de Integração

5.3.4 Diagrama de Instalação

Para finalizar a descrição da implementação do módulo *IMS* é apresentado na figura 5.12 o diagrama de instalação, onde podemos observar os principais componentes, ambiente de execução e dispositivos.

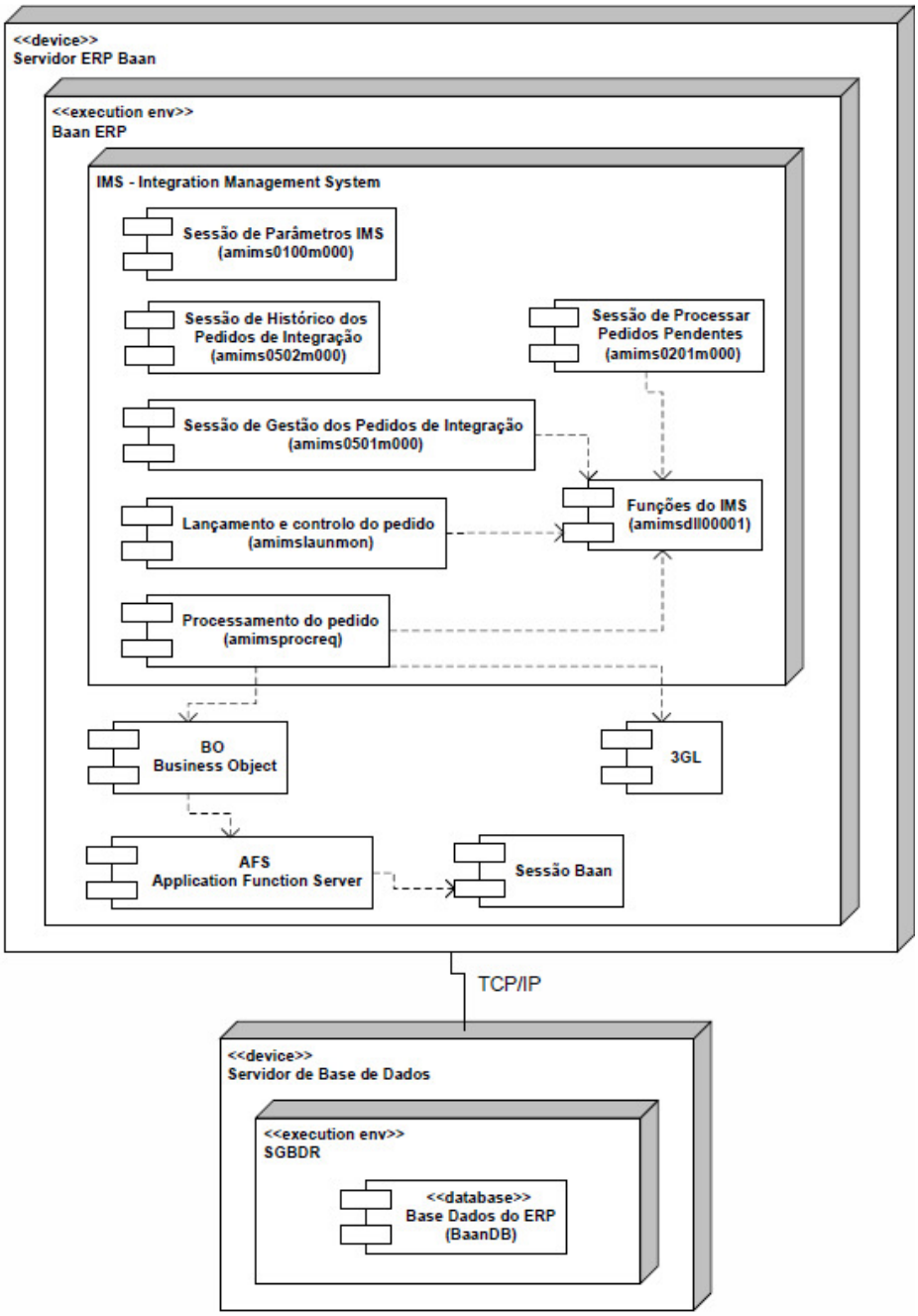


Figura 5.12 – Diagrama de instalação do *Integration Management System - IMS*

5.4 Camada de Acesso ao IMS

Do lado do Servidor de Mobilidade, a plataforma desenvolvida comporta essencialmente o subsistema Camada de Acesso ao *IMS*. Esta camada é responsável por comunicar com o servidor *ERP* e expor as funcionalidades do *IMS* no ambiente do servidor de mobilidade, para que os *Web Services* para a mobilidade as possam utilizar e expor às aplicações móveis.

Os diversos elementos constituintes deste subsistema são formados por um conjunto de classes desenvolvidas segundo o paradigma orientado a objecto, recorrendo à linguagem de programação C# e aplicando alguns padrões de desenho que contribuíram para uma melhor arquitectura do *software*.

Relembramos aqui, na figura 5.13, o diagrama exposto no capítulo anterior relativo à arquitectura da camada de acesso ao *IMS*.

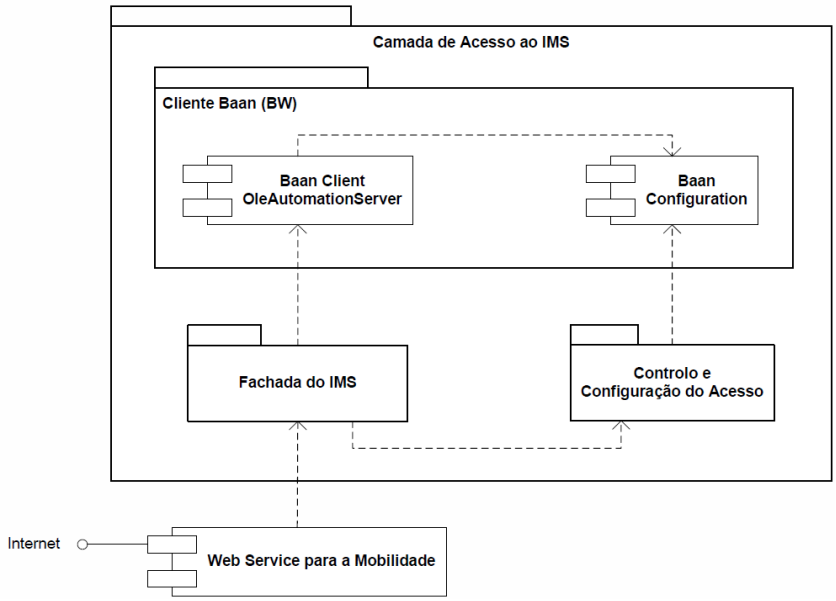


Figura 5.13 – Principais componentes da Camada de Acesso ao *IMS*

A camada de acesso ao *IMS* é composta por dois subsistemas principais: o Controlo e Configuração do Acesso e a Fachada do *IMS*.

O Controlo e Configuração do Acesso permite gerir os acessos à plataforma, utilizando as funcionalidades expostas pelo componente *Baan Configuration*, permitindo desta forma reutilizar o mecanismo de autenticação ao servidor *ERP* Baan.

A Fachada do *IMS* expõe as funcionalidades do módulo remoto *IMS*, de uma forma mais simplificada, encapsulando a complexidade dos detalhes da utilização do

componente *OleAutomationServer* na lógica de acesso ao sistema *ERP* Baan. São igualmente escondidos atrás desta fachada os detalhes da serialização e deserialização dos pedidos e resultados, decorrentes da utilização da plataforma.

O *Web Service* para a Mobilidade representa nesta plataforma a forma de expor as funcionalidades disponibilizadas pelo componente Fachada do *IMS* às aplicações móveis, através da *Internet/Intranet*.

O alto nível de interoperabilidade dos *Web Services* permite a comunicação com um vasto conjunto de aplicações e constituem normalmente a melhor abordagem para a maioria das aplicações móveis.

Para o suporte à execução de *Web Services* em aplicações ocasionalmente conectadas deve existir um agente na aplicação móvel de acordo com o padrão do *Disconnected Service Agent Application Block* já exposto na secção 3.3.3 desta dissertação.

5.4.1 Os Componentes OleAutomationServer e Baan Configuration

Como se pode verificar no diagrama da figura 5.13, os dois subsistemas da camada de acesso ao *IMS* utilizam componentes *COM (Component Object Model)* [MSFT, 1994] fornecidos pelo *software* cliente de acesso ao sistema ERP Baan (*BW*): o *OleAutomationServer* e o *Baan Configuration*.

O Baan Client *OleAutomationServer* é um componente *COM* compatível com *Automation* e permite chamar funções de bibliotecas compiladas dinamicamente (*DLL*) existentes remotamente no servidor *ERP* Baan. Nesta plataforma é responsável por chamar as funções disponibilizadas pela *DLL* amims00001 do módulo *IMS*, anteriormente descrito. O subsistema Fachada do *IMS*, utiliza precisamente este componente para chamar essas funções.

Uma das vantagens da utilização deste componente é a reutilização da forma de comunicar com o servidor *ERP*, evitando a necessidade de protocolos e componentes de comunicação adicionais.

O *Baan Configuration* representa o componente *COM* que expõe, através da classe *BwConfigurationClass*, um conjunto de propriedades e métodos necessários ao carregamento e manipulação de ficheiros de configuração do acesso cliente ao sistema *ERP* Baan. Estes ficheiros definem um conjunto de parâmetros particularmente relevantes para esta plataforma, como por exemplo, o nome do servidor *ERP*, o utilizador e a companhia nesse servidor. O subsistema Controlo e Configuração do Acesso utiliza as funcionalidades da classe *BwConfigurationClass* para criar e carregar dinamicamente informação, em ficheiros de configuração, do acesso cliente ao sistema *ERP* Baan.

A utilização deste componente, permitiu à plataforma desenvolvida reutilizar o mesmo mecanismo de autenticação ao servidor *ERP* Baan.

5.4.2 O subsistema Controlo e Configuração do Acesso

O subsistema Controlo e Configuração do Acesso permite gerir os acessos à plataforma, criando dinamicamente a informação necessária à utilização do mecanismo de autenticação, comum ao programa cliente de acesso ao sistema *ERP* Baan (*BW*). Para o efeito, foi desenvolvido um conjunto de classes que disponibilizam a informação e as funcionalidades necessárias. O diagrama mostrado na figura 5.14 apresenta as principais classes deste subsistema.

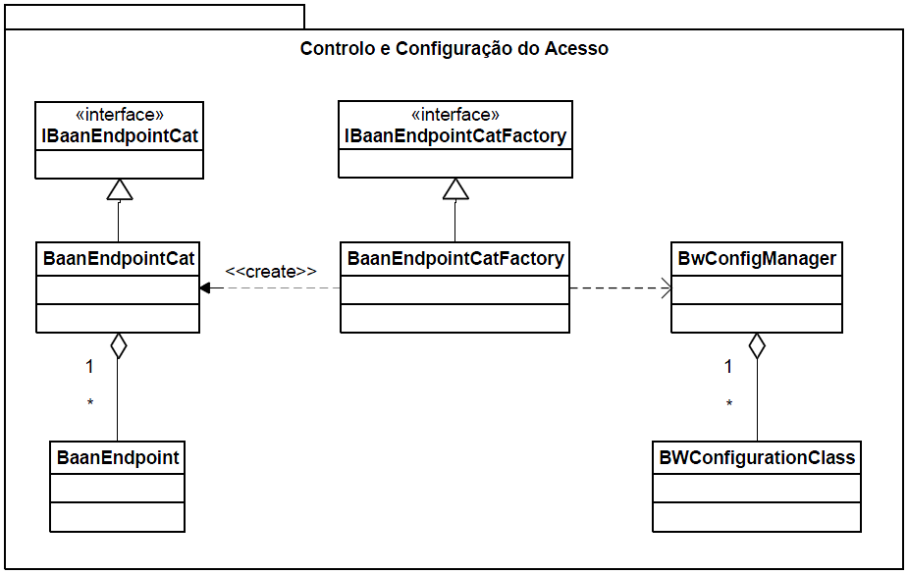


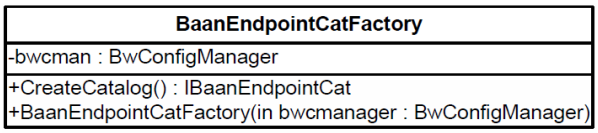
Figura 5.14 – Subsistema de Controlo e Configuração do Acesso

Como podemos observar no diagrama da Figura 5.14 foi aplicado o padrão de desenho *Factory Method* para a criação de instâncias da classe ***BaanEndpointCat***, que possui e gere um catálogo de pontos de ligação (***BaanEndpoint***). A classe ***BaanEndPoint*** representa a informação necessária para o estabelecimento de uma ligação a um servidor *ERP* Baan (servidor, utilizador de sistema, password, utilizador Baan, companhia, protocolo, etc.). Porém, o software cliente de acesso (*BW*), necessita que parte dessa informação esteja guardada num ficheiro de configuração. Como tal, a classe ***BaanEndpointCatFactory***, para além de criar o catálogo, cria os respectivos ficheiros de configuração necessários aos pontos de ligação existentes nesse catálogo, recorrendo ao método ***CreateBwcForEndpoint*** da classe ***BwConfigManager***. Esta classe gere as configurações existentes e cria novas, através das propriedades e métodos expostos pela classe ***BWConfigurationClass*** disponibilizada pelo componente *COM Baan Configuration*.

5.4.2 O subsistema Controlo e Configuração do Acesso (cont.)

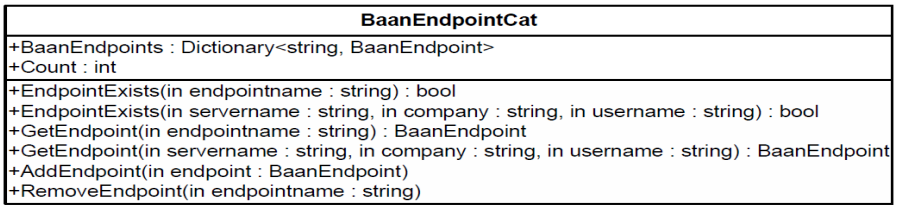
Apresenta-se agora, uma breve descrição das principais classes desenvolvidas para este subsistema, nomeadamente: *BaanEndpointCatFactory*; *BaanEndpointCat*; *BaanEndPoint* e *BwConfigManager*. A classe *BWConfigurationClass* não foi desenvolvida, foi somente utilizada, encontrando-se a sua descrição na secção 2.2.3 desta dissertação.

A classe ***BaanEndpointCatFactory***



Esta classe, que implementa a interface ***IBaanEndpointCatFactory***, é responsável por criar instâncias da classe ***BaanEndpointCat*** através da invocação do seu método ***CreateCatalog***. Recebe no seu método construtor uma instância da classe ***BwConfigManager***, para gerar os respectivos ficheiros de configuração necessários aos pontos de ligação da classe ***BaanEndpointCat*** que criou.

A classe ***BaanEndpointCat***



Esta classe, que implementa a interface ***IBaanEndpointCat***, mantém um dicionário de pontos de ligação (***BaanEndpoint***) e expõe um conjunto de propriedades e métodos que permitem:

obter (propriedade ***BaanEndpoints*** e os dois métodos ***GetEndpoint***);

testar a existência (propriedade ***Count*** e os dois métodos ***EndpointExists***);

e manter (métodos ***AddEndpoint*** e ***RemoveEndpoint***) as diversas instâncias desses pontos de ligação.

5.4.2 O subsistema Controlo e Configuração do Acesso (cont.)

A classe ***BaanEndPoint***

BaanEndpoint
+BwcFileName : string +ServerName : string +ServerBSE : string +Protocol : string +PortNumber : ushort +Company : string +BaanUser : string +UserName : string +Password : string +BShell : string +BwClassName : string
+BaanEndpoint() +BaanEndpoint(in pbwcfilename : string, in pserver : string, in pserverbse : string, in pprotocol : string, in pportnumber : ushort, in pcompany : string, in pbaanuser : string, in pusername : string, in ppassword : string, in pbshell : string, in pbwclassname : string)

Esta classe abstrai a informação necessária para o estabelecimento de uma ligação a um servidor *ERP* Baan e expõe através de um conjunto de propriedades essa informação, por exemplo, o nome do servidor Baan (***ServerName***), utilizador Baan (***BaanUser***), companhia (***Company***), utilizador de sistema (***UserName***) e palavra passe (***Password***).

A classe ***BaanEndpoint*** é utilizada neste subsistema pelas classes ***BaanEndpointCat*** e ***BwConfigManager*** e no subsistema Fachada do *IMS* pelas classes ***BWA*** e ***BaanIMS***.

A classe ***BwConfigManager***

BwConfigManager
+BwcFileDirectory : string +Count : int +BwConfigurations : Dictionary<string, BWConfigurationClass>
+BwConfigManager(in bwcfiledir : string) +CreateBwcForEndpoint(in be : BaanEndpoint) : bool +BwcExists(in BwcFileName : string) : bool +GetBwConfig(in BwcFileName : string) : BWConfigurationClass +DeleteBwc(in BwcFileName : string) : bool

Esta classe gera e mantém um dicionário da informação contida nos ficheiros de configuração (.bwc) existentes na directoria recebida como argumento no seu método construtor. A classe ***BwConfigManager*** expõe um conjunto de propriedades e métodos que permitem:

obter (propriedade ***BwConfigurations*** e o método ***GetBwConfig***);

testar a existência (propriedade ***Count*** e o método ***BwcExists***);

e manter (métodos ***CreateBwcForEndpoint*** e ***DeleteBwc***) as diversas instâncias de ***BWConfigurationClass*** e respectivos ficheiros de configuração (.bwc) associados.

5.4.2 O subsistema Controlo e Configuração do Acesso (cont.)

A título de exemplo do código C# desenvolvido para estas classes apresenta-se um excerto do realizado para a classe *BaanEndpointCat*:

```
public class BaanEndpointCat : IBaanEndpointCat
    {
       private Dictionary<string, BaanEndpoint> endpoints = new
              Dictionary<string, BaanEndpoint>();

        public int Count
        {
            get { return endpoints.Count; }
        }

        public Dictionary<string, BaanEndpoint> BaanEndpoints
        {
            get { return endpoints; }
        }

        public bool EndpointExists(string endpointname)
        {
            if (! String.IsNullOrEmpty(endpointname))
                return endpoints.ContainsKey(endpointname);
            else
                return false;
        }
     ...

        public BaanEndpoint GetEndpoint(string servername,
                                        string company, string username)
        {
            if (String.IsNullOrEmpty(servername) ||
                String.IsNullOrEmpty(company) || String.IsNullOrEmpty(username))
                return null;
            else
            {
                foreach (BaanEndpoint be in endpoints.Values)
                {
                    if (be.ServerName == servername &&
                        be.Company == company && be.UserName == username)
                        return be;
                }

                return null;
            }
        }
     ...

        public void AddEndpoint(BaanEndpoint endpoint)
        {
            if (endpoints.ContainsKey(endpoint.BwcFileName))
                endpoints.Remove(endpoint.BwcFileName);

            endpoints.Add(endpoint.BwcFileName, endpoint);
        }

        public void RemoveEndpoint(string endpointname)
        {
            if (endpoints.ContainsKey(endpointname))
                endpoints.Remove(endpointname);
        }
    }
```

O restante código desta classe, bem como o desenvolvido para as restantes classes deste subsistema, podem ser consultados no anexo B.1.

5.4.3 O subsistema Fachada do IMS

O subsistema Fachada do *IMS* tem por objectivo expor as funcionalidades do módulo *Integration Management System - IMS* desenvolvido no sistema *ERP* Baan.

Para o efeito foi desenvolvido um conjunto de classes que disponibilizam as funcionalidades necessárias, segundo uma lógica de simplificação na utilização da plataforma. O diagrama da figura 5.15 apresenta as principais classes deste subsistema.

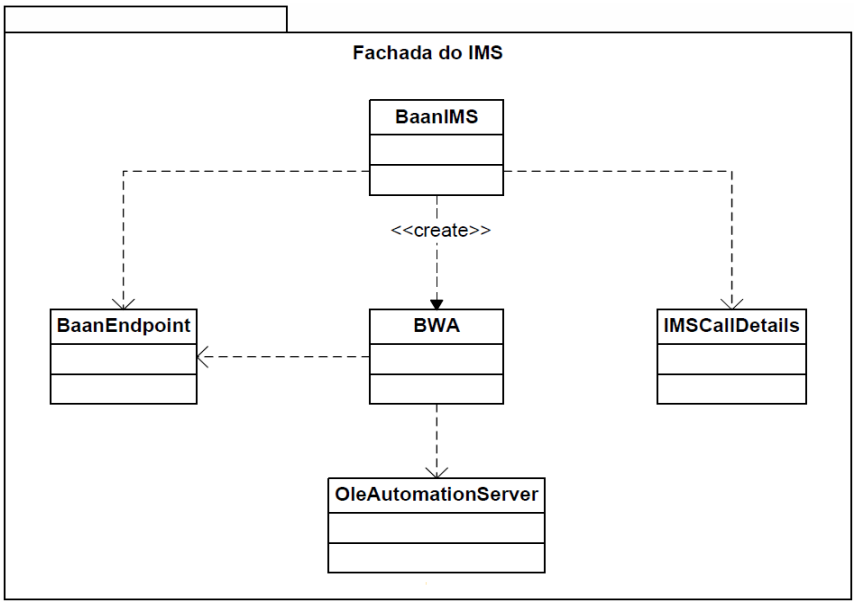


Figura 5.15 – Subsistema Fachada do *IMS*

A ***BaanIMS*** constitui a principal classe deste subsistema. É através dos métodos disponibilizados por esta que são expostas as funcionalidades do módulo *IMS* desenvolvido no lado do sistema *ERP* Baan. Esta classe constitui o ponto de entrada do subsistema e esconde os detalhes da utilização dos diversos componentes envolvidos. Nesta óptica podemos considerar que este subsistema se aproxima do padrão de desenho *Façade*. A classe ***BWA*** tem por objectivo encapsular e simplificar a utilização do ***OleAutomationServer*** na lógica de acesso ao sistema *ERP* Baan. O ***OleAutomationServer*** é um componente *COM* compatível com *Automation* que neste subsistema é responsável por chamar as funções da *DLL* amims00001, indicadas pela classe ***BaanIMS,*** utilizando a classe ***BWA***. A classe ***BaanEndpoint*** fornece a informação relativa ao ponto de ligação que a classe ***BWA*** deve utilizar. A classe ***IMSCallDetails*** é utilizada para acrescentar a informação de controlo associada a um pedido de integração.

5.4.3 O subsistema Fachada do *IMS* (cont.)

Apresenta-se agora, uma breve descrição das principais classes desenvolvidas para este subsistema, nomeadamente: *BaanIMS*; *IMSCallDetails*; *BWA* e *BaanEndpoint*. O componente *OleAutomationServer* foi somente utilizado, encontrando-se a sua descrição na secção 2.2.3 desta dissertação.

A classe ***BaanIMS***

BaanIMS
+EndpointInfo : BaanEndpoint
+BaanIMS(in bep : BaanEndpoint, in tempfiledir : string) +Create(in imsbo : BaanIMSBO, in imscall : IMSCallDetails) : int +PutWaiting(in preqn : int) : bool +Execute(in preqn : int, in pwait : bool) : bool +GetStatus(in preqn : int) : ReqStatus +GetResult(in preqn : int, in imsbo : BaanIMSBO) : bool +CreateAndPutWaiting(in imsbo : BaanIMSBO, in imscall : IMSCallDetails, out preqn : int, out pwaitret : bool) +CreateAndExecute(in imsbo : BaanIMSBO, in imscall : IMSCallDetails, out preqn : int, out pexecret : bool) +CreateExecuteAndGetResult(in imsbo : BaanIMSBO, in imscall : IMSCallDetails, out preqn : int, out pexecret : bool, out pgresret : bool) -SerializeCall(in imsbo : BaanIMSBO, in imscall : IMSCallDetails, in fname : string) : bool -DeserializeResult(in imsbo : BaanIMSBO, in fname : string) : bool -GetTmpFileName() : string

É através desta classe que as funcionalidades do módulo *IMS,* existente no servidor *ERP* Baan, são expostas no servidor da mobilidade.

A classe ***BaanIMS*** expõe uma propriedade e um conjunto de métodos que permitem:

obter os detalhes do ponto de ligação utilizado (propriedade ***EndpointInfo***);

criar um pedido de integração (método ***Create***);

colocar um pedido para execução não imediata (método ***PutWaiting***);

colocar um pedido em execução (método ***Execute***);

obter o estado de um pedido (método ***GetStatus***);

obter o resultado do processamento de um pedido (método ***GetResult***);

criar e colocar pedido para execução não imediata (método ***CreateAndPutWaiting***);

criar e colocar um pedido em execução (método ***CreateAndExecute***);

criar, executar e obter o resultado do processamento (método ***CreateExecuteAndGetResult***).

Esta classe possui ainda três métodos privados que são utilizados no processo de serialização e deserialização dos documentos *XML* relativos aos pedidos e resultados de integração.

5.4.3 O subsistema Fachada do *IMS* (cont.)

A classe ***IMSCallDetails***

IMSCallDetails
+Description : string +MaxTime : int +MaxRun : int +History : bool +DelLocFile : bool
+IMSCallDetails() +IMSCallDetails(in descr : string, in mxtime : int, in mxrun : int, in history : bool, in delfile : bool)

Esta classe disponibiliza um conjunto de propriedades necessárias à caracterização de alguns aspectos de controlo definidos no momento da criação de um pedido de integração sendo utilizada por diversos métodos da classe ***BaanIMS***. As propriedades permitem:

atribuir uma descrição ao pedido (***Description***);
definir o tempo máximo para execução do pedido (***MaxTime***);
definir um número máximo de tentativas para a execução do pedido (***MaxRun***);
definir se é guardado o histórico das transições de estado do pedido (***History***);
definir se o ficheiro intermédio da serialização é eliminado (***DelLocFile***).

A classe ***BWA***

BWA
+EndpointInfo : BaanEndpoint +Timeout : int +ReturnValue : string +ReturnCall : string +Error : int
+BWA(in bep : BaanEndpoint) +ParseExecFunction(in dllname : string, in fname : string, in fparam : object[]) +Quit()

Esta classe tem por objectivo encapsular e simplificar a utilização do ***OleAutomationServer*** expondo as propriedades e métodos deste, embora com a assinatura do método ***ParseExecFunction*** alterada. Foi também acrescentada a propriedade ***EndpointInfo***, que expõe a informação do ponto de ligação utilizado.
Esta classe recebe no seu método construtor uma instância da classe ***BaanEndpoint*** que fornece através da sua propriedade ***BwClassName*** a informação necessária para activar o ***OleAutomationServer*** respectivo.

O código das classes deste subsistema pode ser consultado no anexo B.2.

A classe ***BaanEndpoint***

Esta classe é igualmente utilizada pelo subsistema Controlo e Configuração do Acesso, tendo sido já descrita nessa secção.

5.4.4 Diagrama de Instalação

Concluindo a descrição da implementação do subsistema Camada de acesso ao *IMS,* apresenta-se na figura 5.16 o correspondente diagrama de instalação, onde podemos observar os seus principais componentes, ambiente de execução e dispositivos.

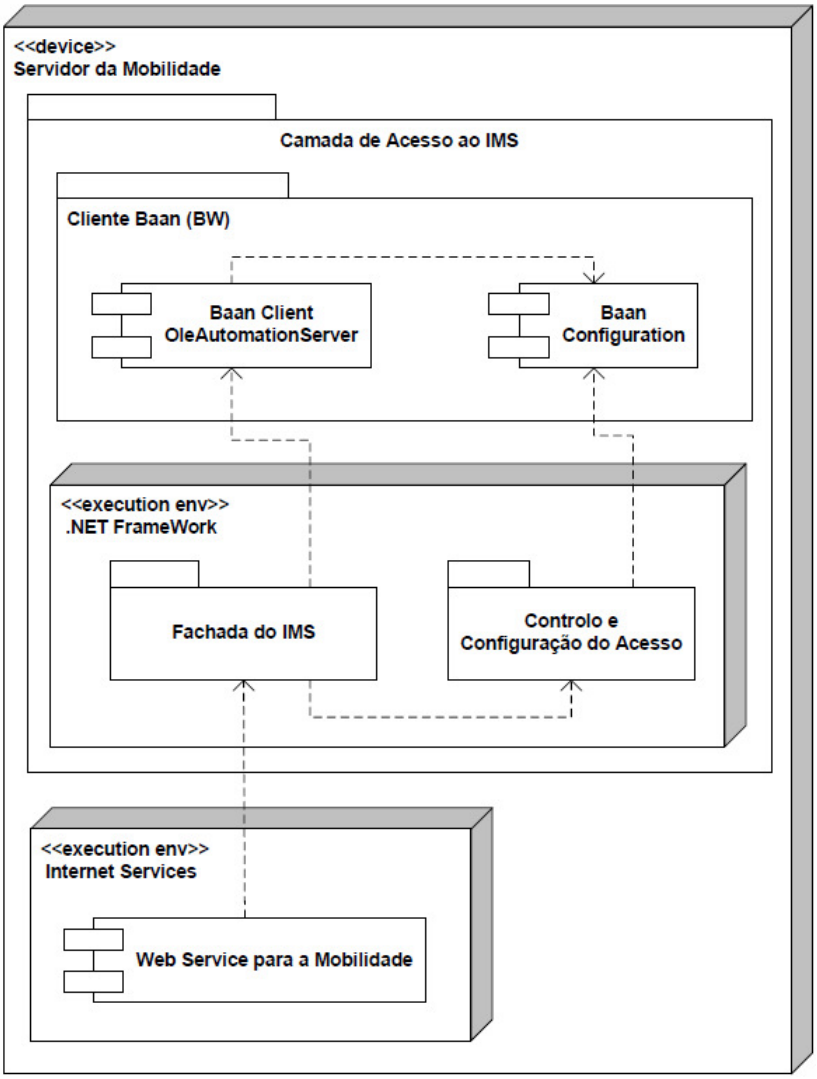


Figura 5.16 – Diagrama de Instalação da Camada de Acesso ao *IMS*

5.5 Classe facilitadora e respectiva decoração com atributos

5.5.1 Objectivo

Com o objectivo de facilitar a utilização de um *Business Object – BO*, desenvolvido no sistema *ERP* Baan, por uma aplicação externa através da plataforma desenvolvida, foi concebido o seguinte esquema:

Para cada *BO* que se pretenda utilizar, deve ser criada uma classe facilitadora que o representa na aplicação externa e cujas propriedades e métodos correspondem respectivamente às variáveis e funções externas existentes no *BO*;

Para que a classe, as propriedades e os métodos possam ter nomes amigáveis e diferentes dos utilizados pelo *BO*, é utilizada a decoração com atributos, de forma a efectuar o necessário mapeamento.

Foi também criada a classe ***BaanIMSBO***, da qual todas as classes facilitadoras devem derivar. Esta classe possui a propriedade ***Errors*** que expõe uma lista necessária ao preenchimento e obtenção das mensagens de erro ocorridas durante a execução do *BO*.

Este método de decoração constitui uma forma declarativa de acrescentar informação acerca do *BO* existente no lado do servidor *ERP* Baan na classe facilitadora utilizada para o representar na aplicação externa. Esta informação pode depois ser obtida em tempo de execução como instâncias de atributos.

A classe ***BaanIMS*** do subsistema Fachada do *IMS* irá levar em conta estes atributos no processo de serialização, decorrente da criação de um pedido de integração.

5.5.2 Classes de atributos criadas

Os atributos são definidos através da declaração de classes que derivem directa ou indirectamente da classe abstracta ***System.Attribute***.

Na implementação desta plataforma foram definidas as seguintes classes de atributos:

BaanIMSRTDDOBJECTAtribute;

BaanIMSFunctionAttribute;

BaanIMSParameterAttribute.

Como podemos observar, as classes de definição de atributos para a decoração das classes facilitadoras estão elas próprias decoradas com atributos a serem também levados em conta no processo de serialização do pedido de integração.

A classe ***BaanIMSRTDDOBJECTAtribute***

É utilizada para decorar os nomes das classes de ajuda, efectuando o mapeamento para o respectivo nome do objecto compilado, existente no *Runtime Data Dictionary - RTDD* do servidor *ERP* Baan. Para além do nome indica qual o seu tipo (ImsBO ou Ims3GL).

```
[XmlRoot("RTDDOBJECT")]
public class BaanIMSRTDDOBJECTAtribute : Attribute
{
    string _otype;
    string _oname;

    public BaanIMSRTDDOBJECTAtribute(string otype, string oname)
    {
        _otype = otype;
        _oname = oname;
    }

    [XmlAttribute("otype")]
    public string ObjType
    {
        get { return _otype; }
    }

    [XmlAttribute("oname")]
    public string ObjName
    {
        get {return _oname;}
    }
}
```

A classe ***BaanIMSFunctionAttribute***

É utilizada para decorar os nomes dos métodos das classes facilitadoras, efectuando o mapeamento para o respectivo nome da função do *BO* existente servidor *ERP* Baan.

```
    [XmlRoot("Function")]
    public class BaanIMSFunctionAttribute : Attribute
    {
        string _fname;

        public BaanIMSFunctionAttribute(string fname)
        {
            _fname = fname;
        }

        [XmlAttribute("fname")]
        public string FuncName
        {
            get {return _fname;}
        }
    }
```

5.5.2 Classes de atributos criadas (cont.)

A classe ***BaanIMSParameterAttribute***

É utilizada para decorar os nomes das propriedades das classes facilitadoras, efectuando o mapeamento para o respectivo nome da variável correspondente utilizada no *BO* existente no servidor *ERP* Baan.

```
[XmlRoot("Parameter")]
public class BaanIMSParameterAttribute : Attribute
{
    string _pname;
    string _pdir;
    string _ptype;

    public BaanIMSParameterAttribute(string pname, string pdir, string ptype)
    {
        _pname = pname;
        _pdir = pdir;
        _ptype = ptype;
    }

    [XmlAttribute("pname")]
    public string ParamName
    {
        get {return _pname;}
    }

    [XmlAttribute("pdir")]
    public string ParamDirection
    {
        get { return _pdir; }
    }

    [XmlAttribute("ptype")]
    public string ParamType
    {
        get { return _ptype; }
    }
}
```

Capítulo 6

Resultados Obtidos

Este capítulo visa descrever as validações efectuadas sobre o trabalho realizado, com o objectivo de verificar a sua adequação ao uso e a sua conformidade com os requisitos funcionais e não funcionais especificados no quarto capítulo. Após a definição da abordagem relativa aos diversos testes a serem realizados, são apresentadas três partes principais que compõem este capítulo. Na primeira parte, é apresentado um exemplo elucidativo da utilização da plataforma que constitui o primeiro teste do sistema como um todo. Na segunda parte é verificada a própria documentação de requisitos, utilizando uma lista de verificação produzida para o efeito, seguindo-se a especificação e resultados da execução dos testes de sistema: funcionais e de carga. Finalmente, na terceira parte, é apresentado o resumo e análise dos resultados obtidos.

6.1 Metodologia

Dada a arquitectura da plataforma e implementação efectuada optou-se pela realização de uma série de testes de sistema com o objectivo de validar a sua adequação ao uso e a sua conformidade com os requisitos funcionais e não funcionais, descritos no quarto capítulo.

Para verificação do correcto funcionamento e adequação da plataforma ao seu objectivo principal, é criado um exemplo elucidativo da sua utilização. Este exemplo expõe todos os passos necessários desde a definição do *Business Object* no lado do servidor *ERP* Baan até ao seu consumo numa aplicação externa, através da plataforma desenvolvida.

Como é compreensível a eficácia dos testes a serem realizados depende em parte da boa concepção tida anteriormente na elaboração da documentação de requisitos. Por essa razão, antes dos testes de sistema, foi feita uma revisão prévia da qualidade dessa documentação baseada numa lista de verificação (*checklist*) composta por um conjunto de atributos.

Os testes funcionais dividiram-se em duas partes: teste das funcionalidades do subsistema *IMS* existente no lado do servidor do sistema *ERP* Baan e teste do subsistema Camada de Acesso ao *IMS* existente no servidor de mobilidade. Os casos de teste foram especificados de forma a exercitar as funções/métodos com valores válidos e inválidos.

A estrutura das especificações de casos de teste apresentada foi inspirada na norma IEEE Std 829-1998 *Standard for Software Test Documentation* [IEEE, 1998a].

Para melhor sistematização e rastreabilidade entre requisitos e especificações de teste, é sempre identificado na especificação do teste o respectivo código e designação do requisito que se está a verificar.

Foram ainda realizados dois testes de carga com objectivos distintos. O primeiro visa verificar o comportamento do sistema perante um cenário em que vai crescendo o número de utilizadores em simultâneo. O segundo visa obter dados de performance, utilizando um padrão de carga de utilização constante. Estes testes têm por objectivo a verificação de alguns requisitos não funcionais.

6.2 Exemplo de utilização da plataforma

Este exemplo compreende a elaboração dos três passos necessários para que uma aplicação externa ao sistema *ERP* Baan possa executar funcionalidades deste através da plataforma desenvolvida.

No passo 1 - Desenvolveu-se um *BO* de teste (**amimsboteste**) que insere o cabeçalho de uma ordem de venda no sistema *ERP* recorrendo ao *Application Function Server - AFS* para executar de forma programática a lógica das sessões envolvidas "Ordens de Venda - tdsls4500m000" (principal) e "Ordens de Venda - tdsls4100s000" (subsessão).

No passo 2 – Definiu-se a classe facilitadora (**OrdemVendaBaan**) que representa o *BO* na aplicação externa e cujas propriedades e métodos correspondem respectivamente às variáveis e funções externas existentes nesse *BO*, conforme o processo e atributos já descritos na secção 5.5 desta dissertação.

No passo 3 – Uma aplicação externa concebida especialmente para este exemplo executa a funcionalidade do *BO* para inserir uma ordem de venda usando a classe facilitadora e a Camada de Acesso ao *IMS*. Após o processamento do pedido de integração, a classe facilitadora fica com a propriedade Numero (nr. ordem de venda) preenchida com o respectivo valor, ou em caso de insucesso a propriedade *Errors* expõe as mensagens de erro devolvidas pela sessão Baan.

Passo 1 - Criação do *Business Object*

Neste ponto são apenas apresentados os blocos de código mais relevantes para a compreensão do exemplo. O código completo deste *BO* pode ser consultado no anexo C.1.

São declaradas as variáveis externas que irão ser os parâmetros de entrada e saída, bem como a função que se pretende utilizar, neste caso, inserir uma ordem de venda:

```
...
14          extern  domain  tccom.bpid      ims.in.ofbp     | Parceiro Cliente
15          extern  domain  tccwoc          ims.in.cofc     | Departamento
16          extern  domain  tccotp          ims.in.sotp     | Tipo de O. Venda
17          extern  domain  tcseri          ims.in.seri     | Série de O. Venda
18          extern  domain  tccdec          ims.in.cdec     | Cond. de Entrega
19          extern  domain  tcemno          ims.in.crep     | Rep. Vendas
20          extern  domain  tcdisc          ims.in.odis     | Desconto de Ordem
21
22          extern  domain  tcorno          ims.out.orno    | Nr. de Ordem de Venda
23          extern  long                    ims.Errors      | Mensagens de Erro
24
25  function extern insert.sales.order.header()
26  {
...
```

É utilizado o *AFS* para executar a lógica da sessão e são preenchidos os campos da sessão com os respectivos valores dos parâmetros de entrada:

```
...
38          ims.out.orno = ""
39
40          ret = stpapi.synchronize.dialog(mainsession, "add", ret_err)
41
42          if ret then
43                  stpapi.put.field(subsession, "tdsls400.ofbp", ims.in.ofbp)
44                  |Inserir endereço?
45                  stpapi.enum.answer(subsession, "tdsls4100.1", tcyesno.no)
46
47                  stpapi.put.field(subsession, "tdsls400.cofc", ims.in.cofc)
...
```

Tenta introduzir o registo do cabeçalho de ordem de venda e no caso de sucesso guarda o número atribuído à ordem de venda na variável externa que representa o parâmetro de saída (*ims.out.orno*):

```
...
66                  ret = stpapi.insert( mainsession, true, last_err)
67
68                  if isspace(last_err) then
69                          stpapi.get.field(mainsession, "tdsls400.orno", ims.out.orno)
70                  else
71                          ret_err = last_err
...
```

No caso de ocorrer algum erro a sua mensagem é adicionada na variável externa criada para este efeito (*ims.Errors*):

```
...
84          amimsxmlhand.insert.error(ret_err, ims.Errors)
85  }
```

Passo 2 - Criação da Classe decorada com os atributos

Neste passo é criada a classe facilitadora **OrdemVendaBaan** que representa o *Business Object* na aplicação externa. O código completo desta classe pode ser consultado no anexo C.2.

A classe **OrdemVendaBaan** é decorada com o atributo do tipo ***BaanIMSRTDDOBJECTAtribute***. Define-se que se trata de um *BO* ("ImsBO") cujo objecto compilado no servidor *ERP* Baan é "oamimsboteste" (o objecto compilado possui o prefixo "o"):

```
[BaanIMSRTDDOBJECTAtribute("ImsBO", "oamimsboteste")]
  public class OrdemVendaBaan : BaanIMSBO
  {
...
```

O método é decorado com o atributo do tipo ***BaanIMSFunctionAttribute*** e define-se que este representa a função "insert.sales.order.header" do *BO*:

```
...
      [BaanIMSFunctionAttribute("insert.sales.order.header")]
      public void InsereCabecalho()
      {}
...
```

As propriedades são decoradas com o atributo do tipo ***BaanIMSParameterAttribute*****,** definindo-se nos seus argumentos o nome da variável externa do *BO,* se esta é de entrada ou saída ("*in*" ou "*out*") e qual o seu tipo de dados.

```
...
      [BaanIMSParameterAttribute("ims.in.ofbp", "in", "string")]
      public string ParceiroCliente
      {
          get { return _parceirocliente; }
          set { _parceirocliente = value; }
      }
...
```

A propriedade Numero é decorada como correspondendo à variável externa "ims.out.orno" do *BO* e é utilizada como parâmetro de saída ("*out*"):

```
...
      [BaanIMSParameterAttribute("ims.out.orno", "out", "string")]
      public string Numero
      {
          get { return _numero; }
          set { _numero = value; }
      }
}
```

Passo 3 – Utilização do *Business Object* pela aplicação externa

Nesta parte é demonstrado como o *BO* criado no servidor *ERP* Baan pode ser utilizado por uma aplicação externa recorrendo à plataforma desenvolvida utilizando a classe **OrdemVendaBaan** para facilitar a interacção. Para a demonstração foi desenvolvida uma aplicação cujo ecrã está representado na figura 6.1.

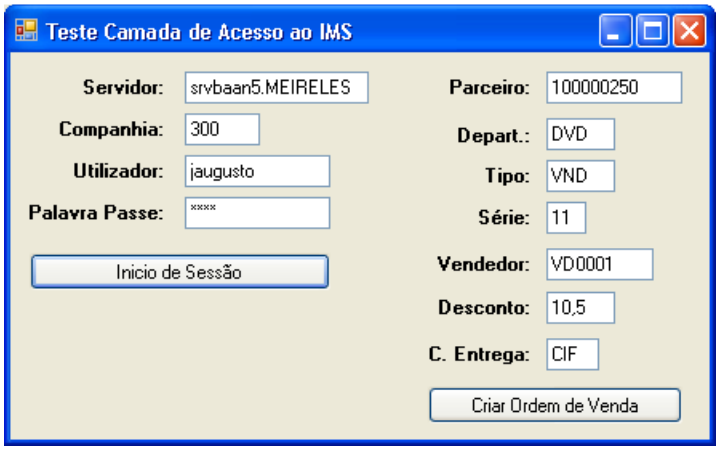


Figura 6.1 – Ecrã da aplicação de utilização do *Business Object*

Como se pode constatar, estão presentes no ecrã da aplicação não só os campos relativos à utilização do *BO* para criar uma ordem de venda, mas também a informação necessária para estabelecer a ligação de um utilizador, numa companhia e servidor *ERP* Baan. Esta aplicação utiliza os subsistemas "Controlo e Configuração do Acesso" e "Fachada do IMS" da "Camada de Acesso ao *IMS*".

Apresenta-se de seguida a explicação das partes mais relevantes do código desta aplicação para a explicação da utilização do *BO* e do controlo do acesso.

Quando o botão "Criar Ordem de Venda" é accionado cria-se uma instância da classe **OrdemVendaBaan** e são preenchidas as suas propriedades com os respectivos valores dos campos do ecrã:

```
...
    OrdemVendaBaan ov = new OrdemVendaBaan();

    ov.ParceiroCliente = txParceiro.Text;
    ov.Departamento = txDepartam.Text;
    ov.Tipo = txTipo.Text;
    ov.Serie = txSerie.Text;
    ov.Vendedor = txVendedor.Text;
    ov.Desconto = Convert.ToDouble(txDesconto.Text);
    ov.CondEntrega = txCEnt.Text;
...
```

Passo 3 – Utilização do *Business Object* pela aplicação externa (cont.)

É também criada uma instância da classe ***IMSCallDetails*** e preenchidas as suas propriedades que definem as características do pedido de integração:

```
…
    IMSCallDetails imc = new IMSCallDetails();

    imc.Description = "Teste O.Venda";
    imc.MaxTime = 30;
    imc.MaxRun = 3;
    imc.History = true;
    imc.DelLocFile = false;
…
```

Finalmente, é utilizado o subsistema "Fachada do *IMS*". Cria-se uma instância da classe ***BaanIMS*** e utiliza-se o método ***CreateExecuteAndGetResult*** para criar, executar e obter o resultado do pedido de integração numa única chamada, passando como argumentos de entrada a classe facilitadora (**ov**) e a classe caracterizadora do pedido (**imc**), criadas anteriormente:

```
…
    try
    {
        BaanIMS bims = new BaanIMS(be, imstempdir);

        bims.CreateExecuteAndGetResult(ov, imc, out req, out execret,
out gresret);
…
```

Nota: a classe *BaanIMS* oferece métodos que permitem desacoplar o pedido do resultado e esta operação poderia ter sido feita nos seus três passos elementares (criar, executar e obter o resultado) separadamente.

Após a execução, a classe facilitadora expõe nas propriedades que representam os parâmetros de saída os valores resultantes do processamento do pedido de integração, neste caso, o número de ordem de venda registado no sistema *ERP* Baan é mostrado. Podem também ser mostradas as mensagens de erro ocorridas durante a execução do *BO* recorrendo à propriedade ***Errors*** exposta pela classe facilitadora:

```
…
    MessageBox.Show(ov.Numero);

    foreach (string err in ov.Errors)
        MessageBox.Show(err);
…
```

Em relação ao controlo do acesso, no método construtor da classe *Form1* é criado o catálogo de pontos de ligação (***bec***) através da invocação de método ***CreateCatalog*** da classe ***BaanEndpointCatFactory*** (implementação do padrão *Method Factory*):

```
...
IBaanEndpointCat bec;
...
    BwConfigManager bwcman = new BwConfigManager(bwcdir);
    BaanEndpointCatFactory becfact = new BaanEndpointCatFactory(bwcman);

    bec = becfact.CreateCatalog();
...
```

Quando o botão "Inicio de Sessão" é accionado verifica-se a existência de um ponto de ligação no catálogo (***bec***) para a combinação servidor, companhia e utilizador usada. Se existir é verificada a palavra passe:

```
...
try
{
    be = bec.GetEndpoint(txsrv1.Text, txcomp1.Text, txuser1.Text);

    if (be == null)
        MessageBox.Show("Combinação Serv./Comp./Utiliz. Inválida");
    else
    {
       if (be.Password != txpassw1.Text)
       {
           MessageBox.Show("Password Errada");
           btCriarOV.Enabled = false;
       }
       else
           btCriarOV.Enabled = true;
    }
...
```

Em suma, ao desencadear os eventos associados aos botões "Inicio de Sessão" e "Criar Ordem de Venda", com os valores do ecrã presentes na figura 6.1, temos como resultado a respectiva criação de uma ordem de venda no sistema *ERP* Baan e o aparecimento da mensagem com o número atribuído a esta, consoante podemos verificar na figura 6.2.

Figura 6.2 – Mensagem com o número de ordem de venda criado

Passo 3 – Utilização do *Business Object* pela aplicação externa (cont.)

No lado do sistema *ERP* Baan pode-se constatar pela observação das figuras 6.3 e 6.4 que os valores criados na ordem de venda referida estão de acordo com os valores lançados pela aplicação externa.

A figura 6.3 apresenta a verificação dos campos Parceiro, Departamento, Número, Tipo, Vendedor (Rep. 1) e Condição de Entrega.

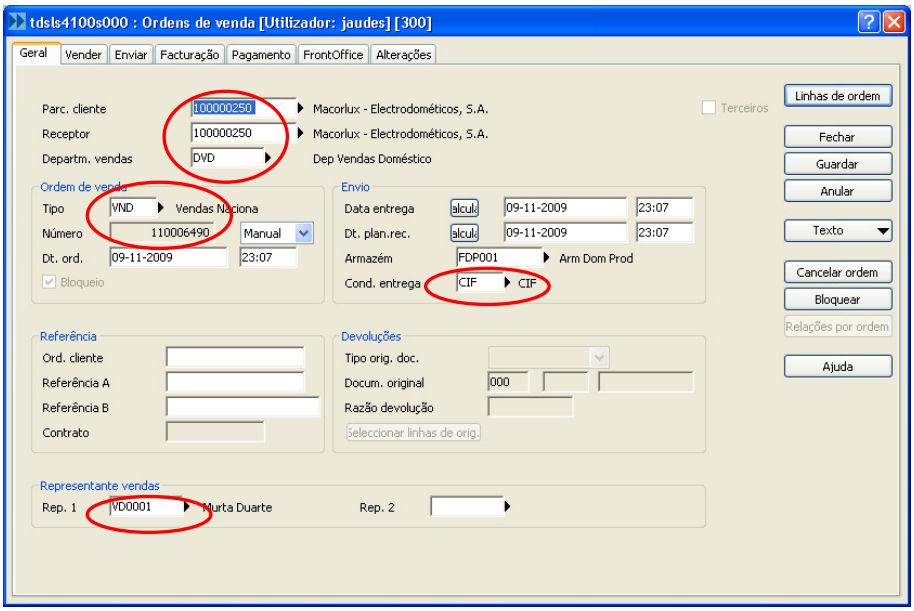


Figura 6.3 – Sessão Ordens de Venda do sistema *ERP* Baan: Ecrã Geral

O campo Desconto da Ordem pode ser verificado na figura 6.4.

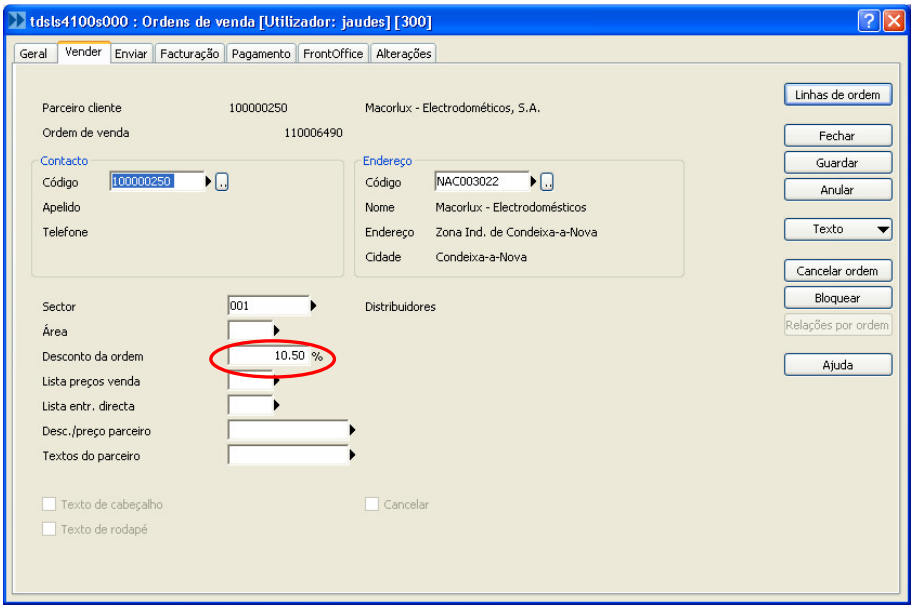


Figura 6.4 – Sessão Ordens de Venda do sistema *ERP* Baan: Ecrã Vender

Passo 3 – Utilização do *Business Object* pela aplicação externa (cont.)

No seguinte exemplo tenta-se criar uma ordem de venda com um código de parceiro cliente inexistente ("Z00000000"), conforme representado na figura 6.5.

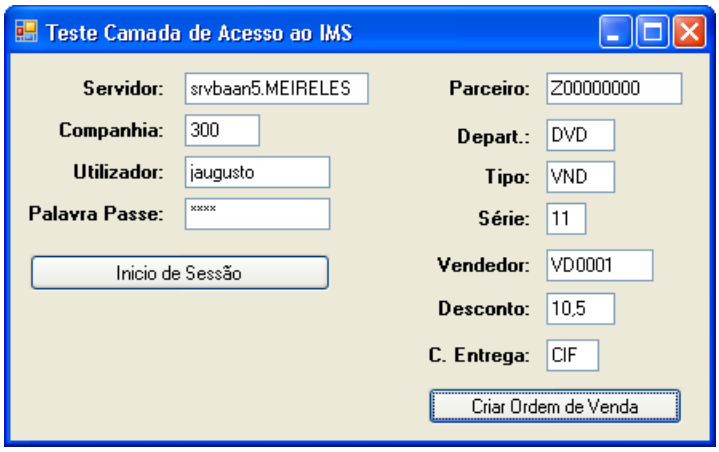


Figura 6.5 - Ecrã da aplicação preenchido com parceiro incorrecto

Nesta situação, o resultado do processamento do pedido de integração não resulta na criação de uma ordem de venda, mas no surgimento da mensagem de erro ocorrido na execução do *BO* e exposto pela propriedade ***Errors*** da classe facilitadora (**ov**). A figura 6.6 mostra o resultado obtido desta execução.

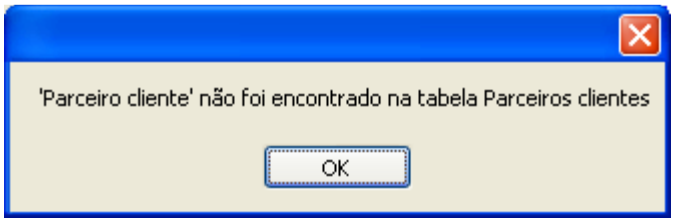


Figura 6.6 – Mensagem de erro devolvida pela execução do *BO*

O código fonte completo da aplicação utilizada neste exemplo pode ser consultado no anexo C.6.

6.3 Revisão da documentação de Requisitos

Na tabela 6.1 podemos observar a lista de atributos que serviram de base à verificação da documentação de requisitos. Esta lista de verificação foi adaptada do livro "S*oftware Testing*" de Ron Patton [Patton, 2001].

Atributo	Considerar	Resultado	Observações
Completude	Falta algum aspecto ou informação?	OK	
	Nível de detalhe suficiente?	OK	
Exactidão	A solução proposta está correcta?	OK	
	Define correctamente o objectivo?	OK	
	Contém erros?	OK	
Preciso, claro e sem ambiguidade	A descrição é exacta e não vaga?	OK	
	Tem apenas uma interpretação?	OK	
	É fácil de ler e entender?	OK	
Consistência	As funcionalidades estão descritas de forma consistente?	OK	
	As funcionalidades estão descritas de forma a não entrarem em conflito internamente nem com outros itens da especificação?	OK	
Relevância	As funcionalidades estão mapeadas com as necessidades?	OK	
Testável	É possível testar as funcionalidades?	OK	Cada requisito específico pretence a um grupo de funcionalidades e cada caso de teste refere o identificador do requisito.
	Existe informação suficiente relativamente às funcionalidades para criar casos de teste?	OK	

Tabela 6.1 – Lista para verificação dos requisitos

6.4 Testes de Sistema

6.4.1 Testes funcionais do subsistema IMS

Dado que as funcionalidades do subsistema *IMS* são essencialmente expostas pela *DLL* amimsdll00001 e por um conjunto de sessões Baan, os testes funcionais incidiram sobre estes itens.

Como o sistema *ERP* Baan não possui uma *framework* de testes, foi criado um programa 3GL para exercitar as funções da *DLL* de acordo com as várias especificações de teste criadas. O código deste programa pode ser consultado no anexo C.5.

Teste da *DLL* amimsdll00001

Para exercitar as várias funções desta *DLL,* foram criadas especificações de casos de teste que orientaram a sua execução e cuja estrutura permitiu uma fácil rastreabilidade entre o teste, requisito funcional a ser verificado e o resultado obtido. A sua estrutura pode ser observada na tabela 6.2, que constitui um exemplo dos testes de sistema funcionais realizados. Os restantes casos de teste especificados para esta *DLL* podem ser consultados no anexo D.1.

Identificador de Teste:	**TS1**				
Requisito a testar:	RF01	Criar pedido de integração			
ID do Caso de Teste:	**TS1_1**				
Objectivo:	Exercitar a função de criar pedido, utilizando valores válidos.				
Procedimento:	Efectuar uma chamada à função **create.request** com os valores de entrada especificados e verificar o valor de retorno da função. Verificar com a sessão amims0501m000 se os dados de controlo do pedido estão de acordo com os parâmetros especificados nos valores de entrada.				
Valores de Entrada:	("c:\ims\testes\BaanIMS.xml", "Teste TS1_1", 5, 3, 1, 0)				
Resultado Esperado:	> 0	**Resultado Obtido:**	Valor de retorno: 1402 Verificação com a sessão amims0501m000: OK		
		Resultado:	OK	**Defeito nr.:**	

Teste da DLL amimsdll00001 (cont.)

ID do Caso de Teste:	**TS1_2**				
Objectivo:	Exercitar a função de criar pedido, utilizando valores inválidos (1º argumento).				
Procedimento:	Efectuar uma chamada à função **create.request** com os valores de entrada especificados e verificar o valor de retorno da função.				
Valores de Entrada:	(**“”**, “Teste TS1_2”, 5, 3, 1, 0)				
Resultado Esperado:	0	**Resultado Obtido:**	0		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS1_3**				
Objectivo:	Exercitar a função de criar pedido, utilizando valores inválidos (3º argumento).				
Procedimento:	Efectuar uma chamada à função **create.request** com os valores de entrada especificados e verificar o valor de retorno da função.				
Valores de Entrada:	(“c:\ims\testes\BaanIMS.xml”, “Teste TS1_3”, **-1**, 3, 1, 0)				
Resultado Esperado:	0	**Resultado Obtido:**	0		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS1_4**				
Objectivo:	Exercitar a função de criar pedido, utilizando valores inválidos (4º argumento).				
Procedimento:	Efectuar uma chamada à função **create.request** com os valores de entrada especificados e verificar o valor de retorno da função.				
Valores de Entrada:	(“c:\ims\testes\BaanIMS.xml”, “Teste TS1_4”, 5, **-1**, 1, 0)				
Resultado Esperado:	0	**Resultado Obtido:**	0		
		Resultado:	OK	**Defeito nr.:**	

Tabela 6.2 – Especificação do teste de sistema TS1

Teste das Sessões do subsistema *IMS*

As sessões foram igualmente verificadas recorrendo à especificação de casos de teste baseados na estrutura anterior. A tabela 6.3 apresenta o exemplo da verificação efectuada na sessão que processa os pedidos de integração com o estado "Espera" (pendentes de execução). Os casos de teste das restantes sessões podem ser consultados no anexo D.2.

<table>
<tr><td>Identificador de Teste:</td><td>TS6</td><td colspan="4"></td></tr>
<tr><td>Requisito a testar:</td><td>RF06</td><td colspan="4">Sessão para tratamento de pedidos de integração pendentes</td></tr>
<tr><td>ID do Caso de Teste:</td><td colspan="5">TS6_1</td></tr>
<tr><td>Objectivo:</td><td colspan="5">Exercitar a funcionalidade da sessão para processamento de pedidos pendentes.</td></tr>
<tr><td>Procedimento:</td><td colspan="5">Iniciar a sessão amims0201m000, preencher os campos com os valores de entrada e accionar a opção continuar. Escolher um intervalo de números de pedido que inclua não só o estado "Espera" mas também outros estados. Após a execução verificar com a sessão amims0501m000 se os pedidos seleccionados com o estado "Espera" efectuaram uma transição válida e se o estado dos restantes não se alterou.</td></tr>
<tr><td>Valores de Entrada:</td><td colspan="5">Pedido De: 1353 A: 1358</td></tr>
<tr><td rowspan="2">Resultado Esperado:</td><td rowspan="2">Os pedidos com estado "Espera" devem realizar uma transição válida. Os restantes não devem alterar o seu estado.</td><td>Resultado Obtido:</td><td colspan="3">Os pedidos com estado "Espera" passaram para o estado "Executado". Os restantes não se alteraram.</td></tr>
<tr><td>Resultado:</td><td>OK</td><td>Defeito nr.:</td><td></td></tr>
</table>

Tabela 6.3 – Especificação do teste de sistema TS6

Resultados dos testes funcionais do subsistema *IMS*

A tabela 6.4 mostra os resultados dos testes funcionais realizados à DLL amimsdll00001 e às sessões que compõem o subsistema *IMS*.

ID do Teste	ID do Requisito	Descrição do Requisito Testado	Casos de Teste Especificados				
			Nr. de Casos	Sucesso	Perc.	Falharam	Perc.
TS1	RF01	Criar Pedido de integração	4	4	100%	0	0%
TS2	RF02	Executar pedido de integração	6	6	100%	0	0%
TS3	RF03	Colocar pedido de integração no estado "Espera"	3	3	100%	0	0%
TS4	RF04	Consultar estado de um pedido de integração	2	2	100%	0	0%
TS5	RF05	Obter resultado de um pedido de integração	4	4	100%	0	0%
TS6	RF06	Sessão para tratamento de pedidos de integração pendentes	1	1	100%	0	0%
TS7	RF07	Criar e executar pedido de integração numa única chamada	4	4	100%	0	0%
TS8	RF08	Criar e colocar o pedido em Espera numa única chamada	4	4	100%	0	0%
TS10	RF10	Sessão que permita a gestão dos pedidos de integração de forma visual	4	4	100%	0	0%
TS11	RF11	Validador de transições de estado do pedido de integração	2	2	100%	0	0%
TS12	RF12	Lançamento e monitorização do processo de execução	4	4	100%	0	0%
TS13	RF13	Processar pedido de integração	3	3	100%	0	0%
TS14	RF14	Histórico das transições de estado dos pedidos de integração	1	1	100%	0	0%
TS15	RF15	Sessão de consulta de histórico dos pedidos de integração	1	1	100%	0	0%
TS16	RF16	Parametrização do tempo máximo pré-definido para a execução dos pedidos de integração	1	1	100%	0	0%
TS17	RF17	Parametrização da localização do repositório XML de pedidos e resultados de integração	1	1	100%	0	0%
TS18	RF18	Sessão para manutenção de parâmetros do módulo *IMS*	1	1	100%	0	0%
		Totais	**46**	**46**	**100%**	**0**	**0%**

Tabela 6.4 – Resultados dos testes funcionais do módulo *IMS*

6.4.2 Testes funcionais da Camada de Acesso ao IMS

As verificações incidiram essencialmente no subsistema "Fachada do *IMS*", mais concretamente na sua classe principal *BaanIMS*. Esta classe expõe as funcionalidades do módulo *IMS* e foi desenvolvida utilizando um ambiente integrado de desenvolvimento (*Visual Studio Team Suite*) que possui uma *framework* de testes, permitindo automatizar os casos de teste especificados para o módulo *IMS,* salvaguardando as respectivas diferenças de paradigma de programação. Para manter a rastreabilidade, os testes foram identificados com o código do caso de teste referido na especificação, como se pode observar na figura 6.7.

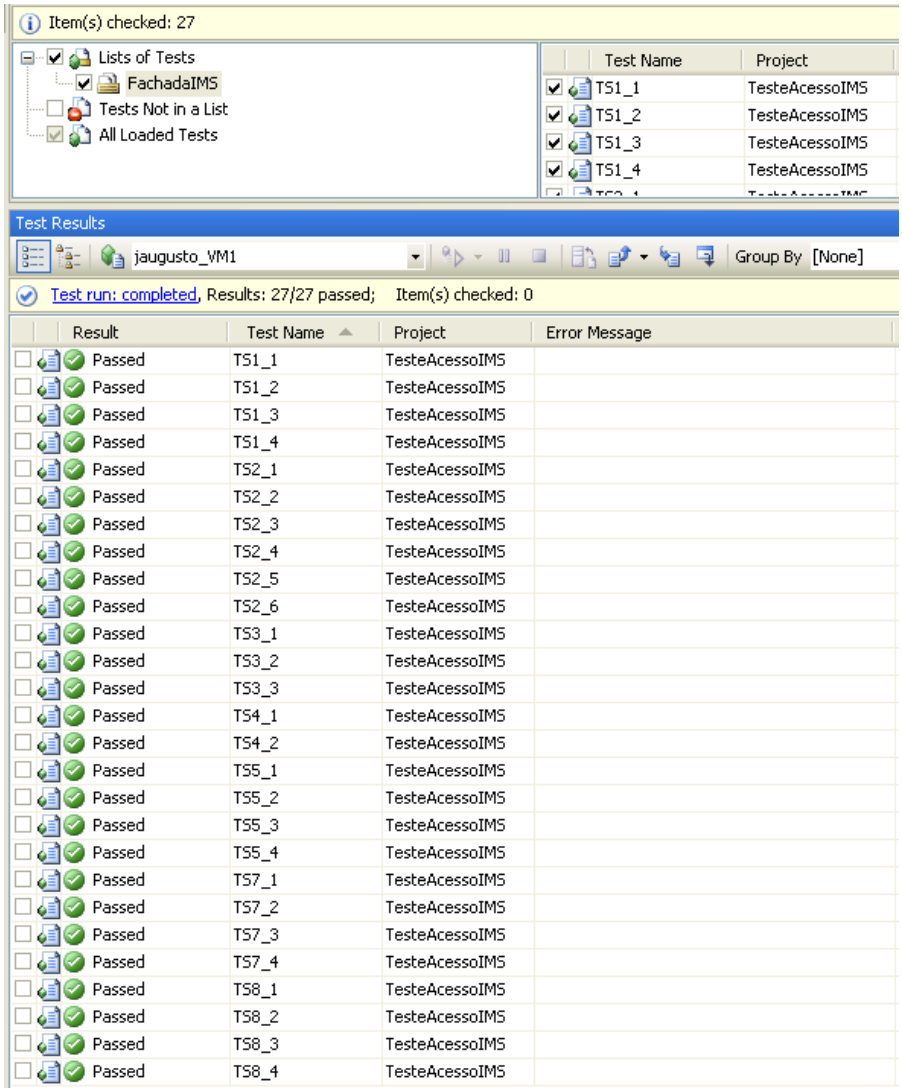


Figura 6.7 – Testes realizados à classe BaanIMS e respectivos resultados

Resultados dos testes funcionais do subsistema Fachada do *IMS*

A tabela 6.5 apresenta os resultados dos testes funcionais realizados ao subsistema Fachada do *IMS*.

ID do Teste	ID do Requisito	Descrição do Requisito Testado	Casos de Teste Especificados				
			Nr. de Casos	Sucesso	Perc.	Falharam	Perc.
TS1	RF01	Criar Pedido de integração	4	4	100%	0	0%
TS2	RF02	Executar pedido de integração	6	6	100%	0	0%
TS3	RF03	Colocar pedido de integração no estado "Espera"	3	3	100%	0	0%
TS4	RF04	Consultar estado de um pedido de integração	2	2	100%	0	0%
TS5	RF05	Obter resultado de um pedido de integração	4	4	100%	0	0%
TS7	RF07	Criar e executar pedido de integração numa única chamada	4	4	100%	0	0%
TS8	RF08	Criar e colocar o pedido em Espera numa única chamada	4	4	100%	0	0%
		Totais	**27**	**27**	**100%**	**0**	**0%**

Tabela 6.5 – Resultados dos testes funcionais do subsistema Fachada do *IMS*

6.4.3 Testes de Carga

Teste de Carga TC1

O primeiro teste de carga teve por objectivo analisar o comportamento do sistema perante um cenário em que vai crescendo o número de utilizadores em simultâneo, até um valor máximo especificado de 200. O teste utilizado na carga foi o TS1_1 – "Criar Pedido de Integração", tendo em vista a análise dos dados para verificação dos requisitos não funcionais RNF04 e RNF05. Os resultados obtidos podem ser observados nas figuras 6.8 e 6.9. A figura 6.10 permite verificar o incremento da carga de utilização ao longo do tempo.

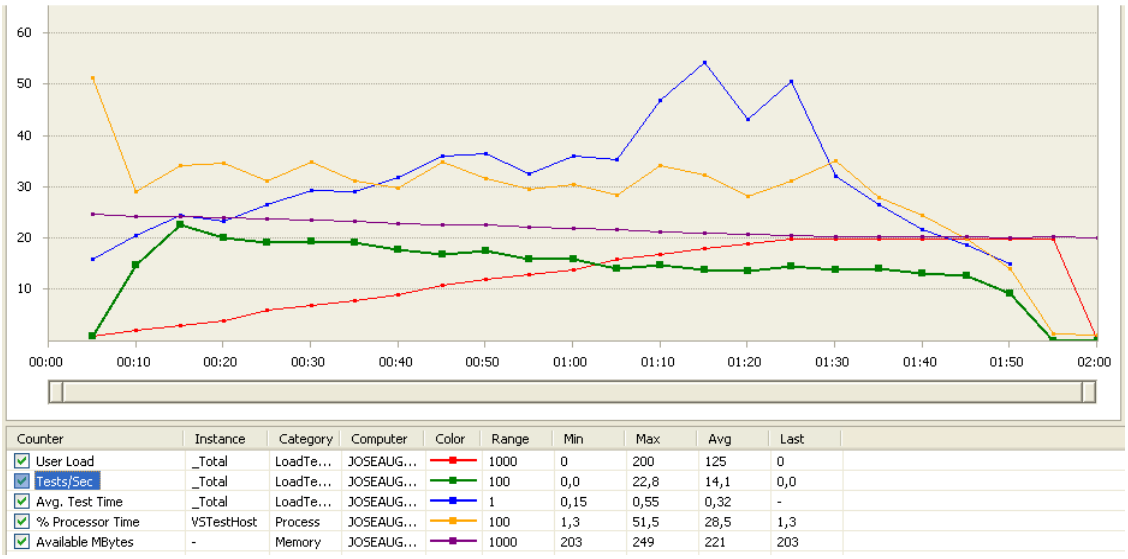


Figura 6.8 – Teste de carga TC1

Tests/Sec	
Time	Value
-	-
00:05	1,0
00:10	15,0
00:15	22,8
00:20	20,2
00:25	19,4
00:30	19,6
00:35	19,4
00:40	18,0
00:45	17,0
00:50	17,6
00:55	16,2
01:00	16,2
01:05	14,2
01:10	15,0
01:15	14,0
01:20	13,8
01:25	14,6
01:30	14,0
01:35	14,2
01:40	13,2
01:45	12,8
01:50	9,4
01:55	0,0
02:00	0,0

Figura 6.9 – Número de testes/segundo no teste de carga TC1

User Load	
Time	Value
-	-
00:05	11
00:10	21
00:15	31
00:20	41
00:25	61
00:30	71
00:35	81
00:40	91
00:45	111
00:50	121
00:55	131
01:00	141
01:05	161
01:10	171
01:15	181
01:20	191
01:25	200
01:30	200
01:35	200
01:40	200
01:45	200
01:50	200
01:55	200
02:00	0

Figura 6.10 – Carga de utilização no teste de carga TC1

6.4.3 Testes de Carga (cont.)

Teste de Carga TC2

O segundo teste de carga visa obter dados de performance, utilizando um padrão de carga de utilização constante, com um valor especificado de 25. O teste utilizado na carga foi igualmente o TS1_1 – "Criar Pedido de Integração", tendo em vista a análise dos dados para verificação dos requisitos não funcionais RNF04 e RNF05. Este teste foi realizado em horário laboral (15:52) e os resultados obtidos podem ser observados nas figuras 6.11 e 6.12.

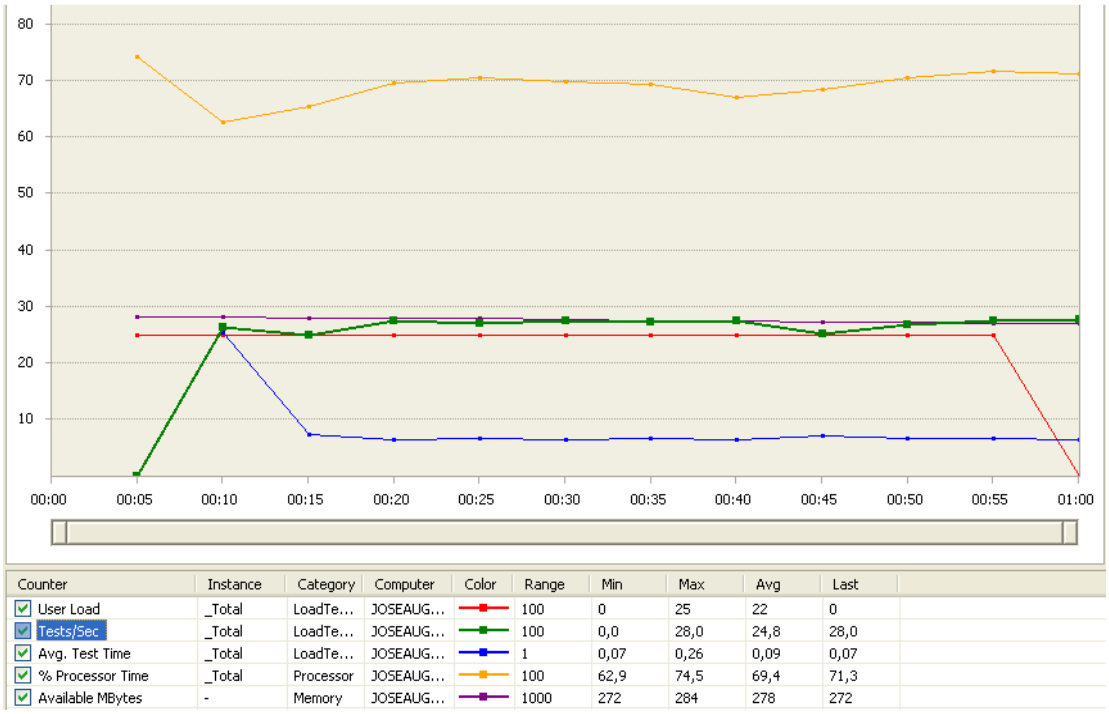


Figura 6.11 – Teste de carga TC2

Tests/Sec

Time	Value
-	-
00:05	0,0
00:10	26,6
00:15	25,2
00:20	27,6
00:25	27,2
00:30	27,6
00:35	27,4
00:40	27,6
00:45	25,4
00:50	27,0
00:55	27,6
01:00	28,0

Figura 6.12 – Número de testes/segundo no teste carga TC2

6.5 Resumo e Análise dos Resultados Obtidos

Testes Funcionais de Sistema

A tabela 6.6 apresenta o resumo dos resultados da execução dos testes funcionais de sistema.

Subsistema Principal	Componente / Subsistema	Casos de Teste Especificados				
		Nr. de Casos	Sucesso	Perc.	Falharam	Perc.
Integration Management System IMS	*DLL* amimsdll00001	39	39	100%	0	0%
	Sessões Baan do *IMS*	7	7	100%	0	0%
Camada de Acesso ao *IMS*	Fachada do *IMS*	27	27	100%	0	0%
	Totais	**73**	**73**	**100%**	**0**	**0%**

Tabela 6.6 – Resumo de resultados dos testes funcionais de sistema

Como é óbvio, a actividade de teste permite detectar a presença de defeitos, mas não a ausência destes. Os resultados verificados permitem concluir que os componentes/subsistemas da plataforma apresentam um bom índice de conformidade com os requisitos especificados e denotam um correcto funcionamento da plataforma. Porém é de equacionar a elaboração de mais casos de teste com situações ainda não cobertas pelos 73 casos apresentados.

O subsistema Controlo e Configuração do Acesso, existente na Camada de Acesso ao *IMS*, não foi testado de forma sistemática como os restantes subsistemas. Porém, de forma indirecta foi realizado um teste primário da sua adequação ao uso. Se este subsistema não cumprisse a sua funcionalidade, não seria possível estabelecer uma ligação ao servidor *ERP* Baan e consequentemente não seria possível realizar os testes do subsistema Fachada do *IMS*, situação que não ocorreu.

Testes de Carga

Relativamente aos resultados obtidos nos dois testes de carga (TC1 e TC2) expostos na secção 6.4.3 podemos concluir que a plataforma desenvolvida satisfaz os requisitos não funcionais relativos quer ao número mínimo de utilizadores em simultâneo suportado (RNF04) quer ao tempo máximo para criação do pedido de integração (RNF05).

O RNF04 especifica que a plataforma deverá suportar no mínimo 25 utilizadores em simultâneo, sendo esse valor cumprido no teste TC2 e claramente ultrapassado no TC1, em oito vezes superior.

O RNF05 especifica que relativamente à criação de um pedido de integração, 95% das transacções deverão ser concluídas em menos de 2 segundos em horário normal de trabalho. Os restantes 5% não deverão exceder os 10 segundos. Como se pode verificar no TC2 com um padrão de carga constante de 25 utilizadores e realizado em horário laboral, o tempo médio de criação do pedido de integração foi de 0,09 segundos, tendo atingido um valor máximo de 0,26 segundos o que é largamente satisfatório. Mesmo com uma carga simulada que atingiu os 200 utilizadores no TC1 o valor máximo de criação do pedido foi de 0,55 segundos.

Um outro aspecto a realçar destes testes de carga é a estabilidade apresentada durante a sua execução em termos de % do tempo do processador e da memória RAM disponível.

Outras Verificações

Finalmente, cabe nesta parte apresentar a conformidade relativa aos restantes requisitos não funcionais.

O RNF01 (Preservação dos níveis de acesso do sistema *ERP* Baan) está validado uma vez que a implementação da Camada de Acesso ao *IMS* reutilizou a forma de aceder ao sistema *ERP* Baan através do programa cliente (BW).

O RNF02 (Optimização do uso das licenças de acesso ao sistema ERP Baan) está validado pela forma de funcionamento do servidor de automação OLE do *software* cliente de acesso ao sistema *ERP* Baan.

O RNF03 (Disponibilidade da plataforma equivalente à do sistema *ERP*) está validado pelo facto do subsistema *IMS* ser implementado no próprio sistema *ERP* Baan.

O RNF06 especifica que a linguagem de desenvolvimento da camada de acesso ao *IMS* deve ser o C#. Está validado uma vez que foi esta a linguagem de programação utilizada.

Capítulo 7

Conclusões e Trabalho Futuro

7.1 Conclusões

As conclusões relativas a este trabalho de dissertação irão ser efectuadas em três partes. A primeira parte diz respeito à fase de estudo do estado da arte realizada nos capítulos dois e três. Na segunda parte serão tecidos alguns comentários relacionados com aspectos da arquitectura do sistema concebido e da implementação realizada. Finalmente, a terceira parte diz respeito à eficácia da metodologia adoptada e análise dos resultados obtidos, terminando com algumas considerações de carácter global.

O estudo inicial realizado no capítulo dois permitiu enquadrar de forma eficaz as principais características dos sistemas *ERP* em geral, e principalmente os aspectos mais específicos do sistema *ERP* Baan, como a sua arquitectura e as tecnologias decisivas para o desenvolvimento da plataforma. O estudo do software *ERP* Baan como servidor de automação OLE permitiu a correcta implementação dos subsistemas "Controlo e Configuração do Acesso" e "Fachada do *IMS*". O estudo do *Application Function Server – AFS* contribuiu para o desenvolvimento do mecanismo de execução dos pedidos de integração no acesso à lógica das sessões Baan de forma programática. O estudo da *API* de processamento *XML* em programas 3GL do sistema Baan, permitiu o correcto tratamento dos pedidos e resultados de integração num formato interoperável. O terceiro capítulo forneceu uma breve perspectiva das principais abordagens no desenho da arquitectura de aplicações ocasionalmente conectadas e apresentou, com detalhe, uma forma de suportar o modo ocasionalmente conectado recorrendo a boas práticas e padrões fornecidos pela *Smart Client Software Factory – SCSF*. Este estudo foi fundamental tendo em vista o trabalho futuro de desenvolvimento do protótipo da solução de mobilidade que irá utilizar as funcionalidades da plataforma desenvolvida.

A arquitectura proposta revelou-se adequada, permitindo que a plataforma apresente algumas características importantes de salientar nesta fase, nomeadamente:

- a utilização de protocolos e tecnologias normalizadas;
- responde a cenários com vários servidores, utilizadores e companhias de sistema *ERP* Baan;
- reutiliza mecanismos de comunicação e autenticação;
- preserva os níveis de acesso existentes no sistema *ERP*;
- acompanha a cobertura do sistema *ERP* em relação aos sistemas operativos e bases de dados suportados;
- tira partido da escalabilidade do sistema *ERP*.

A plataforma desenvolvida oferece um conjunto de funcionalidades que possibilitam uma gestão mínima requerida no tratamento das questões associadas a processos que lançam e modificam dados no sistema *ERP* e consequentemente com impacto no negócio da organização.

A aplicação de alguns padrões de desenho na fase de implementação permitiu que os diversos subsistemas ficassem mais robustos e flexíveis relativamente a alterações futuras. Por exemplo a aplicação do padrão *Factory Method* no subsistema de Controlo e Configuração do Acesso permite que a informação relativa aos pontos de ligação possa futuramente ser proveniente de diversas origens (base de dados, Web Services, serviços de directório), de forma transparente aos restantes componentes do subsistema.

Relativamente à metodologia adoptada, a divisão do trabalho de dissertação em três fases principais revelou-se eficaz, permitindo atingir os objectivos inicialmente propostos.

Na fase de determinação dos requisitos da plataforma, a estruturação inspirada na norma IEEE Std 830-1998 *Recommended Practice for Software Requirements Specifications* [IEEE, 1998b] foi extremamente produtiva e eficaz na sistematização e elaboração desses requisitos. Esta abordagem permitiu igualmente uma fácil rastreabilidade entre os requisitos, implementação e resultados obtidos, conforme foi demonstrado ao longo do trabalho. O enquadramento dos requisitos em grupos de funcionalidade e a definição da sua prioridade revelaram-se eficazes ao orientar e segmentar as tarefas sequentes de desenvolvimento.

Na fase de desenvolvimento, a abordagem segundo o modelo iterativo e incremental, ofereceu diversas vantagens particularmente úteis em relação à plataforma concebida, nomeadamente: a robustez da arquitectura foi testada cedo; os diversos artefactos desenvolvidos foram validados em cada iteração; o modelo centrado nas funcionalidades e na arquitectura do sistema permitiu sistematizar a criação dos diversos subsistemas de forma eficaz. É igualmente de realçar que o desenvolvimento orientado pela prioridade dos requisitos permitiu uma boa articulação com a lógica de incremento de funcionalidades em sucessivas iterações, até à obtenção do produto final. Para terminar os comentários relativos à metodologia, importa ainda referir que a abordagem relativa à validação do trabalho desenvolvido permitiu uma apresentação quantificável e objectiva dos resultados obtidos. Para este resultado foi importante o contributo fornecido pela análise da norma IEEE Std 829-1998 *Standard for Software*

Test Documentation [IEEE, 1998a], permitindo que a especificação dos casos de teste fosse realizada de forma mais eficaz e eficiente.

Finalmente, os resultados e análise apresentados na secção 6.5 desta dissertação, bem como o exemplo de utilização fornecido na secção 6.2, permitem concluir que a plataforma desenvolvida dá resposta aos objectivos inicialmente propostos e demonstram o seu correcto funcionamento.

Como considerações finais importa referir que o tipo de resultados produzidos no âmbito deste trabalho de dissertação vai de encontro às necessidades das organizações em manter os seus sistemas de informação mais abrangentes e capazes de responder à crescente tendência verificada para a mobilidade da força de trabalho.

Num âmbito mais alargado, a flexibilidade e a rapidez com que os sistemas de informação empresariais respondem ao ambiente externo cada vez mais competitivo e exigente, irá ditar o seu sucesso na missão de constituírem mais-valias competitivas para as organizações. Neste ponto, a existência de infra-estruturas como as produzidas no âmbito deste trabalho de dissertação irá contribuir seguramente para essa flexibilidade e rapidez.

7.2 Trabalho Futuro

O trabalho a desenvolver no futuro deverá seguir duas directrizes. A primeira diz respeito ao incremento e refinamento das funcionalidades da plataforma desenvolvida, numa lógica de melhoria contínua. A segunda diz respeito ao desenvolvimento de aplicações externas ao sistema *ERP* Baan para situações específicas onde exista a necessidade de executar funcionalidades deste, através da plataforma desenvolvida, utilizando o modo ocasionalmente conectado.

Assim, na perspectiva de incremento das funcionalidades da plataforma desenvolvida, está planeado o desenvolvimento do grupo de funcionalidades não implementado nesta fase, mas identificado nos requisitos, nomeadamente, a geração das classes facilitadoras para linguagem C#. Para automatizar o processo de criação das classes facilitadoras deverá ser desenvolvida uma sessão Baan, pertencente ao módulo *IMS*, que a partir do código fonte do *Business Object* crie a respectiva classe facilitadora utilizando o processo e atributos já descritos na secção 5.5 desta dissertação.

Na vertente do desenvolvimento de aplicações externas o trabalho futuro de desenvolvimento passa pela concepção do protótipo da solução de mobilidade que irá utilizar as funcionalidades da plataforma desenvolvida, expostas por um *Web Service*, recorrendo ao bloco aplicacional *Disconnected Service Agent*, já apresentado na secção 3.3.3 desta dissertação. Numa primeira fase deverá ser contemplado o sector da assistência técnica pós-venda, que poderá beneficiar de uma maior rapidez na actualização dos dados no sistema *ERP*, diminuição de recursos administrativos, melhor qualidade dos dados introduzidos, planeamento mais eficiente das ordens de serviço e aproveitamento das capacidades dos dispositivos móveis actuais que englobam câmaras digitais e informação sobre o posicionamento geográfico. A criação deste protótipo de aplicação ocasionalmente conectada e consequente desenvolvimento do *Web Service* para a mobilidade deverá causar uma necessidade de avaliação do impacto na performance da plataforma. Constitui neste âmbito trabalho futuro o estudo da utilização de *Web Services* com compressão da informação, dado que normalmente são verbosos, fruto da ampla adopção do XML.

Referências e Bibliografia

Referências e Bibliografia

Adam, F.; Sammon, D., [2004] – *The Enterprise Resource Planning Decade: Lessons Learned and Issues for the Future*. Idea Group Publishing, 2004

Baan Co., [2001a] – *BaanERP 5.0c: Administrator's Guide*. Baan Development B.V., 2001

Baan Co., [2001b] – *BaanERP 5.0c Tools: Technical Manual*. Baan Development B.V., 2001

Baan Co., [2001c] – *BaanERP 5.0c Tools: User Management*. Baan Development B.V., 2001

Baan Co., [2001d] – *White Paper on Client/Server Configuration*. Baan Development B.V., 2001

Bennet, S.; McRobb S.; Farmer, R., [1999] – *Object-Oriented Systems Analysis and Design using UML*. McGraw-Hill, 1999

Boulter, M., [2004] – *Smart Client Architecture and Design Guide, Patterns & Practices*. Microsoft Corporation, 2004

Cooper, W.J., [2002] – *Introduction to Design Patterns in C#*. IBM T J Watson Research Center, 2002

Freeman, E. [et al], [2004] – *Head First Design Patterns*. O'Reilly, 2004

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J., [1995] – *Design Patterns: Elements of Reusable Object Oriented Software*. Addison-Wesley, 1995

Hossain, L.; Patrick, J.; Rashid, M.A., [2002] – *Enterprise Resource Planning: Global Opportunities & Challenges*. Idea Group Publishing, 2002

IEEE, [1998a] – *IEEE Std 829-1998, IEEE Standard for Software Test Documentation*. IEEE Computer Society, 1998

IEEE, [1998b] – *IEEE Std 830-1998*, *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Computer Society, 1998

IETF, [1991] – *A TCP/IP Tutorial*, The Internet Society, 1991, http://www.apps.ietf.org/rfc/rfc1180.html

IETF, [2000] – *Upgrading to TLS Within HTTP/1.1*, The Internet Society, 2000, http://www.apps.ietf.org/rfc/rfc2817.html

Infor, [2008] – *Differences Study Baan IVc- ERP LN 6.1 Overview*. Infor, 2008

Infor, [2009] – *Infor Enterprise Server: Technical Notes for Porting Set 8.5a.03*. Infor, 2009

Jacobson, I.; Booch, G.; Rumbaugh, J., [1999] – *The Unified Software Development Process*. Addison Wesley, 1999

Kotonya, G.; Sommerville, I., [1998] – *Requirements engineering: Processes and techniques*. Wiley, 1998

Laudon, C.; Laudon J., [2006] – *Management Information Systems: Managing the Digital Firm*. Prentice Hall, 2006

Macaulay, L.A., [1996] – *Requirements Engineering*. Springer, 1996

Microsoft, [2002] – *MSDN Training: Overview of the Microsoft .NET Framework*. Microsoft, 2002

MSFT, [1994] – *The Component Object Model: A Technical Overview*, Microsoft, 1994, http://msdn.microsoft.com/en-us/library/ms809980.aspx

MSFT, [2008] – *Smart Client Software Factory - April 2008*. Microsoft, 2008

MSFT [2009] – *Composite Smart Client Applications*, Microsoft, 2009, http://msdn.microsoft.com/en-us/library/cc540675.aspx

MSFT [2009a] – *Microsoft Patterns & Practices*, Microsoft, 2009, http://msdn.microsoft.com/en-us/practices

MSFT, [2009b] – *Microsoft .NET Framework*, Microsoft, 2009, http://www.microsoft.com/NET/

MSFT, [2009c] – *MSDN: The C# Language*, Microsoft, 2009, http://msdn.microsoft.com/en-us/vcsharp/aa336809.aspx

MSFT, [2009d] – *MSDN: Visual Studio,* Microsoft, 2009, http://msdn.microsoft.com/en-us/vstudio/default.aspx

Patton, R., [2001] – *Software Testing*. SAMS, 2001

Schach, R. S., [2004] – *Object-oriented & Classical Software Engineering 6th Edition*. McGraw-Hill, 2004

SSA Global, [2006] – *Developer's Guide for Application Function Server*. SSA Global, 2006

W3C, [1999] – World Wide Web Consortium, *Hypertext Transfer Protocol - HTTP/1.1*, The Internet Society, 1999, http://www.w3.org/Protocols/rfc2616/rfc2616.html

W3C, [2009] – World Wide Web Consortium, *Document Type Definition*, http://www.w3.org/TR/xml/#dt-doctype

W3C, [2009a] – World Wide Web Consortium, *eXtensible Markup Language*, 2009, http://www.w3.org/XML/

W3C, [2009b] – World Wide Web Consortium, *SOAP Version 1.2 Part 1: Messaging Framework*, 2009, http://www.w3.org/TR/soap12-part1/

W3C, [2009c] – World Wide Web Consortium, *W3C XML Schema Definition Language (XSD) 1.1 Part 1*, 2009, http://www.w3.org/TR/xmlschema11-1/

W3C, [2009d] – World Wide Web Consortium, *Web Services Glossary*, 2009, http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

Lista de Sítios da Internet

Adicionalmente, às referências e bibliografia foram consultados os sítios da Internet abaixo descritos e validada a efectividade dos seus endereços em Novembro de 2009.

Altova, [2009] – *XMLSpy, XML Editor*, Altova, 2009, http://www.altova.com/xmlspy.html

Baanboard, [2009] – *Baan Tools Development Forum*, Baanboard, 2009, http://www.baanboard.com/

Baanboard, [2009a] – *BaanERP Programmer's Manual*, Baanboard, 2009, http://www.baanboard.com/programmers_manual_baanerp

Infor Global Solutions, [2009] – *Infor Global Solutions*, Infor, 2009, http://www.infor.com/

Infor Global Support, [2009] – *Infor365 Online Support*, Infor, 2009, http://www.infor365.com

Microsoft, [2008] – *Smart Client Software Factory, Microsoft Patterns & Practices*, Microsoft, 2008, http://smartclient.codeplex.com/

Microsoft, [2009] – *C# Language Specification: Attributes*, Microsoft, 2009, http://msdn.microsoft.com/en-us/library/aa664611(VS.71).aspx

Microsoft, [2009a] – *C# Programming Guide: Interoperability*, Microsoft, 2009, http://msdn.microsoft.com/en-us/library/ms173184.aspx

Microsoft, [2009b] – *C# Programmer's Reference: Introduction to Attributes*, Microsoft, 2009, http://msdn.microsoft.com/en-us/library/aa288059(VS.71).aspx

Microsoft, [2009c] – *.NET Framework Developer's Guide: Advanced COM Interoperability*, Microsoft, 2009, http://msdn.microsoft.com/en-us/library/bd9cdfyx.aspx

Microsoft, [2009d] – *.NET Framework Developer's Guide: Interoperability*, Microsoft, 2009, http://msdn.microsoft.com/en-us/library/ms172270.aspx

Microsoft, [2009e] – *.NET Framework Developer's Guide: Interoperating with Unmanaged Code*, Microsoft, 2009, http://msdn.microsoft.com/en-us/library/sd10k43k.aspx

Microsoft, [2009g] – *.NET Framework Developer's Guide: .NET Framework Conceptual Overview*, Microsoft, 2009, http://msdn.microsoft.com/en-us/library/zw4w595w.aspx

Microsoft, [2009f] – *.NET Framework Developer's Guide: Reflection*, Microsoft, 2009, http://msdn.microsoft.com/en-us/library/cxz4wk15.aspx

Microsoft, [2009h] – *.NET Framework Developer's Guide: XML Documents and Data*, Microsoft, 2009, http://msdn.microsoft.com/en-us/library/2bcctyt8.aspx

Microsoft, [2009i] – *Visual Studio Programmer's Guide for the .NET Framework: .NET Framework Advanced Development*, Microsoft, 2009, http://msdn.microsoft.com/en-us/library/29eafad8.aspx

SoftArtisans, [2008] – *.NET XML Best Practices: Reading XML Documents*, SoftArtisans, 2008, http://support.softartisans.com/kbview.aspx?ID=674

Anexos

Anexo A

Código Fonte do módulo *Integration Management System - IMS*

A.1 – Código da *DLL* amimsdll00001

A.2 – Código do programa amimslaunmon

A.3 – Código do programa amimsprocreq

A.1 – Código da *DLL* amimsdll00001

```
|*****************************************************************************
|* amimsdll00001  0  VRC B50O c  expl
|* Funções do IMS
|* jaugusto
|* 03-02-09 [11:17]
|*****************************************************************************
|* Script Type: Library
|*****************************************************************************

        table   tamims000                       |Parâmetros do IMS
        table   tamims001                       |Pedidos de Integração
        table   tamims002                       |Histórico dos Pedidos de Integração

        #pragma used dll ottdllbw

        #define BINARYMODE     0

function extern long amimsdll00001.read.parameters()
{
        select amims000.*
        from amims000
        selectdo
                return(1)
        endselect

        return(0)
}

function extern long amimsdll00001.get.free.num()
{
        long ret

        ret = 0

        db.retry.point()

        select amims000.lnum
        from amims000 for update
        as set with 1 rows
        selectdo
                amims000.lnum = amims000.lnum + 1
                db.update(tamims000, DB.RETRY)
                ret = commit.transaction() = 0 ? amims000.lnum : 0
        endselect

        return(ret)
}

function extern domain tcmcs.str215m amimsdll00001.get.in.model.fname()
{
        domain tcmcs.str215m   aux.fname

        if amimsdll00001.read.parameters() then

                aux.fname = strip$(amims000.fsin)
        endif

        return(aux.fname)
}

function extern domain tcmcs.str215m amimsdll00001.get.in.fname.from.req.num(domain
tcmcs.long p.reqn)
{

        domain tcmcs.str215m   aux.fname

        if amimsdll00001.read.parameters() then

                aux.fname = strip$(amims000.fdin) & strip$(amims000.fpin) & str$(p.reqn)
& "_" & str$(get.compnr()) & "." & strip$(amims000.fein)
        endif
```

```
        return(aux.fname)
}

function extern domain tcmcs.str215m amimsdll00001.get.out.fname.from.req.num(domain
tcmcs.long p.reqn)
{

        domain tcmcs.str215m   aux.fname

        if amimsdll00001.read.parameters() then

                aux.fname = strip$(amims000.fdou) & strip$(amims000.fpou) & str$(p.reqn)
& "_" & str$(get.compnr()) & "." & strip$(amims000.feou)
        endif

        return(aux.fname)
}

function extern long amimsdll00001.create.request(domain tcmcs.str215m p.fname,
                                                  domain tcdsca p.dsca,
                                                  domain tcmcs.long p.mxpt,
                                                  domain tcmcs.long p.mxrn,
                                                  domain tcbool p.hist,
                                                  domain tcbool p.delfile)
{
        domain  tcmcs.str215m  aux.fname
        domain  tcdate         aux.date
        long                   ret, reqn.num

        ret = 0

        if p.mxpt >= 0 and p.mxrn >= 0 then

                reqn.num = amimsdll00001.get.free.num()

                if reqn.num > 0 then

                        aux.fname = amimsdll00001.get.in.fname.from.req.num(reqn.num)

                        if not isspace(aux.fname) then
                                if client2server(strip$(shiftl$(p.fname)), aux.fname,
BINARYMODE, p.delfile) = 0 then

                                        db.retry.point()

                                        aux.date = utc.num()

                                        select amims001.*
                                        from amims001 for update
                                        where amims001._index1 = {0}
                                        selectdo
                                        selectempty
                                                amims001.reqn = reqn.num
                                                amims001.dreq = aux.date
                                                amims001.logn = logname$
                                                amims001.dsca = p.dsca
                                                amims001.mxpt = p.mxpt
                                                amims001.mxrn = p.mxrn
                                                amims001.lmdt = aux.date
                                                amims001.hist = p.hist ? tcyesno.yes :
tcyesno.no

                                                db.insert(tamims001, DB.RETRY)

                                                if p.hist then

                                                        select amims002.*
                                                        from amims002 for update
                                                        where amims002._index1 = {0}
                                                        selectdo
                                                        selectempty
                                                                amims002.reqn = reqn.num
                                                                amims002.sqnb = 1
                                                                amims002.modt = aux.date
                                                                amims002.logn = logname$
```

```
                                                                db.insert(tamims002, DB.RETRY)
                                                        endselect
                                                endif

                                                ret = commit.transaction() = 0 ? reqn.num : 0
                                        endselect
                                endif
                        endif
                endif
        endif

        return(ret)
}

function extern long amimsdll00001.put.request.waiting(domain tcmcs.long p.reqn)
{
        return(amimsdll00001.update.request.status(p.reqn, amims.stat.waiting, ""))
}

function extern long amimsdll00001.execute.request(domain tcmcs.long p.reqn, domain
tcbool p.wait)
{
        long   ret, maxtime

        ret = 0

        if amimsdll00001.update.request.status(p.reqn, amims.stat.in.proc, "") then

                if amims001.mxpt = 0 then
                        if amimsdll00001.read.parameters() then
                                maxtime = amims000.mxpt
                        endif
                else
                        maxtime = amims001.mxpt
                endif

                if maxtime > 0 then

                        if p.wait then
                                if wait.and.activate("oamimslaunmon", STR$(p.reqn),
STR$(maxtime), STR$(amims001.rnum), STR$(amims001.mxrn)) > 0 then
                                        ret = 1
                                endif
                        else
                                if activate("oamimslaunmon", STR$(p.reqn), STR$(maxtime),
STR$(amims001.rnum), STR$(amims001.mxrn)) > 0 then
                                        ret = 1
                                endif
                        endif
                endif

        endif

        return(ret)
}

function extern long amimsdll00001.get.request.stat(domain tcmcs.long p.reqn, ref domain
amims.stat p.stat)
{
        long   ret

        ret = 0

        select amims001.stat:p.stat
        from amims001
        where amims001._index1 = {:p.reqn}
        selectdo
                ret = 1
        endselect

        return(ret)
}

function extern long amimsdll00001.get.request.stat.const(domain tcmcs.long p.reqn)
{
```

```
        domain  amims.stat     aux.stat
        long                   ret

        ret = 0

        if amimsdll00001.get.request.stat(p.reqn, aux.stat) then
                ret = etol(aux.stat)
        endif

        return(ret)
}

function extern long amimsdll00001.get.request.proc.result(domain tcmcs.long p.reqn,
domain tcmcs.str215m p.fname)
{
        domain tcmcs.str215m   result.file
        domain  amims.stat     aux.stat
        long                   ret

        ret = 0

        if amimsdll00001.get.request.stat(p.reqn, aux.stat) then

                if aux.stat = amims.stat.processed or aux.stat = amims.stat.proc.errors
then

                        result.file = amimsdll00001.get.out.fname.from.req.num(p.reqn)

                        if server2client(result.file, p.fname, BINARYMODE) = 0 then
                                ret = 1
                        endif
                endif
        endif

        return(ret)
}

function extern void amimsdll00001.create.and.put.waiting.request(domain tcmcs.str215m
p.fname,
                                                                  domain tcdsca p.dsca,
                                                                 domain tcmcs.long p.mxpt,
                                                                 domain tcmcs.long p.mxrn,
                                                                 domain tcbool p.hist,
                                                                 domain tcbool p.delfile,
                                                                 ref long p.reqn.res, ref
long p.wait.res)
{
        p.reqn.res = 0
        p.wait.res = 0

        p.reqn.res = amimsdll00001.create.request(p.fname, p.dsca, p.mxpt, p.mxrn,
p.hist, p.delfile)

        if p.reqn.res > 0 then
                p.wait.res = amimsdll00001.put.request.waiting(p.reqn.res)
        endif
}

function extern void amimsdll00001.create.and.execute.request(domain tcmcs.str215m
p.fname,
                                                              domain tcdsca p.dsca,
                                                              domain tcmcs.long p.mxpt,
                                                              domain tcmcs.long p.mxrn,
                                                              domain tcbool p.hist,
                                                              domain tcbool p.delfile,
                                                              ref long p.reqn.res, ref long
p.exec.res)
{
        p.reqn.res = 0
        p.exec.res = 0

        p.reqn.res = amimsdll00001.create.request(p.fname, p.dsca, p.mxpt, p.mxrn,
p.hist, p.delfile)

        if p.reqn.res > 0 then
                p.exec.res = amimsdll00001.execute.request(p.reqn.res, false)
```

```
        endif
}

function extern void amimsdll00001.create.exec.and.get.result(domain tcmcs.str215m
p.ifname,
                                                              domain tcdsca p.dsca,
                                                              domain tcmcs.long p.mxpt,
                                                              domain tcmcs.long p.mxrn,
                                                              domain tcbool p.hist,
                                                              domain tcbool p.delfile,
                                                              domain tcmcs.str215m p.ofname,
                                                              ref long p.reqn.res, ref long
p.exec.res, ref long p.resu.res)
{
        p.reqn.res = 0
        p.exec.res = 0
        p.resu.res = 0

        p.reqn.res = amimsdll00001.create.request(p.ifname, p.dsca, p.mxpt, p.mxrn,
p.hist, p.delfile)

        if p.reqn.res > 0 then

                p.exec.res = amimsdll00001.execute.request(p.reqn.res, true)

                if p.exec.res then
                        p.resu.res = amimsdll00001.get.request.proc.result(p.reqn.res,
p.ofname)
                endif

        endif
}

function extern long amimsdll00001.update.request.status(domain tcmcs.long p.reqn,
                                                         domain amims.stat p.stat,
                                                         domain tcmcs.str215m p.lmes)
{
        domain  tcdate          aux.date
        long                    ret

        ret = 0

        db.retry.point()

        select amims001.*
        from amims001 for update
        where amims001._index1 = {:p.reqn}
        selectdo
                if amimsdll00001.check.stat.trans(amims001.stat, p.stat) then

                        if p.stat = amims.stat.in.proc then
                                amims001.rnum = amims001.rnum + 1
                        endif

                        if p.stat <> amims.stat.in.proc or
                           ((p.stat = amims.stat.in.proc and amims001.rnum <=
amims001.mxrn)
                           or amims001.mxrn = 0) then

                                aux.date = utc.num()

                                if amims001.hist = tcyesno.yes then

                                        select amims002.*
                                        from amims002 for update
                                        where amims002._index1 = {:p.reqn}
                                        order by amims002._index1 desc
                                        as set with 1 rows
                                        selectdo
                                                amims002.sqnb = amims002.sqnb + 1
                                                amims002.modt = aux.date
                                                amims002.logn = logname$
                                                amims002.stat = p.stat
                                                amims002.lmes = p.lmes

                                                db.insert(tamims002, DB.RETRY)
```

```
                                        endselect
                                endif

                                amims001.stat = p.stat
                                amims001.lmdt = aux.date
                                amims001.lmes = p.lmes

                                db.update(tamims001, DB.RETRY)

                                ret = commit.transaction() = 0 ? 1 : 0
                        endif
                endif
        endselect

        return(ret)
}

function extern long amimsdll00001.check.stat.trans(domain amims.stat old.stat, domain
amims.stat new.stat)
{
        long   ret

        ret = 0

        if old.stat = amims.stat.created and new.stat = amims.stat.in.proc then
                ret = 1
        endif
        if old.stat = amims.stat.created and new.stat = amims.stat.waiting then
                ret = 1
        endif
        if old.stat = amims.stat.waiting and new.stat = amims.stat.in.proc then
                ret = 1
        endif
        if old.stat = amims.stat.in.proc and new.stat = amims.stat.waiting then
                ret = 1
        endif
        if old.stat = amims.stat.in.proc and new.stat > amims.stat.in.proc then
                ret = 1
        endif

        return(ret)
}
```

A.2 – Código do programa amimslaunmon

```
|*****************************************************************************
|* amimslaunmon  0  VRC B50O c  expl
|* Controlo de pedidos de integração
|* jaugusto
|* 20-02-09 [14:50]
|*****************************************************************************
|* Script Type: 0
|*****************************************************************************

        #pragma used dll oamimsdll00001

function main()
{
        long            arg.reqn, arg.mxpt, arg.rnum, arg.mxrn
        long            ret, nsec, maxtime, prid, prgr, lead.pid
        long            pinfo(PSMAXSIZE)
        string          progname(512)

        arg.reqn = lval(argv$(1))
        arg.mxpt = lval(argv$(2))
        arg.rnum = lval(argv$(3))
        arg.mxrn = lval(argv$(4))

        ret = 0
        nsec = 0
        maxtime = arg.mxpt

        prid = act.and.sleep("oamimsprocreq", str$(arg.reqn))

        if prid > 0 then

                prgr = set.pgrp(prid, prid)

                lead.pid = prid

                reactivate(prid)

                ret = 1

                while get.pgrp(lead.pid) > 0

                        suspend(1000)

                        nsec = nsec + 1

                        if nsec > maxtime then

                                kill.pgrp(get.pgrp(prgr))

                                while prid > 0

                                        prid = pstat(prid, progname, pinfo)

                                        if ps.parent(pinfo) = lead.pid then

                                                kill.pgrp(get.pgrp(prid))

                                        endif

                                endwhile

                                ret = 2
                                break
                        endif

                endwhile
        endif

        if ret > 0 then
                if ret = 2 then
                        if arg.rnum = arg.mxrn then
```

```
                                amimsdll00001.update.request.status(arg.reqn,
                                amims.stat.canceled, "")
                        else
                                amimsdll00001.update.request.status(arg.reqn,
                                amims.stat.waiting, "")
                        endif
                endif
        else
                amimsdll00001.update.request.status(arg.reqn, amims.stat.proc.errors,
                form.text$("amims00001", language$))
        endif
}
```

A.3 – Código do programa amimsprocreq

```
|*****************************************************************************
|* amimsprocreq  0  VRC B50O c  expl
|* Processamento de pedidos de Integração
|* jaugusto
|* 10-02-09 [16:54]
|*****************************************************************************
|* Script Type: 0
|*****************************************************************************

        #pragma used dll oamimsdll00001
        #pragma used dll oamimsxmlhand

        #define PTYPE_BASIC (ptype = "string" or ptype = "long" or ptype = "double" or
ptype = "date")
        #define PTYPE_XMLST (ptype = "xmlst")

        long                    arg.reqn
        long                    ret.xml
        string                  lmess(10)

function main()
{
        long                    ret, xml.tree, np, inp, onp, ap, ap1
        domain tcmcs.str15      bprogram
        domain tcmcs.str10      otype
        domain tcmcs.str215     funcname

        ret = 0

        arg.reqn = lval(argv$(1))

        if open.request.xml(arg.reqn, xml.tree) then

                np = xmlFindFirst("RTDDOBJECT", xml.tree)

                if np <> 0 then

                        ap = xmlGetAttribute(np, "otype", otype)
                        ap1 = xmlGetAttribute(np, "oname", bprogram)

                        if ap <> 0 and ap1 <> 0 then

                                np = xmlFindFirst("Function", np)

                                if np > 0 then

                                        ap = xmlGetAttribute(np, "fname", funcname)

                                        if ap <> 0 then

                                                if
amimsxmlhand.create.result.xml.structure(otype, bprogram, funcname, ret.xml) then

                                                        on case otype
                                                                case "ImsBO":
                                                                        inp =
xmlFindFirst("InParams", np)
                                                                        onp =
xmlFindFirst("OutParams", np)
                                                                        ret =
exec.ImsBO.request(bprogram, funcname, inp, onp)
                                                                        break
                                                                case "Ims3GL":
                                                                        inp =
xmlFindFirst("InParams", np)
                                                                        ret =
exec.Baan3GL.request(bprogram, inp)
                                                                        break
                                                                default:
                                                                        lmess = "amims00004"
                                                                        break
```

```
                                                        endcase

                                                else
                                                        lmess ="amims00002"
                                                endif

                                        else
                                                lmess = "amims00004"
                                        endif
                                else
                                        lmess = "amims00004"
                                endif
                        else
                                lmess = "amims00004"
                        endif
                else
                        lmess ="amims00003"
        endif

        if ret then
                if write.result.xml(arg.reqn, ret.xml) then
                        amimsdll00001.update.request.status(arg.reqn,
amims.stat.processed, form.text$(lmess, language$))
                else
                        amimsdll00001.update.request.status(arg.reqn,
amims.stat.proc.errors,  form.text$("amims00007", language$))
                endif
        else
                amimsdll00001.update.request.status(arg.reqn, amims.stat.proc.errors,
form.text$(lmess, language$))
        endif
}

function long exec.ImsBO.request(domain tcmcs.str15 p.bprogram, domain tcmcs.str215
p.funcname, const long p.inp, const long p.onp)
{
        long                    ret, np, np.out, ap1, ap2, error.sec
        long                    param.ok, has.out.param, outsect, dllid, funcid
        string                  imsvar(500)
        string                  auxdata(1024)
        string                  ptype(6)

        ret = 0

        dllid = load_dll(strip$(shiftl$(p.bprogram)))
        funcid = get_function(dllid, p.funcname)

        if funcid > 0 then

                param.ok = true
                has.out.param = false

                np = xmlGetFirstChild(p.inp)

                while np <> 0

                        ap1 = xmlGetAttribute(np, "pname", imsvar)
                        ap2 = xmlGetAttribute(np, "ptype", ptype)

                        if ap1 <> 0 and ap2 <> 0 then

                                if PTYPE_BASIC then
                                        if xmlGetData(np, auxdata) > 0 then
                                                if not put.var(pid, imsvar,
                                                strip$(shiftl$(auxdata)) ) then
                                                        param.ok = false
                                                        break
                                                endif
                                        endif
                                else
                                        if PTYPE_XMLST then
                                                if not put.var(pid, imsvar, np) then
                                                        param.ok = false
```

```
                                        break
                                endif
                        else
                                param.ok = false
                                break
                        endif
                endif
        else
                if xmlGetType(np) = XML_ELEMENT then
                        param.ok = false
                        break
                endif
        endif

        np = xmlGetRightSibling(np)
endwhile

error.sec = xmlFindFirst("Errors", ret.xml)

if error.sec <> 0 then
        if not put.var(pid, "ims.Errors", error.sec) then
                param.ok = false
        endif
endif

np = xmlGetFirstChild(p.onp)

while np <> 0 and param.ok

        ap1 = xmlGetAttribute(np, "pname", imsvar)
        ap2 = xmlGetAttribute(np, "ptype", ptype)

        if ap1 <> 0 and ap2 <> 0 then

                has.out.param = true

                if PTYPE_XMLST then
                        if not put.var(pid, imsvar, np) then
                                param.ok = false
                                break
                        endif
                else
                        if PTYPE_BASIC then
                                if not put.var(pid, imsvar, "") then
                                        param.ok = false
                                        break
                                endif
                        else
                                param.ok = false
                                break
                        endif
                endif
        else
                if xmlGetType(np) = XML_ELEMENT then
                        param.ok = false
                        break
                endif
        endif

        np = xmlGetRightSibling(np)
endwhile

if param.ok then

        exec_function(dllid, funcid)

        ret = 1

        if has.out.param then

                outsect = xmlFindFirst("OutParams", ret.xml)

                if outsect <> 0 then

                        np = xmlGetFirstChild(p.onp)
```

```
                                        while np <> 0

                                                ap1 = xmlGetAttribute(np, "pname", imsvar)

                                                ap2 = xmlGetAttribute(np, "ptype", ptype)

                                                if ap1 <> 0 and ap2 <> 0 then

                                                        if PTYPE_BASIC then

                                                                np.out =
xmlDuplicateAndAppendToChilds(outsect, np)

                                                                if np.out <> 0 then

                                                                        if get.var(pid,
imsvar, auxdata) then

                                                                                if
xmlNewNode(strip$(shiftl$(auxdata)), XML_DATA, np.out) = 0 then
                                                                                        lmess
= "amims00015"
                                                                                endif
                                                                        else
                                                                                lmess =
"amims00015"
                                                                        endif
                                                                else
                                                                        lmess = "amims00015"
                                                                endif
                                                        else
                                                                if get.var(pid, imsvar,
np.out) then

                                                                        if
xmlDuplicateAndAppendToChilds(outsect, np.out) = 0 then
                                                                                lmess =
"amims00015"
                                                                        endif
                                                                else
                                                                        lmess = "amims00015"
                                                                endif
                                                        endif
                                                endif

                                                np = xmlGetRightSibling(np)
                                        endwhile
                                endif
                        endif

                        error.sec = xmlFindFirst("Errors", ret.xml)

                        if error.sec <> 0 then
                                if get.var(pid, "ims.Errors", np.out) then
                                        error.sec = np.out
                                else
                                        lmess = "amims00016"
                                endif
                        endif
                else
                        lmess ="amims00006"
                endif
        else
                lmess ="amims00005"
        endif

        return(ret)
}

function long exec.Baan3GL.request(domain tcmcs.str15 p.bprogram, const long p.inp)
{
        long                    ret, prid, np, ap1, ap2, i
        long                    param.ok
        string                  imsvar(500)
        string                  auxdata(1024)
        string                  strparam(1,1) BASED
```

```
        string                  ptype(6)

        ret = 0
        i = 0
        param.ok = true

        np = xmlGetFirstChild(p.inp)

        while np > 0

                ap1 = xmlGetAttribute(np, "pname", imsvar)
                ap2 = xmlGetAttribute(np, "ptype", ptype)

                if ap1 <> 0 and ap2 <> 0 and PTYPE_BASIC then

                        i = i + 1

                        if alloc.mem(strparam, 1024, i) = 0 then
                                if xmlGetData(np, auxdata) > 0 then
                                        strparam(1, i) = strip$(shiftl$(auxdata))
                                else
                                        strparam(1, i) = " "
                                endif
                        else
                                param.ok = false
                                break
                        endif

                else
                        if xmlGetType(np) = XML_ELEMENT then
                                param.ok = false
                                break
                        endif
                endif

                np = xmlGetRightSibling(np)
        endwhile

        if param.ok then

                prid = wait.and.activate(p.bprogram, strparam)

                if prid > 0 then
                        ret = 1
                else
                        lmess ="amims00005"
                endif
        else
                lmess ="amims00006"
        endif

        free.mem(strparam)

        return(ret)
}

function long open.request.xml(domain tcmcs.long p.reqn, ref long p.xmltree)
{
        domain  tcmcs.str215m  fname
        long                   ret, fp
        string                 error.str(120)

        ret = 0

        fname = amimsdll00001.get.in.fname.from.req.num(p.reqn)

        fp = seq.open(fname, "rt")

        if fp > 0 then

                p.xmltree = xmlRead(fp, error.str, XmlPreserveWhiteSpace)

                seq.close(fp)

                if p.xmltree <> 0 then
                        ret = 1
```

```
                endif
        endif

        return(ret)
}

function long write.result.xml(domain tcmcs.long p.reqn, const long p.xmltree)
{
        domain  tcmcs.str215m  fname
        long                   ret, fp

        ret = 0

        fname = amimsdll00001.get.out.fname.from.req.num(p.reqn)

        fp = seq.open(fname, "wt")

        if fp > 0 then

                ret = xmlWritePretty(fp, p.xmltree)

                seq.close(fp)

                if ret > 0 then
                        ret = 1
                endif
        endif

        return(ret)
}
```

Anexo B

Código Fonte do subsistema Camada de Acesso ao *IMS*

B.1 – Código do susbsistema Controlo e Configuração do Acesso

B.2 – Código do susbsistema Fachada do *IMS*

B.1 – Código do susbsistema Controlo e Configuração do Acesso

Interface ***IBaanEndpointCatFactory*** e classe ***BaanEndpointCatFactory***

```
namespace Baan.WCA
{
    public interface IBaanEndpointCatFactory
    {
        IBaanEndpointCat CreateCatalog();
    }
}

namespace Baan.WCA
{
    public class BaanEndpointCatFactory : IBaanEndpointCatFactory
    {
        BwConfigManager bwcman;

        public BaanEndpointCatFactory(BwConfigManager bwcmanager)
               {

            bwcman = bwcmanager;

               }

        public IBaanEndpointCat CreateCatalog()
        {
            BaanEndpointCat bec = new BaanEndpointCat();

            BaanEndpointSection endpointsection;

            try
            {
                endpointsection =
ConfigurationManager.GetSection("BaanSettings/BaanEndpoints") as BaanEndpointSection;
            }
            catch
            {
                return bec;
            }

            List<BaanEndpointElement> libe =
endpointsection.BaanEndpointItemsCollection.GetEndpoints();

            foreach(BaanEndpointElement bee in libe)
            {
                BaanEndpoint be = new BaanEndpoint(bee.BwcFileName, bee.ServerName,
bee.ServerBSE, bee.Protocol, bee.PortNumber, bee.Company, bee.BaanUser, bee.UserName,
bee.Password, bee.BShell, bee.BwClassName);

                if (bwcman.CreateBwcForEndpoint(be))
                    bec.AddEndpoint(be);
            }

            return bec;
        }
    }
}
```

Interface ***IBaanEndpointCat*** e classe ***BaanEndpointCat***

```
namespace Baan.WCA
{
    public interface IBaanEndpointCat
    {
        int Count { get; }

        Dictionary<string, BaanEndpoint> BaanEndpoints { get; }

        bool EndpointExists(string endpointname);

        bool EndpointExists(string servername, string company, string username);

        BaanEndpoint GetEndpoint(string endpointname);

        BaanEndpoint GetEndpoint(string servername, string company, string username);

        void AddEndpoint(BaanEndpoint endpoint);

        void RemoveEndpoint(string endpointname);
    }
}

namespace Baan.WCA
{
    public class BaanEndpointCat : IBaanEndpointCat
    {
        private Dictionary<string, BaanEndpoint> endpoints = new Dictionary<string,
BaanEndpoint>();

        public int Count
        {
            get { return endpoints.Count; }
        }

        public Dictionary<string, BaanEndpoint> BaanEndpoints
        {
            get { return endpoints; }
        }

        public bool EndpointExists(string endpointname)
        {
            if (! String.IsNullOrEmpty(endpointname))
                return endpoints.ContainsKey(endpointname);
            else
                return false;
        }

        public bool EndpointExists(string servername, string company, string username)
        {
            if (String.IsNullOrEmpty(servername) || String.IsNullOrEmpty(company) ||
String.IsNullOrEmpty(username))
                return false;
            else
            {
                foreach (BaanEndpoint be in endpoints.Values)
                {
                    if (be.ServerName == servername && be.Company == company &&
be.UserName == username)
                        return true;
                }

                return false;
            }
        }

        public BaanEndpoint GetEndpoint(string endpointname)
        {
            if (! String.IsNullOrEmpty(endpointname))
                return endpoints[endpointname];
            else
                return null;
        }
```

```
        public BaanEndpoint GetEndpoint(string servername, string company, string
username)
        {
            if (String.IsNullOrEmpty(servername) || String.IsNullOrEmpty(company) ||
String.IsNullOrEmpty(username))
                return null;
            else
            {
                foreach (BaanEndpoint be in endpoints.Values)
                {
                    if (be.ServerName == servername && be.Company == company &&
be.UserName == username)
                        return be;
                }

                return null;
            }
        }

        public void AddEndpoint(BaanEndpoint endpoint)
        {
            if (endpoints.ContainsKey(endpoint.BwcFileName))
                endpoints.Remove(endpoint.BwcFileName);

            endpoints.Add(endpoint.BwcFileName, endpoint);
        }

        public void RemoveEndpoint(string endpointname)
        {
            if (endpoints.ContainsKey(endpointname))
                endpoints.Remove(endpointname);
        }
    }
}
```

Classe ***BaanEndpoint***

```
namespace Baan.WCA
{
    public class BaanEndpoint
    {
        string bwcfilename;
        string servername;
        string serverbse;
        string protocol;
        ushort portnumber;
        string company;
        string baanuser;
        string username;
        string password;
        string bshell;
        string bwclassname;

        public BaanEndpoint()
        {
        }

        public BaanEndpoint(string pbwcfilename,
                            string pserver,
                            string pserverbse,
                            string pprotocol,
                            ushort pportnumber,
                            string pcompany,
                            string pbaanuser,
                            string pusername,
                            string ppassword,
                            string pbshell,
```

```
                                string pbwclassname)
        {
            bwcfilename = pbwcfilename;
            servername = pserver;
            serverbse = pserverbse;
            protocol = pprotocol;
            portnumber = pportnumber;
            company = pcompany;
            baanuser = pbaanuser;
            username = pusername;
            password = ppassword;
            bshell = pbshell;
            bwclassname = pbwclassname;
        }

        public string BwcFileName {
            get { return bwcfilename; }
            set { bwcfilename = value; }
        }

        public string ServerName {
            get { return servername; }
            set { servername = value; }
        }

        public string ServerBSE {
            get { return serverbse; }
            set { serverbse = value; }
        }

        public string Protocol {
            get { return protocol; }
            set { protocol = value; }
        }

        public ushort PortNumber {
            get { return portnumber; }
            set { portnumber = value; }
        }

        public string Company {
            get { return company; }
            set { company = value; }
        }

        public string BaanUser {
            get { return baanuser; }
            set { baanuser = value; }
        }

        public string UserName {
            get { return username; }
            set { username = value; }
        }

        public string Password {
            get { return password; }
            set { password = value; }
        }

        public string BShell {
            get { return bshell; }
            set { bshell = value; }
        }

        public string BwClassName {
            get { return bwclassname; }
            set { bwclassname = value; }
        }
    }
}
```

Classe ***BwConfigManager***

```
namespace Baan.WCA
{
    public class BwConfigManager
    {
        [NonSerialized]
        private Dictionary<string, BWConfigurationClass> bwconfigs = new
Dictionary<string, BWConfigurationClass>();
        string bwcdir;

        public BwConfigManager(string bwcfiledir)
        {
            bwcdir = bwcfiledir;

            DirectoryInfo di = new DirectoryInfo(bwcdir);
            FileInfo[] bwcfiles = di.GetFiles("*.bwc");

            foreach (FileInfo fi in bwcfiles)
            {
                BWConfigurationClass bc = new BWConfigurationClass();

                if(bc.Load(fi.FullName))
                    bwconfigs.Add(fi.Name, bc);

                bc = null;
            }
        }

        public string BwcFileDirectory
        {
            get {return bwcdir;}
        }

        public Dictionary<string, BWConfigurationClass> BwConfigurations
        {
            get { return bwconfigs; }
        }

        public int Count
        {
            get { return bwconfigs.Count; }
        }

        public bool CreateBwcForEndpoint(BaanEndpoint be)
        {
            BWConfigurationClass bc = new BWConfigurationClass();
            bool ret;

            bc.Hostname = be.ServerName;
            bc.ServerBSE = be.ServerBSE;
            bc.Protocol = be.Protocol;

            if (be.PortNumber > 0)
                bc.Portnumber = be.PortNumber;

            bc.LoginDialog = "NO";
            bc.Username = be.UserName;
            bc.Password = be.Password;

            if (be.BShell.Length > 0)
                bc.BShell = be.BShell;

            if (be.BaanUser != be.UserName)
                bc.Command = "-- -set USER=" + be.BaanUser + " -set BSE_COMPNR=" +
be.Company;
            else
                bc.Command = "-- -set BSE_COMPNR=" + be.Company;

            try
            {
                bc.Save(bwcdir + be.BwcFileName);
                be.BwClassName = bc.ClassName;
                ret = true;
            }
```

```
            catch
            {
                ret = false;
            }

            if (ret)
            {
                if (bwconfigs.ContainsKey(be.BwcFileName))
                    bwconfigs.Remove(be.BwcFileName);

                bwconfigs.Add(be.BwcFileName, bc);
            }

            return ret;
        }

        public bool BwcExists(string BwcFileName)
        {
            if (!String.IsNullOrEmpty(BwcFileName))
                return bwconfigs.ContainsKey(BwcFileName);
            else
                return false;
        }

        public BWConfigurationClass GetBwConfig(string BwcFileName)
        {
            if (!String.IsNullOrEmpty(BwcFileName))
            {
                return bwconfigs[BwcFileName];
            }
            else
                return null;
        }

        public bool DeleteBwc(string BwcFileName)
        {
            bool ret;

            try
            {
                ret = bwconfigs[BwcFileName].Delete(bwcdir + BwcFileName);
            }
            catch
            {
                ret = false;
            }

            if (ret)
                bwconfigs.Remove(BwcFileName);

            return ret;
        }
    }
}
```

B.2 – Código do susbsistema Fachada do *IMS*

Classe *IMSCallDetails*

```
namespace Baan.IMS
{
    [XmlRoot("BaanIMSCall")]
    public class IMSCallDetails
    {
        string _descr;
        int    _mxtime;
        int    _mxrun;
        bool   _history;
        bool   _delfile;

        public IMSCallDetails()
        {}

        public IMSCallDetails(string descr, int mxtime, int mxrun, bool history, bool
delfile)
        {
            _descr = descr;
            _mxtime = mxtime;
            _mxrun = mxrun;
            _history = history;
            _delfile = delfile;
        }

        [XmlAttribute("descr")]
        public string Description
        {
            get { return _descr; }
            set { _descr = value; }
        }

        [XmlAttribute("mxtime")]
        public int MaxTime
        {
            get { return _mxtime; }
            set { _mxtime = value; }
        }

        [XmlAttribute("mxrun")]
        public int MaxRun
        {
            get { return _mxrun; }
            set { _mxrun = value; }
        }

        [XmlAttribute("history")]
        public bool History
        {
            get { return _history; }
            set { _history = value; }
        }

        [XmlAttribute("delfile")]
        public bool DelLocFile
        {
            get { return _delfile; }
            set { _delfile = value; }
        }
    }
}
```

Classe ***BaanIMSBO***

```
namespace Baan.IMS
{
    public class BaanIMSBO
    {
        List<string> _errors = new List<string>();

        public List<string> Errors
        {
            get { return _errors; }
            set { _errors = value; }
        }
    }
}
```

Classe ***BaanIMS***

```
namespace Baan.IMS
{
    public enum ReqStatus : byte { NoInfo, Created, Waiting, InExecution, Executed,
WithErrors, Canceled };

    public sealed class BaanIMS
    {
        BWA ba;
        string tempdir;
        BaanEndpoint endpointinfo;

        public BaanIMS(BaanEndpoint bep, string tempfiledir)
        {
            endpointinfo = bep;
            tempdir = tempfiledir;
            ba = new BWA(bep);
        }

        public BaanEndpoint EndpointInfo
        {
            get { return endpointinfo; }
        }

        public int Create(BaanIMSBO imsbo, IMSCallDetails imscall)
        {
            int ret = 0;
            string locfname;

            locfname = GetTmpFileName();

            if (SerializeCall(imsbo, imscall, locfname))
            {
            ba.ParseExecFunction("oamimsdll00001", "amimsdll00001.create.request", new
object[] { '"' + locfname + '"',

'"' + imscall.Description + '"',

imscall.MaxTime,

imscall.MaxRun,

imscall.History ? 1 : 0,

imscall.DelLocFile ? 1 : 0});

            ret = Convert.ToInt32(ba.ReturnValue);
            }
            return (ret);
        }

        public bool PutWaiting(int preqn)
```

```
        {
            bool ret;

            ba.ParseExecFunction("oamimsdll00001", "amimsdll00001.put.request.waiting",
new object[] { preqn });

            ret = ba.ReturnValue == "1" ? true : false;

            return (ret);
        }

        public bool Execute(int preqn, bool pwait)
        {
            bool ret;

            ba.ParseExecFunction("oamimsdll00001", "amimsdll00001.execute.request", new
object[] { preqn, pwait ? 1 : 0 });

            ret = ba.ReturnValue == "1" ? true : false;

            return (ret);
        }

        public ReqStatus GetStatus(int preqn)
        {
            ReqStatus ret;
            byte statconst;

            ba.ParseExecFunction("oamimsdll00001",
"amimsdll00001.get.request.stat.const", new object[] { preqn });

            statconst = Convert.ToByte(ba.ReturnValue);

            ret = (ReqStatus)statconst;

            return (ret);
        }

        public bool GetResult(int preqn, BaanIMSBO imsbo)
        {
            string locfname;
            bool ret;

            locfname = GetTmpFileName();

            ba.ParseExecFunction("oamimsdll00001",
"amimsdll00001.get.request.proc.result", new object[] { preqn, '"' + locfname + '"' });

            ret = ba.ReturnValue == "1" ? true : false;

            if (ret)
                ret = DeserializeResult(imsbo, locfname);

            return (ret);
        }

        public void CreateAndPutWaiting(BaanIMSBO imsbo, IMSCallDetails imscall, out int
preqn, out bool pwaitret)
        {
            string locfname;
            string[] retcall;

            locfname = GetTmpFileName();

            preqn = 0;
            pwaitret = false;

            if (SerializeCall(imsbo, imscall, locfname))
            {

            ba.ParseExecFunction("oamimsdll00001",
"amimsdll00001.create.and.put.waiting.request", new object[] { '"' + locfname + '"',

'"' + imscall.Description + '"',

imscall.MaxTime,
```

```
imscall.MaxRun,

imscall.History ? 1 : 0,

imscall.DelLocFile ? 1 : 0,

0,

0});
            retcall = ba.ReturnCall.Split(',');

            preqn = Convert.ToInt32(retcall[retcall.Length - 2]);
            pwaitret = retcall[retcall.Length - 1].Substring(0, 1) == "1" ? true :
false;
            }
        }

        public void CreateAndExecute(BaanIMSBO imsbo, IMSCallDetails imscall, out int
preqn, out bool pexecret)
        {
            string locfname;
            string[] retcall;

            locfname = GetTmpFileName();

            preqn = 0;
            pexecret = false;

            if (SerializeCall(imsbo, imscall, locfname))
            {

            ba.ParseExecFunction("oamimsdll00001",
"amimsdll00001.create.and.execute.request", new object[] { '"' + locfname + '"',

'"' + imscall.Description + '"',

imscall.MaxTime,

imscall.MaxRun,

imscall.History ? 1 : 0,

imscall.DelLocFile ? 1 : 0,

0,

0});
            retcall = ba.ReturnCall.Split(',');

            preqn = Convert.ToInt32(retcall[retcall.Length-2]);
            pexecret = retcall[retcall.Length - 1].Substring(0, 1) == "1" ? true :
false;
            }
        }

        public void CreateExecuteAndGetResult(BaanIMSBO imsbo, IMSCallDetails imscall,
out int preqn, out bool pexecret, out bool pgresret)
        {
            string infname, oufname;
            string[] retcall;

            infname = GetTmpFileName();
            oufname = "";

            preqn = 0;
            pexecret = false;
            pgresret = false;

            if (SerializeCall(imsbo, imscall, infname))
            {
                oufname = GetTmpFileName();

                ba.ParseExecFunction("oamimsdll00001",
"amimsdll00001.create.exec.and.get.result", new object[] { '"' + infname + '"',
```

```
'"' + imscall.Description + '"',

imscall.MaxTime,

imscall.MaxRun,

imscall.History ? 1 : 0,

imscall.DelLocFile ? 1 : 0,

'"' + oufname + '"',

0,

0,

0});
                retcall = ba.ReturnCall.Split(',');

                preqn = Convert.ToInt32(retcall[retcall.Length - 3]);
                pexecret = retcall[retcall.Length - 2].Substring(0, 1) == "1" ? true :
false;
                pgresret = retcall[retcall.Length - 1].Substring(0, 1) == "1" ? true :
false;
            }

            if (pgresret)
                pgresret = DeserializeResult(imsbo, oufname);

        }

        private bool SerializeCall(BaanIMSBO imsbo, IMSCallDetails imscall, string
fname)
        {
            bool ret;

            XmlWriterSettings wrset = new XmlWriterSettings();
            wrset.Indent = true;
            wrset.CloseOutput = true;
            wrset.Encoding = Encoding.Unicode;

            try
            {
                XmlWriter writer = XmlWriter.Create(fname, wrset);

                object[] attr =
imscall.GetType().GetCustomAttributes(typeof(XmlRootAttribute), false);

                foreach (XmlRootAttribute o in attr)
                {
                    writer.WriteStartElement(o.ElementName);
                }

                PropertyInfo[] pi = imscall.GetType().GetProperties();

                foreach (PropertyInfo prop in pi)
                {
                    object[] attr0 =
prop.GetCustomAttributes(typeof(XmlAttributeAttribute), false);

                    foreach (XmlAttributeAttribute o in attr0)
                    {
                        writer.WriteAttributeString(o.AttributeName,
prop.GetValue(imscall, null).ToString());
                    }
                }

                object[] attr1 =
imsbo.GetType().GetCustomAttributes(typeof(BaanIMSRTDDOBJECTAtribute), false);

                foreach (BaanIMSRTDDOBJECTAtribute rtdd in attr1)
                {
                    object[] attr2 =
rtdd.GetType().GetCustomAttributes(typeof(XmlRootAttribute), false);
```

```
                    foreach (XmlRootAttribute o in attr2)
                    {
                        writer.WriteStartElement(o.ElementName);
                    }

                    PropertyInfo[] pi1 = rtdd.GetType().GetProperties();

                    foreach (PropertyInfo prop in pi1)
                    {
                        object[] attr3 =
prop.GetCustomAttributes(typeof(XmlAttributeAttribute), false);

                        foreach (XmlAttributeAttribute o in attr3)
                        {
                            writer.WriteAttributeString(o.AttributeName,
prop.GetValue(rtdd, null).ToString());
                        }
                    }
                }

                MethodInfo[] meth1 = imsbo.GetType().GetMethods();

                foreach (MethodInfo meth in meth1)
                {
                    object[] attr4 =
meth.GetCustomAttributes(typeof(BaanIMSFunctionAttribute), false);

                    foreach (BaanIMSFunctionAttribute fun in attr4)
                    {
                        object[] attr5 =
fun.GetType().GetCustomAttributes(typeof(XmlRootAttribute), false);

                        foreach (XmlRootAttribute o in attr5)
                        {
                            writer.WriteStartElement(o.ElementName);
                        }

                        PropertyInfo[] pi2 = fun.GetType().GetProperties();

                        foreach (PropertyInfo prop in pi2)
                        {
                            object[] attr6 =
prop.GetCustomAttributes(typeof(XmlAttributeAttribute), false);

                            foreach (XmlAttributeAttribute o in attr6)
                            {
                                writer.WriteAttributeString(o.AttributeName,
prop.GetValue(fun, null).ToString());
                            }
                        }
                    }
                }

                PropertyInfo[] pi3 = imsbo.GetType().GetProperties();

                List<PropertyInfo> inparams = new List<PropertyInfo>();
                List<PropertyInfo> outparams = new List<PropertyInfo>();

                foreach (PropertyInfo parprop in pi3)
                {
                    object[] attr61 =
parprop.GetCustomAttributes(typeof(BaanIMSParameterAttribute), false);

                    foreach (BaanIMSParameterAttribute par in attr61)
                    {
                        if (par.ParamDirection == "in")
                            inparams.Add(parprop);
                        else
                            outparams.Add(parprop);
                    }
                }

                writer.WriteStartElement("InParams");

                foreach (PropertyInfo parprop in inparams)
                {
```

```
                    object[] attr7 =
parprop.GetCustomAttributes(typeof(BaanIMSParameterAttribute), false);

                    foreach (BaanIMSParameterAttribute par in attr7)
                    {
                        object[] attr8 =
par.GetType().GetCustomAttributes(typeof(XmlRootAttribute), false);

                        foreach (XmlRootAttribute o in attr8)
                        {
                            writer.WriteStartElement(o.ElementName);
                        }

                        PropertyInfo[] pi4 = par.GetType().GetProperties();

                        foreach (PropertyInfo prop in pi4)
                        {
                            object[] attr9 =
prop.GetCustomAttributes(typeof(XmlAttributeAttribute), false);

                            foreach (XmlAttributeAttribute o in attr9)
                            {
                                writer.WriteAttributeString(o.AttributeName,
prop.GetValue(par, null).ToString());
                            }
                        }

                        object ob = parprop.GetValue(imsbo, null);

                        if (ob != null)
                        {
                            if (ob.GetType() == typeof(System.Xml.XmlDocument))
                            {
                                XmlDocument xd;
                                xd = (XmlDocument)ob;
                                xd.WriteTo(writer);
                            }
                            else
                            {
                                writer.WriteValue(ob.ToString());
                            }
                        }

                        writer.WriteEndElement();
                    }
                }

                writer.WriteEndElement();

                writer.WriteStartElement("OutParams");

                foreach (PropertyInfo parprop in outparams)
                {
                    object[] attr7 =
parprop.GetCustomAttributes(typeof(BaanIMSParameterAttribute), false);

                    foreach (BaanIMSParameterAttribute par in attr7)
                    {
                        object[] attr8 =
par.GetType().GetCustomAttributes(typeof(XmlRootAttribute), false);

                        foreach (XmlRootAttribute o in attr8)
                        {
                            writer.WriteStartElement(o.ElementName);
                        }

                        PropertyInfo[] pi4 = par.GetType().GetProperties();

                        foreach (PropertyInfo prop in pi4)
                        {
                            object[] attr9 =
prop.GetCustomAttributes(typeof(XmlAttributeAttribute), false);

                            foreach (XmlAttributeAttribute o in attr9)
                            {
```

```
                                writer.WriteAttributeString(o.AttributeName,
prop.GetValue(par, null).ToString());
                            }
                        }

                        writer.WriteEndElement();
                    }
                }

                writer.Flush();
                writer.Close();

                ret = true;
            }
            catch
            {
                ret = false;
            }

            return (ret);
        }

        private bool DeserializeResult(BaanIMSBO imsbo, string fname)
        {
            bool ret;
            string parname, partype;

            Dictionary<string, PropertyInfo> outparams = new Dictionary<string,
PropertyInfo>();

            PropertyInfo[] pi = imsbo.GetType().GetProperties();

            foreach (PropertyInfo parprop in pi)
            {
                object[] attr1 =
parprop.GetCustomAttributes(typeof(BaanIMSParameterAttribute), false);

                foreach (BaanIMSParameterAttribute par in attr1)
                {
                    if (par.ParamDirection == "out")
                        outparams.Add(par.ParamName, parprop);
                }
            }

            XmlTextReader reader = new XmlTextReader(fname);

            reader.ReadToDescendant("Function");

            while (! reader.EOF)
            {
                if(reader.LocalName == "Parameter")
                {
                    reader.MoveToAttribute("pname");
                    parname = reader.GetAttribute("pname");

                    reader.MoveToAttribute("ptype");
                    partype = reader.GetAttribute("ptype");

                    reader.MoveToElement();

                    if(outparams.ContainsKey(parname))
                    {
                        switch(partype)
                        {
                            case "string":
                                outparams[parname].SetValue(imsbo,
reader.ReadElementContentAsString(), null);
                                break;

                            case "long":
                                outparams[parname].SetValue(imsbo,
reader.ReadElementContentAsInt(), null);
                                break;

                            case "double":
```

```
                                outparams[parname].SetValue(imsbo,
reader.ReadElementContentAsDouble(), null);
                                break;

                            case "date":
                                outparams[parname].SetValue(imsbo,
reader.ReadElementContentAsDateTime(), null);
                                break;

                            case "xmlst":
                                XmlDocument xd = new XmlDocument();
                                xd.LoadXml(reader.ReadInnerXml());
                                outparams[parname].SetValue(imsbo, xd, null);
                                break;
                        }
                    }
                }

                if(reader.LocalName == "Errors")
                {
                    while (!reader.EOF)
                    {
                        if (!reader.ReadToFollowing("Error"))
                            break;

                        imsbo.Errors.Add(reader.ReadElementContentAsString());
                    }
                }

                if (!reader.Read())
                    break;
            }

            reader.Close();

            ret = true;

            return (ret);
        }

        private string GetTmpFileName()
        {
            return (tempdir + DateTime.Now.ToString("yyyyMMddHHmmssfffffff") + ".xml");
        }
    }
}
```

Classe *BWA*

```
namespace Baan.WCA
{

    public sealed class BWA
    {
        Type bt;
        Object bo;
        BaanEndpoint endpointinfo;
        int timeout;

        public BWA(BaanEndpoint bep)
        {
            endpointinfo = bep;

            bt = System.Type.GetTypeFromProgID(bep.BwClassName);
            bo = Activator.CreateInstance(bt);
        }

        public BaanEndpoint EndpointInfo
        {
            get { return endpointinfo; }
```

```
        }

        public int Timeout
        {
            get {return timeout;}

            set {
                timeout = value;
                bt.InvokeMember("Timeout", BindingFlags.SetProperty, null, bo, new
object[] { timeout });
                }
        }

        public void ParseExecFunction(string dllname, string fname, object[] fparam)
        {
            string fcall;

            fcall = fname + "(";

            for (int i=0; i < fparam.Length; i++)
            {
                fcall += fparam[i].ToString();

                if (i < fparam.Length-1)
                    fcall += ",";
            }

            fcall += ")";

            bt.InvokeMember("ParseExecFunction", BindingFlags.InvokeMethod, null, bo,
new object[] { dllname, fcall });
        }

        public string ReturnValue
        {
            get {
                string st;
                st = (string)bt.InvokeMember("ReturnValue", BindingFlags.GetProperty,
null, bo, null);
                return st;
                }
        }

        public string ReturnCall
        {
            get {
                string st;
                st = (string)bt.InvokeMember("ReturnCall", BindingFlags.GetProperty,
null, bo, null);
                return st;
                }
        }

        public int Error
        {
            get
            {
                int err;
                err = (int)bt.InvokeMember("Error", BindingFlags.GetProperty, null, bo,
null);
                return err;
            }
        }

        public void Quit()
        {
            bt.InvokeMember("Quit", BindingFlags.InvokeMethod, null, bo, new object[]
{});
        }
    }
}
```

Anexo C

Código Utilizado nos Testes de Sistema

C.1 – Código do *BO* para inserir Ordem de Venda amimsboteste

C.2 – Código da classe facilitadora OrdemVendaBaan

C.3 – Código do *BO* de Teste da Execução

C.4 – Código da classe facilitadora BBOteste

C.5 – Código 3GL utilizado no teste da DLL amimsdll00001

C.1 – Código do *BO* para inserir Ordem de Venda amimsboteste

```
|*****************************************************************************
|* amimsboteste  0  VRC B50O c  expl
|* bo teste
|* jaugusto
|* 02-11-09 [14:41]
|*****************************************************************************
|* Script Type: Library
|*****************************************************************************

        #pragma used dll ottstpapihand
        #pragma used dll oamimsxmlhand

        extern  domain  tccom.bpid     ims.in.ofbp    | Parceiro Cliente
        extern  domain  tccwoc         ims.in.cofc    | Departamento
        extern  domain  tccotp         ims.in.sotp    | Tipo de O. Venda
        extern  domain  tcseri         ims.in.seri    | Série de O. Venda
        extern  domain  tccdec         ims.in.cdec    | Cond. de Entrega
        extern  domain  tcemno         ims.in.crep    | Rep. Vendas
        extern  domain  tcdisc         ims.in.odis    | Desconto de Ordem

        extern  domain  tcorno         ims.out.orno   | Nr. de Ordem de Venda
        extern  long                   ims.Errors     | Mensagens de Erro

function extern        insert.sales.order.header()
{
        long    ret

        string  last_err(500)
        string  ret_err(500)
        string  mainsession(20)
        string  subsession(20)

        mainsession = "tdsls4500m000"

        subsession  = "tdsls4100s000"

        ims.out.orno = ""

        ret = stpapi.synchronize.dialog(mainsession, "add", ret_err)

        if ret then
                stpapi.put.field(subsession, "tdsls400.ofbp", ims.in.ofbp)
                |Inserir endereço?
                stpapi.enum.answer(subsession, "tdsls4100.1", tcyesno.no)

                stpapi.put.field(subsession, "tdsls400.cofc", ims.in.cofc)
                |Recolher predefinições da combinação depart./parceiro facturado?
                stpapi.enum.answer(subsession, "tdsls41005", tcyesno.yes)

                stpapi.put.field(subsession, "tdsls400.sotp", ims.in.sotp)
                stpapi.put.field(subsession, "tdsls400.orno", ims.in.seri)
                if not isspace(ims.in.cdec) then
                        stpapi.put.field(subsession, "tdsls400.cdec", ims.in.cdec)
                endif
                if not isspace(ims.in.crep) then
                        stpapi.put.field(subsession, "tdsls400.crep", ims.in.crep)
                endif
                stpapi.put.field(subsession, "tdsls400.odis", str$(ims.in.odis))
                stpapi.put.field(subsession, "tdsls400.cpls", "")
                stpapi.put.field(subsession, "tdsls400.pldd", "")

                ret = stpapi.insert( mainsession, true, last_err)

                if isspace(last_err) then
                        stpapi.get.field(mainsession, "tdsls400.orno", ims.out.orno)
                else
                        ret_err = last_err

                        ret = stpapi.recover(subsession, last_err)
                        ret = stpapi.recover(mainsession, last_err)
```

```
                endif

                stpapi.end.session(subsession, last_err)
                stpapi.end.session(mainsession, last_err)

        endif

        stpapi.end.session(mainsession, last_err)

        amimsxmlhand.insert.error(ret_err, ims.Errors)
}
```

C.2 – Código da classe facilitadora OrdemVendaBaan

```
namespace Baan.BOS
{
    [BaanIMSRTDDOBJECTAtribute("ImsBO", "oamimsboteste")]
    public class OrdemVendaBaan : Baan.IMS.BaanIMSBO
    {
        string  _parceirocliente;
        string  _departamento;
        string  _tipo;
        string  _serie;
        string  _condentrega;
        string  _vendedor;
        string  _numero;
        double _desconto;

        [BaanIMSFunctionAttribute("insert.sales.order.header")]
        public void InsereCabecalho() {}

        [BaanIMSParameterAttribute("ims.in.ofbp", "in", "string")]
        public string ParceiroCliente
        {
            get { return _parceirocliente; }
            set { _parceirocliente = value; }}

        [BaanIMSParameterAttribute("ims.in.cofc", "in", "string")]
        public string Departamento
        {
            get { return _departamento; }
            set { _departamento = value; }}

        [BaanIMSParameterAttribute("ims.in.sotp", "in", "string")]
        public string Tipo
        {
            get { return _tipo; }
            set { _tipo = value; }}

        [BaanIMSParameterAttribute("ims.in.seri", "in", "string")]
        public string Serie
        {
            get { return _serie; }
            set { _serie = value; }}

        [BaanIMSParameterAttribute("ims.in.cdec", "in", "string")]
        public string CondEntrega
        {
            get { return _condentrega; }
            set { _condentrega = value; }}

        [BaanIMSParameterAttribute("ims.in.crep", "in", "string")]
        public string Vendedor
        {
            get { return _vendedor; }
            set { _vendedor = value; }}

        [BaanIMSParameterAttribute("ims.in.odis", "in", "double")]
        public double Desconto
        {
            get { return _desconto; }
            set { _desconto = value; }}

        [BaanIMSParameterAttribute("ims.out.orno", "out", "string")]
        public string Numero
        {
            get { return _numero; }
            set { _numero = value; }}
    }
}
```

C.3 – Código do *BO* de Teste da Execução amimsexecteste

```
|*****************************************************************************
|* amimsexecteste  0  VRC B50O c  expl
|* Exec teste
|* jaugusto
|* 16-07-09 [14:41]
|*****************************************************************************
|* Script Type: Library
|*****************************************************************************

        extern  long            ims.Errors

        extern  domain  tcorno  imsvar.orno
        extern  long            imsvar.soactlin
        extern  long            imsvar.stime
        extern  string          imsvar.txt1(500)
        extern  long            imsvar.out1
        extern  double          imsvar.dbltype
        extern  long            imsvar.lngtype
        extern  string          imsvar.tcdate(20)

function extern        test.ims.exec.time()
{
        long                    np, ap
        string                  acln(6)

        np = xmlFindFirst("SoActLines", imsvar.soactlin)

        if np > 0 then

                np = xmlFindFirst("ActLine", np)

                if np > 0 then
                        ap = xmlGetAttribute(np, "acln", acln)
                endif
        endif

        imsvar.txt1 = "Texto Output"

        imsvar.dbltype = 0.123456

        imsvar.lngtype = 18

        imsvar.tcdate = sprintf$("%D(%04Y-%02m-%02d)T00:00:00", date.num())


        np = xmlNewDataElement("xmlstOUT1", "", imsvar.out1)

        if  np <> 0 then
                np = xmlNewDataElement("Dados", "", np)

                if  np <> 0 then
                        xmlNewNode("   OS:" & imsvar.orno & "  Pos:" & acln, XML_DATA, np)
                endif
        endif

        suspend(imsvar.stime * 1000)
}
```

C.4 – Código da classe facilitadora BBOteste

```
namespace Baan.BOS
{
    [BaanIMSRTDDOBJECTAtribute("ImsBO", "oamimsexecteste")]
    public class BBOteste : BaanIMSBO
    {
        string imsvar_orno;
        XmlDocument imsvar_soactlin;
        int imsvar_stime;
        string imsvar_txt1;
        XmlDocument imsvar_out1;
        double imsvar_dbltype;
        int imsvar_lngtype;
        DateTime imsvar_tcdate;

        [BaanIMSFunctionAttribute("test.ims.exec.time")]
        public void test_ims_exec_time(string orno, XmlDocument soactlin, int stime)
        {
            imsvar_orno = orno;
            imsvar_soactlin = soactlin;
            imsvar_stime = stime;
        }

        [BaanIMSParameterAttribute("imsvar.orno", "in", "string")]
        public string orno {
            get { return imsvar_orno; }
            set { imsvar_orno = value; }}

        [BaanIMSParameterAttribute("imsvar.soactlin", "in", "xmlst")]
        public XmlDocument soactlin {
            get { return imsvar_soactlin; }
            set { imsvar_soactlin = value; }}

        [BaanIMSParameterAttribute("imsvar.stime", "in", "long")]
        public int stime {
            get { return imsvar_stime; }
            set { imsvar_stime = value; }}

        [BaanIMSParameterAttribute("imsvar.txt1", "out", "string")]
        public string txt1 {
            get { return imsvar_txt1; }
            set { imsvar_txt1 = value; }}

        [BaanIMSParameterAttribute("imsvar.out1", "out", "xmlst")]
        public XmlDocument out1 {
            get { return imsvar_out1; }
            set { imsvar_out1 = value; }}

        [BaanIMSParameterAttribute("imsvar.dbltype", "out", "double")]
        public double dbltype {
            get { return imsvar_dbltype; }
            set { imsvar_dbltype = value; }}

        [BaanIMSParameterAttribute("imsvar.lngtype", "out", "long")]
        public int lngtype {
            get { return imsvar_lngtype; }
            set { imsvar_lngtype = value; }}

        [BaanIMSParameterAttribute("imsvar.tcdate", "out", "date")]
        public DateTime tcdate {
            get { return imsvar_tcdate; }
            set { imsvar_tcdate = value; }}
    }
}
```

C.5 – Código 3GL utilizado no teste da DLL amimsdll00001

```
|*****************************************************************************
|* amimsreqcmd  0  VRC B50O c  expl
|* Pedidos de integração por Linha de Comandos
|* jaugusto
|* 15-07-09 [14:50]
|*****************************************************************************
|* Script Type: 0
|*****************************************************************************

        #pragma used dll oamimsdll00001
        #pragma used dll ottdllbw

        #define P_WAIT         1
        #define P_NOWAIT       0

function main()
{
        long                   arg.opt, arg.reqn, arg.mode
        string                 exitcode(15)

        arg.opt = lval(argv$(1))

        on case arg.opt
                case -1:
                        exitcode = str$(process.create.request())
                        break

                case -2:
                        arg.reqn = lval(argv$(2))
                        arg.mode = lval(argv$(3))
                        exitcode = str$(amimsdll00001.execute.request(arg.reqn, arg.mode))
                        break

                case -3:
                        arg.reqn = lval(argv$(2))
                        exitcode = str$(amimsdll00001.put.request.waiting(arg.reqn))
                        break

                case -4:
                        arg.reqn = lval(argv$(2))
                        exitcode = str$(process.get.request.stat(arg.reqn))
                        break

                case -5:
                        arg.reqn = lval(argv$(2))
                        exitcode = str$(process.get.result(arg.reqn))
                        break

                case -12:
                        arg.reqn = process.create.request()
                        if arg.reqn > 0 then
                                exitcode = str$(arg.reqn) & "|" &
str$(amimsdll00001.execute.request(arg.reqn, P_NOWAIT))
                        else
                                exitcode = "0"
                        endif
                        break

                case -13:
                        arg.reqn = process.create.request()
                        if arg.reqn > 0 then
                                exitcode = str$(arg.reqn) & "|" &
str$(amimsdll00001.put.request.waiting(arg.reqn))
                        else
                                exitcode = "0"
                        endif
                        break

                case -25:
                        arg.reqn = lval(argv$(2))
                        exitcode = str$(amimsdll00001.execute.request(arg.reqn, P_WAIT))
                        exitcode = exitcode & "|" &  str$(process.get.result(arg.reqn))
```

```
                        break

                case -125:
                        arg.reqn = process.create.request()
                        if arg.reqn > 0 then
                                exitcode = str$(arg.reqn) & "|" &
str$(amimsdll00001.execute.request(arg.reqn, P_WAIT))
                                exitcode = exitcode & "|" &
str$(process.get.result(arg.reqn))
                        else
                                exitcode = "0"
                        endif
                        break

                default:
                        exitcode = "0"
                        break
        endcase

        message(exitcode)
}

function long process.create.request()
{
        domain  tcmcs.str215m  arg.fname
        domain  tcdsca         arg.dsca
        long                   arg.mxpt, arg.mxrn, arg.hist, arg.delfile
        long                   ret

        arg.dsca = argv$(2)
        arg.fname = argv$(3) & (isspace(argv$(4)) ? "" : "." ) & argv$(4)
        arg.mxpt = lval(argv$(5))
        arg.mxrn = lval(argv$(6))
        arg.hist = lval(argv$(7))
        arg.delfile = lval(argv$(8))

        ret = 0

        if file.exists(arg.fname) then
                ret = amimsdll00001.create.request(arg.fname, arg.dsca, arg.mxpt,
arg.mxrn, arg.hist, arg.delfile)
        endif

        return(ret)
}

function long process.get.result(domain tcmcs.long p.reqn)
{
        domain  tcmcs.str215m  arg.fname
        long                   ret

        arg.fname = argv$(3) & (isspace(argv$(4)) ? "" : "." ) & argv$(4)

        ret = 0

        ret = amimsdll00001.get.request.proc.result(p.reqn, arg.fname)

        return(ret)
}

function long process.get.request.stat(domain tcmcs.long p.reqn)
{
        domain  amims.stat     aux.stat
        long                   ret

        ret = 0

        if amimsdll00001.get.request.stat(p.reqn, aux.stat) then
                ret = etol(aux.stat)
        endif

        return(ret)
}

function long file.exists(domain tcmcs.str215m p.fname)
{
```

```
        long aux.fp

        aux.fp = seq.open.local(p.fname, "rt", false)
        if aux.fp > 0 then
                seq.close.local(aux.fp)
                return(1)
        endif

        return(0)
}
```

C.6 – Código do Exemplo de Utilização da Plataforma

```
public partial class Form1 : Form
{
    string imstempdir;
    IBaanEndpointCat bec;
    BaanEndpoint be;

    public Form1()
    {
        InitializeComponent();

        string bwcdir;

        try
        {
            imstempdir = ConfigurationManager.AppSettings["ImsTempDir"].ToString();
            bwcdir = ConfigurationManager.AppSettings["BwcDir"].ToString();

            BwConfigManager bwcman = new BwConfigManager(bwcdir);

            BaanEndpointCatFactory becfact = new BaanEndpointCatFactory(bwcman);

            bec = becfact.CreateCatalog();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }


    private void btIniSess_Click(object sender, EventArgs e)
    {
        try
        {
            be = bec.GetEndpoint(txsrv1.Text, txcomp1.Text, txuser1.Text);

            if (be == null)
                MessageBox.Show("Combinação Serv./Comp./Utiliz. Inválida");
            else
            {
                if (be.Password != txpw1.Text)
                {
                    btCriarOV.Enabled = false;
                    MessageBox.Show("Password errada");
                }
                else
                    btCriarOV.Enabled = true;
            }
        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    private void btCriarOV_Click(object sender, EventArgs e)
    {
        int req;
        bool execret;
        bool gresret;

        OrdemVendaBaan ov = new OrdemVendaBaan();

        ov.ParceiroCliente = txParceiro.Text;
        ov.Departamento = txDepartam.Text;
        ov.Tipo = txTipo.Text;
        ov.Serie = txSerie.Text;
        ov.Vendedor = txVendedor.Text;
        ov.Desconto = Convert.ToDouble(txDesconto.Text);
        ov.CondEntrega = txCEnt.Text;
```

```
            IMSCallDetails imc = new IMSCallDetails();

            imc.Description = "Teste O.Venda";
            imc.MaxTime = 30;
            imc.MaxRun = 3;
            imc.History = true;
            imc.DelLocFile = false;

            try
            {
                BaanIMS bims = new BaanIMS(be, imstempdir);

                bims.CreateExecuteAndGetResult(ov, imc, out req, out execret, out
gresret);

                bims = null;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

            MessageBox.Show(ov.Numero);

            foreach (string err in ov.Errors)
                MessageBox.Show(err);
        }

    }
```

Anexo D

Testes de Sistema Funcionais

Especificações de Casos de Teste

D.1 – Casos de Teste Funcionais da DLL amimsdll00001

D.2 – Casos de Teste Funcionais das Sessões do módulo IMS

D.1 – Casos de Teste Funcionais da DLL amimsdll00001

Identificador de Teste:	**TS1**				
Requisito a testar:	RF01	Criar pedido de integração			
ID do Caso de Teste:	**TS1_1**				
Objectivo:	Exercitar a função de criar pedido, utilizando valores válidos.				
Procedimento:	Efectuar uma chamada à função **create.request** com os valores de entrada especificados e verificar o valor de retorno da função. Verificar com a sessão amims0501m000 se os dados de controlo do pedido estão de acordo com os parâmetros especificados nos valores de entrada.				
Valores de Entrada:	("c:\ims\testes\BaanIMS.xml", "Teste TS1_1", 5, 3, 1, 0)				
Resultado Esperado:	> 0	**Resultado Obtido:**	Valor de retorno: 1402 Verificação com a sessão amims0501m000: OK		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS1_2**				
Objectivo:	Exercitar a função de criar pedido, utilizando valores inválidos (1º argumento).				
Procedimento:	Efectuar uma chamada à função **create.request** com os valores de entrada especificados e verificar o valor de retorno da função.				
Valores de Entrada:	(**""**, "Teste TS1_2", 5, 3, 1, 0)				
Resultado Esperado:	0	**Resultado Obtido:**	0		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS1_3**				
Objectivo:	Exercitar a função de criar pedido, utilizando valores inválidos (3º argumento).				
Procedimento:	Efectuar uma chamada à função **create.request** com os valores de entrada especificados e verificar o valor de retorno da função.				
Valores de Entrada:	("c:\ims\testes\BaanIMS.xml", "Teste TS1_3", **-1**, 3, 1, 0)				
Resultado Esperado:	0	**Resultado Obtido:**	0		
		Resultado:	OK	**Defeito nr.:**	

ID do Caso de Teste:	**TS1_4**				
Objectivo:	Exercitar a função de criar pedido, utilizando valores inválidos (4º argumento).				
Procedimento:	Efectuar uma chamada à função **create.request** com os valores de entrada especificados e verificar o valor de retorno da função.				
Valores de Entrada:	("c:\ims\testes\BaanIMS.xml", "Teste TS1_4", 5, **-1**, 1, 0)				
Resultado Esperado:	0	**Resultado Obtido:**	0		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS2**				
Requisito a testar:	RF02	Executar pedido de integração			
ID do Caso de Teste:	**TS2_1**				
Objectivo:	Exercitar a função de colocar em execução um pedido com o estado "Criado" e retornando só após a execução do pedido.				
Procedimento:	Efectuar uma chamada à função **execute.request** com os valores de entrada especificados e verificar o valor de retorno da função. Deve ser escolhido um número de pedido cujo estado seja "Criado". Após a execução verificar se o estado do pedido obedece a uma transição válida.				
Valores de Entrada:	(1402, 1)				
Resultado Esperado:	1	**Resultado Obtido:**	1		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS2_2**				
Objectivo:	Exercitar a função de colocar em execução um pedido com o estado "Espera" e retornando só após a execução do pedido.				
Procedimento:	Efectuar uma chamada à função **execute.request** com os valores de entrada especificados e verificar o valor de retorno da função. Deve ser escolhido um número de pedido cujo estado seja "Espera". Após a execução verificar se o estado do pedido obedece a uma transição válida.				
Valores de Entrada:	(1385, 1)				
Resultado Esperado:	1	**Resultado Obtido:**	1		
		Resultado:	OK	**Defeito nr.:**	

ID do Caso de Teste:	**TS2_3**				
Objectivo:	Exercitar a função de colocar em execução um pedido com o estado "Criado" e retornando sem esperar pela execução do pedido.				
Procedimento:	Efectuar uma chamada à função **execute.request** com os valores de entrada especificados e verificar o valor de retorno da função. Deve ser escolhido um número de pedido cujo estado seja "Criado". Após a execução verificar se o estado do pedido obedece a uma transição válida.				
Valores de Entrada:	(1403, 0)				
Resultado Esperado:	1	**Resultado Obtido:**	1		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS2_4**				
Objectivo:	Exercitar a função de colocar em execução um pedido com o estado "Espera" e retornando sem esperar pela execução do pedido.				
Procedimento:	Efectuar uma chamada à função **execute.request** com os valores de entrada especificados e verificar o valor de retorno da função. Deve ser escolhido um número de pedido cujo estado seja " Espera". Após a execução verificar se o estado do pedido obedece a uma transição válida.				
Valores de Entrada:	(1390, 0)				
Resultado Esperado:	1	**Resultado Obtido:**	1		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS2_5**				
Objectivo:	Exercitar a função de colocar em execução um pedido com o estado "Em Execução", "Com Erros", "Executado" e "Cancelado" (valores inválidos).				
Procedimento:	Efectuar uma chamada à função **execute.request** com os valores de entrada especificados e verificar o valor de retorno da função. Devem ser escolhidos pedidos cujos estados sejam os referidos no objectivo. Após a execução verificar se o estado do pedido permanece inalterado.				
Valores de Entrada:	(1400, 0) (1369, 0) (1395, 0) (1388, 0)				
Resultado Esperado:	0 0 0 0	**Resultado Obtido:**	0 0 0 0		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS2_6**				
Objectivo:	Exercitar a função de colocar em execução um pedido com um número inexistente (valor inválido).				
Procedimento:	Efectuar uma chamada à função **execute.request** com os valores de entrada especificados e verificar o valor de retorno da função.				
Valores de Entrada:	(0, 0)				
Resultado Esperado:	0	**Resultado Obtido:**	0		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS3**				
Requisito a testar:	RF03	Colocar pedido de integração no estado "Espera"			
ID do Caso de Teste:	**TS3_1**				
Objectivo:	Exercitar a função de colocar no estado "Espera" um pedido cujo estado seja igual a "Criado".				
Procedimento:	Efectuar uma chamada à função **put.request.waiting** com os valores de entrada especificados e verificar o valor de retorno da função. Deve ser escolhido um número de pedido cujo estado seja "Criado". Após a execução verificar se o estado do pedido obedece a uma transição válida.				
Valores de Entrada:	(1404)				
Resultado Esperado:	1	**Resultado Obtido:**	1		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS3_2**				
Objectivo:	Exercitar a função de colocar no estado "Espera" um pedido com o estado "Espera", "Em Execução", "Com Erros", "Executado" e "Cancelado" (valores inválidos).				
Procedimento:	Efectuar uma chamada à função **put.request.waiting** com os valores de entrada especificados e verificar o valor de retorno da função. Devem ser escolhidos pedidos cujos estados sejam os referidos no objectivo. Após a execução verificar se o estado do pedido permanece inalterado.				
Valores de Entrada:	(1361) (1304) (1369) (1395) (1388)				
Resultado Esperado:	0 0 0 0 0	**Resultado Obtido:**	0 0 0 0 0		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS3_3**				
Objectivo:	Exercitar a função de colocar um pedido no estado "Espera" com um número de pedido inexistente (valor inválido).				
Procedimento:	Efectuar uma chamada à função **put.request.waiting** com os valores de entrada especificados e verificar o valor de retorno da função.				
Valores de Entrada:	(0)				
Resultado Esperado:	0	**Resultado Obtido:**	0		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS4**				
Requisito a testar:	RF04	Consultar estado de um pedido de integração			
ID do Caso de Teste:	**TS4_1**				
Objectivo:	Exercitar a função de consultar o estado de um pedido de integração.				
Procedimento:	Efectuar uma chamada à função **get.request.stat.const** com o valor de entrada especificado e verificar o valor de retorno da função. Deve ser escolhido um número de pedido para cada estado possível.				
Valores de Entrada:	(1405) (1361) (1400) (1395) (1369) (1388)				
Resultado Esperado:	1 2 3 4 5 6	**Resultado Obtido:**	1 2 3 4 5 6		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS4_2**				
Objectivo:	Exercitar a função de consultar o estado de um pedido de integração com um número de pedido inexistente (valor inválido).				
Procedimento:	Efectuar uma chamada à função **get.request.stat.const** com os valores de entrada especificados e verificar o valor de retorno da função.				
Valores de Entrada:	(0)				
Resultado Esperado:	0	**Resultado Obtido:**	0		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS5**				
Requisito a testar:	RF05	Obter resultado de um pedido de integração			
ID do Caso de Teste:	**TS5_1**				
Objectivo:	Exercitar a função de obter o resultado do processamento de um pedido de integração.				
Procedimento:	Efectuar uma chamada à função **get.request.proc.result** com o valor de entrada especificado e verificar o valor de retorno da função. Deve ser escolhido um número de pedido cujo estado seja "Executado". Após a execução verificar a existência do documento *XML* de saída.				
Valores de Entrada:	(1395, "c:\ims\testes\TS5_1out.xml")				
Resultado Esperado:	1	**Resultado Obtido:**	Valor de retorno: 1 Verificação do ficheiro: OK		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS5_2**				
Objectivo:	Exercitar a função de obter o resultado do processamento de um pedido de integração.				
Procedimento:	Efectuar uma chamada à função **get.request.proc.result** com o valor de entrada especificado e verificar o valor de retorno da função. Deve ser escolhido um número de pedido cujo estado seja "Com Erros". Após a execução verificar a existência do documento *XML* de saída.				
Valores de Entrada:	(1369, "c:\ims\testes\TS5_2out.xml")				
Resultado Esperado:	1	**Resultado Obtido:**	Valor de retorno: 1 Verificação do ficheiro: OK		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS5_3**				
Objectivo:	Exercitar a função de obter o resultado do processamento de um pedido de integração com um número de pedido inexistente (valor inválido).				
Procedimento:	Efectuar uma chamada à função **get.request.proc.result** com o valor de entrada especificado e verificar o valor de retorno da função.				
Valores de Entrada:	(0, "c:\ims\testes\TS5_3out.xml")				
Resultado Esperado:	0	**Resultado Obtido:**	0		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS5_4**				
Objectivo:	Exercitar a função de obter o resultado do processamento de um pedido de integração sem especificar o ficheiro de saída (valor inválido).				
Procedimento:	Efectuar uma chamada à função **get.request.proc.result** com o valor de entrada especificado e verificar o valor de retorno da função.				
Valores de Entrada:	(1395, "")				
Resultado Esperado:	0	**Resultado Obtido:**	0		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS7**				
Requisito a testar:	RF07	Criar e executar pedido de integração numa única chamada			
ID do Caso de Teste:	**TS7_1**				
Objectivo:	Exercitar a função de criar e executar pedido numa única chamada, utilizando valores válidos.				
Procedimento:	Efectuar uma chamada à função **create.and.execute.request** com os valores de entrada especificados e verificar os parâmetros de saída da função. Verificar com a sessão amims0501m000 se os dados de controlo do pedido estão de acordo com os parâmetros especificados nos valores de entrada.				
Valores de Entrada:	(“c:\ims\testes\BaanIMS.xml”, “Teste TS7_1”, 5, 3, 1, 0)				
Resultado Esperado:	Parâmetro Nr. Pedido: > 0 Parâmetro Execução: 1	**Resultado Obtido:**	Pedido: 1414 Execução: 1 Verificação com a sessão amims0501m000: OK		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS7_2**				
Objectivo:	Exercitar a função de criar e executar pedido numa única chamada, utilizando valores inválidos (1º argumento).				
Procedimento:	Efectuar uma chamada à função **create.and.execute.request** com os valores de entrada especificados e verificar os parâmetros de saída da função.				
Valores de Entrada:	(**“”**, “Teste TS7_2”, 5, 3, 1, 0)				
Resultado Esperado:	Parâmetro Nr. Pedido: 0 Parâmetro Execução: 0	**Resultado Obtido:**	Pedido: 0 Execução: 0		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS7_3**				
Objectivo:	Exercitar a função de criar e executar pedido numa única chamada, utilizando valores inválidos (3º argumento).				
Procedimento:	Efectuar uma chamada à função **create.and.execute.request** com os valores de entrada especificados e verificar os parâmetros de saída da função.				
Valores de Entrada:	(“c:\ims\testes\BaanIMS.xml”, “Teste TS7_3”, **-1**, 3, 1, 0)				
Resultado Esperado:	Parâmetro Nr. Pedido: 0 Parâmetro Execução: 0	**Resultado Obtido:**	Pedido: 0 Execução: 0		
		Resultado:	OK	**Defeito nr.:**	

ID do Caso de Teste:	**TS7_4**				
Objectivo:	Exercitar a função de criar e executar pedido numa única chamada, utilizando valores inválidos (4º argumento).				
Procedimento:	Efectuar uma chamada à função **create.and.execute.request** com os valores de entrada especificados e verificar os parâmetros de saída da função.				
Valores de Entrada:	("c:\ims\testes\BaanIMS.xml", "Teste TS7_4", 5, **-1**, 1, 0)				
Resultado Esperado:	Parâmetro Nr. Pedido: 0 Parâmetro Execução: 0	**Resultado Obtido:**	Pedido: 0 Execução: 0		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS8**				
Requisito a testar:	RF08	Criar e colocar o pedido em Espera numa única chamada			
ID do Caso de Teste:	**TS8_1**				
Objectivo:	Exercitar a função de criar e colocar em espera um pedido numa única chamada, utilizando valores válidos.				
Procedimento:	Efectuar uma chamada à função **create.and.put.waiting.request** com os valores de entrada especificados e verificar os parâmetros de saída da função. Verificar com a sessão amims0501m000 se os dados de controlo do pedido estão de acordo com os parâmetros especificados nos valores de entrada.				
Valores de Entrada:	("c:\ims\testes\BaanIMS.xml", "Teste TS8_1", 5, 3, 1, 0)				
Resultado Esperado:	Parâmetro Nr. Pedido: > 0 Parâmetro Espera: 1	**Resultado Obtido:**	Pedido: 1415 Espera: 1 Verificação com a sessão amims0501m000: OK		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS8_2**				
Objectivo:	Exercitar a função de criar e colocar em espera um pedido, utilizando valores inválidos (1º argumento).				
Procedimento:	Efectuar uma chamada à função **create.and.put.waiting.request** com os valores de entrada especificados e verificar os parâmetros de saída da função.				
Valores de Entrada:	(**""**, "Teste TS8_2", 5, 3, 1, 0)				
Resultado Esperado:	Parâmetro Nr. Pedido: 0 Parâmetro Espera: 0	**Resultado Obtido:**	Pedido: 0 Espera: 0		
		Resultado:	OK	**Defeito nr.:**	

<table>
<tr><th>ID do Caso de Teste:</th><th colspan="4">TS8_3</th></tr>
<tr><td>Objectivo:</td><td colspan="4">Exercitar a função de criar e colocar em espera um pedido, utilizando valores inválidos (3º argumento).</td></tr>
<tr><td>Procedimento:</td><td colspan="4">Efectuar uma chamada à função create.and.put.waiting.request com os valores de entrada especificados e verificar os parâmetros de saída da função.</td></tr>
<tr><td>Valores de Entrada:</td><td colspan="4">(“c:\ims\testes\BaanIMS.xml”, “Teste TS8_3”, -1, 3, 1, 0)</td></tr>
<tr><td rowspan="2">Resultado Esperado:</td><td rowspan="2">Parâmetro Nr. Pedido: 0
Parâmetro Espera: 0</td><td>Resultado Obtido:</td><td colspan="2">Pedido: 0
Espera: 0</td></tr>
<tr><td>Resultado:</td><td>OK</td><td>Defeito nr.:</td><td></td></tr>
<tr><th>ID do Caso de Teste:</th><th colspan="4">TS8_4</th></tr>
<tr><td>Objectivo:</td><td colspan="4">Exercitar a função de criar e colocar em espera um pedido, utilizando valores inválidos (4º argumento).</td></tr>
<tr><td>Procedimento:</td><td colspan="4">Efectuar uma chamada à função create.and.put.waiting.request com os valores de entrada especificados e verificar os parâmetros de saída da função.</td></tr>
<tr><td>Valores de Entrada:</td><td colspan="4">(“c:\ims\testes\BaanIMS.xml”, “Teste TS8_4”, 5, -1, 1, 0)</td></tr>
<tr><td rowspan="2">Resultado Esperado:</td><td rowspan="2">Parâmetro Nr. Pedido: 0
Parâmetro Espera: 0</td><td>Resultado Obtido:</td><td colspan="2">Pedido: 0
Espera: 0</td></tr>
<tr><td>Resultado:</td><td>OK</td><td>Defeito nr.:</td><td></td></tr>
</table>

Identificador de Teste:	**TS11**				
Requisito a testar:	RF11	Validador de transições de estado do pedido de integração			
ID do Caso de Teste:	**TS11_1**				
Objectivo:	Exercitar as transições de estado de um pedido de integração com valores válidos.				
Procedimento:	Efectuar diversas chamadas à função **check.stat.trans** com os valores de entrada especificados e verificar o valor de retorno da função.				
Valores de Entrada:	(amims.stat.created, amims.stat.waiting) (amims.stat.created, amims.in.proc) (amims.stat.waiting, amims.in.proc) (amims.in.proc, amims.stat.waiting) (amims.in.proc, amims.stat.processed) (amims.in.proc, amims.stat.proc.errors) (amims.in.proc, amims.stat.canceled)				
Resultado Esperado:	1	**Resultado Obtido:**	1		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS11_2**				
Objectivo:	Exercitar as transições de estado de um pedido de integração com valores inválidos.				
Procedimento:	Efectuar diversas chamadas à função **check.stat.trans** com os valores de entrada especificados e verificar o valor de retorno da função.				
Valores de Entrada:	(amims.stat.created, amims.stat.created) (amims.stat.created, amims.stat.processed) (amims.stat.created, amims.stat.proc.errors) (amims.stat.created, amims.stat.canceled) (amims.stat.waiting, amims.stat.waiting) (amims.stat.waiting, amims.created) (amims.stat.waiting, amims.canceled) (amims.stat.waiting, amims.processed) (amims.stat.waiting, amims.proc.errors) (amims.in.proc, amims.in.proc) (amims.in.proc, amims.stat.created) (amims.canceled, *) (amims.processed, *) (amims.proc.errors, *)				
Resultado Esperado:	0	**Resultado Obtido:**	0		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS12**				
Requisito a testar:	RF12	Lançamento e monitorização do processo de execução			
ID do Caso de Teste:	**TS12_1**				
Objectivo:	Verificar o controlo da execução de um pedido com um tempo máximo de execução superior ao seu tempo de processamento.				
Procedimento:	Utilizar o *BO* de teste **oamimsexecteste** definindo no tempo de processamento (imsvar.stime) o valor especificado em valores de entrada. O pedido deve ser criado e executado com os parâmetros especificados em valores de entrada.				
Valores de Entrada:	("c:\ims\testes\TS12_1.xml","Teste TS12_1", **10**, 3, 1, 0) imsvar.stime = **9** s				
Resultado Esperado:	Após processamento, o pedido deve apresentar o estado "Executado".	**Resultado Obtido:**	Estado do pedido: Executado		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS12_2**				
Objectivo:	Verificar o controlo da execução de um pedido com um tempo máximo de execução inferior ao seu tempo de processamento.				
Procedimento:	Idem TS12_1				
Valores de Entrada:	("c:\ims\testes\ c:\ims\testes\TS12_2.xml","Teste TS12_2", **8**, 3, 1, 0) imsvar.stime = **9** s				
Resultado Esperado:	O pedido deve ficar no estado "Espera".	**Resultado Obtido:**	Estado do pedido: Espera		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS12_3**				
Objectivo:	Verificar o controlo da execução de um pedido com tempo máximo de execução definido com o valor zero.				
Procedimento:	Idem TS12_1 + colocar em parâmetros do *IMS* o valor máximo para execução dos processos igual a 10 segundos.				
Valores de Entrada:	("c:\ims\testes\ TS12_3.xml","Teste TS12_3", **0**, 3, 1, 0) imsvar.stime = **9** s				
Resultado Esperado:	Após processamento, o pedido deve apresentar o estado "Executado".	**Resultado Obtido:**	Estado do pedido: Executado		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS12_4**				
Objectivo:	Verificar o controlo da execução de um pedido que ultrapasse o número máximo de tentativas para a sua execução.				
Procedimento:	Efectuar 2 chamadas à função **execute.request** utilizando o número de pedido (X) criado pelo caso de teste TS12_2.				
Valores de Entrada:	(X, 1)				
Resultado Esperado:	O pedido deve ficar no estado "Cancelado".	**Resultado Obtido:**	Estado do pedido: Cancelado		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS13**				
Requisito a testar:	RF13	Processar pedido de integração			
ID do Caso de Teste:	**TS13_1**				
Objectivo:	Verificar o controlo da execução de um pedido de integração em que o respectivo documento *XML* apresenta uma estrutura e valores válidos.				
Procedimento:	Utilizar o documento *XML* resultante da serialização do pedido de integração realizado pelo exemplo da secção 5.6 (*BO* de teste oamimsboteste), criar e executar o pedido com os parâmetros especificados em valores de entrada. Os atributos do pedido e o resultado podem ser obtidos através da sessão amims0501m000 e respectivas opções.				
Valores de Entrada:	("c:\ims\testes\TS13_1.xml","Teste TS13_1", 10, 3, 1, 0)				
Resultado Esperado:	Após processamento, o pedido deve apresentar o estado "Executado" e o documento *XML* de resultado deve apresentar o número de ordem de venda criada.	**Resultado Obtido:**	Estado do pedido: Executado Ordem de venda: 110006506		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS13_2**				
Objectivo:	Verificar o controlo da execução de um pedido de integração em que o respectivo documento *XML* apresenta uma estrutura inválida.				
Procedimento:	Idem TS13_1, porém deverão ser feitas alterações ao documento *XML* antes da execução de acordo com o seguinte: - modificar o nome dos atributos e executar (deverá ser modificado apenas um atributo em cada execução) ; - acrescentar parâmetros de entrada e saída que não existam no *BO*; - eliminar elementos.				
Valores de Entrada:	("c:\ims\testes\TS13_2.xml","Teste TS13_2", 10, 3, 1, 0)				
Resultado Esperado:	O pedido deve apresentar o estado "Com Erros" e o campo "última mensagem" deverá estar preenchido com a descrição do erro.	**Resultado Obtido:**	Estado do pedido: Com Erros Mensagem: "XML de entrada com estrutura incorrecta"		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS13_3**				
Objectivo:	Verificar o controlo da execução de um pedido de integração em que o respectivo documento *XML* apresenta atributos com valores inválidos.				
Procedimento:	Idem TS13_1, porém deverão ser feitas alterações ao documento *XML* antes da execução de acordo com o seguinte: - alterar o nome do *BO* para um valor inexistente; - alterar para um nome inexistente a função que se pretende executar do *BO*.				
Valores de Entrada:	("c:\ims\testes\TS13_3.xml","Teste TS13_3", 10, 3, 1, 0)				
Resultado Esperado:	O pedido deve apresentar o estado "Com Erros" e o campo do pedido "última mensagem" deverá estar preenchido com a descrição do erro.	**Resultado Obtido:**	Estado do pedido: Com Erros Mensagem: "Objecto ou função não encontrados"		
		Resultado:	OK	**Defeito nr.:**	

D.2 – Casos de Teste Funcionais das Sessões do módulo IMS

Identificador de Teste:	**TS6**				
Requisito a testar:	RF06	Sessão para tratamento de pedidos de integração pendentes			
ID do Caso de Teste:	**TS6_1**				
Objectivo:	Exercitar a funcionalidade da sessão para processamento de pedidos pendentes.				
Procedimento:	Iniciar a sessão amims0201m000, preencher os campos com os valores de entrada e accionar a opção continuar. Escolher um intervalo de números de pedido que inclua não só o estado "Espera" mas também outros estados. Após a execução verificar com a sessão amims0501m000 se os pedidos seleccionados com o estado "Espera" efectuaram uma transição válida e se o estado dos restantes não se alterou.				
Valores de Entrada:	Pedido De: 1353 A: 1358				
Resultado Esperado:	Os pedidos com estado "Espera" devem realizar uma transição válida. Os restantes não devem alterar o seu estado.	**Resultado Obtido:**	Os pedidos com estado "Espera" passaram para o estado "Executado". Os restantes não se alteraram.		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS10**				
Requisito a testar:	RF10	Sessão que permita a gestão dos pedidos de integração de forma visual			
ID do Caso de Teste:	**TS10_1**				
Objectivo:	Exercitar a funcionalidade de criar um pedido de integração.				
Procedimento:	Iniciar a sessão amims0501m000 e escolher a opção "Novo Pedido" do menu específico. Preencher os campos da sessão amims0101m000 e activar a opção "Novo Pedido". Verificar na sessão amims0501m000 se os dados do pedido estão de acordo com os valores inseridos na sessão amims0101m000.				
Valores de Entrada:	Descrição: "Teste TS10_1" Tempo max. Exec. Proc.: 5 Nr. max tentativas: 3 Histórico: Sim Última mensagem: ""				
Resultado Esperado:	Deverá aparecer a mensagem "Foi criado o pedido:" + nr. do pedido criado.	**Resultado Obtido:**	Foi criado o pedido: 1416 Verificação com a sessão amims0501m000: OK		
		Resultado:	OK	**Defeito nr.:**	

ID do Caso de Teste:	**TS10_2**				
Objectivo:	Exercitar a funcionalidade de colocar um pedido de integração em "Espera".				
Procedimento:	Iniciar a sessão amims0501m000 e após seleccionar um registo com o estado "Criado", escolher a opção "Espera" do menu específico.				
Valores de Entrada:	Seleccionar um registo com o estado igual a "Criado".				
Resultado Esperado:	O pedido deve ficar no estado "Espera".	**Resultado Obtido:**	Estado do pedido: Espera		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS10_3**				
Objectivo:	Exercitar a funcionalidade de colocar um pedido de integração em execução.				
Procedimento:	Iniciar a sessão amims0501m000 e após seleccionar um registo com o estado "Criado" ou "Espera", escolher a opção "Executar" do menu específico.				
Valores de Entrada:	Seleccionar um registo com o estado igual a "Criado" ou "Espera".				
Resultado Esperado:	O pedido deve efectuar uma transição válida.	**Resultado Obtido:**	Estado do pedido: Executado		
		Resultado:	OK	**Defeito nr.:**	
ID do Caso de Teste:	**TS10_4**				
Objectivo:	Exercitar a funcionalidade de obter o resultado de um pedido de integração.				
Procedimento:	Iniciar a sessão amims0501m000 e após seleccionar um registo com o estado "Executado", abrir a sessão de detalhe amims0101s000 e escolher a opção "Fich. Saída".				
Valores de Entrada:	Seleccionar um registo com o estado igual a "Executado".				
Resultado Esperado:	O documento *XML* de resultado do processamento do pedido deverá aparecer na aplicação associada à extensão ".xml".	**Resultado Obtido:**	O documento *XML* de resultado foi apresentado.		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS14**				
Requisito a testar:	RF14	Histórico das transições de estado dos pedidos de integração			
ID do Caso de Teste:	**TS14_1**				
Objectivo:	Verificação da informação de histórico das transições de estado dos pedidos de integração com o campo histórico igual a verdadeiro.				
Procedimento:	Iniciar a sessão amims0501m000 e após seleccionar um registo, escolher a opção "Histórico" do menu específico. Verificar se a informação requerida existe. Este processo deve ser repetido para pedidos em diversos estados.				
Valores de Entrada:	Seleccionar um registo de pedido de integração com histórico.				
Resultado Esperado:	Deve estar presente a informação sobre o estado, a data hora minuto e segundo, o utilizador, e no caso de erro a descrição do mesmo.	**Resultado Obtido:**	A informação de histórico está presente		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS15**				
Requisito a testar:	RF15	Sessão de consulta de histórico dos pedidos de integração			
ID do Caso de Teste:	**TS15_1**				
Objectivo:	Exercitar as funcionalidades da sessão de consulta do histórico de pedidos de integração.				
Procedimento:	Iniciar a sessão amims0502m000 e verificar a informação de histórico. Efectuar uma pesquisa e aplicar filtros sobre os diversos registos existentes.				
Valores de Entrada:	Registos de histórico.				
Resultado Esperado:	Deve estar presente a informação de histórico e ser possível pesquisar e aplicar filtros sobre os registos existentes.	**Resultado Obtido:**	A informação de histórico está presente É possível pesquisar e filtrar informação.		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS16**				
Requisito a testar:	RF16	Parametrização do tempo máximo pré-definido para a execução dos pedidos de integração			
ID do Caso de Teste:	**TS16_1**				
Objectivo:	Verificar a existência da parametrização do tempo máximo pré-definido para a execução dos pedidos.				
Procedimento:	Iniciar a sessão amims0100m000 e verificar a existência e a possibilidade de manutenção do parâmetro referido no objectivo.				
Valores de Entrada:	Registo de parâmetros existente na sessão.				
Resultado Esperado:	Deve existir um campo no ecrã da sessão para manter o tempo máximo pré-definido para a execução dos pedidos de integração.	**Resultado Obtido:**	O parâmetro existe e pode ser mantido.		
		Resultado:	OK	**Defeito nr.:**	

Identificador de Teste:	**TS17**				
Requisito a testar:	RF17	Parametrização da localização do repositório XML de pedidos e resultados de integração			
ID do Caso de Teste:	**TS17_1**				
Objectivo:	Verificar a existência da parametrização referente ao local de repositório *XML* dos pedidos e resultados.				
Procedimento:	Iniciar a sessão amims0100m000 e verificar a existência e a possibilidade de manutenção do parâmetro referido no objectivo.				
Valores de Entrada:	Registo de parâmetros existente na sessão.				
Resultado Esperado:	Deve existir um campo no ecrã da sessão para manter o local de repositório *XML* dos pedidos e resultados	**Resultado Obtido:**	O parâmetro existe quer para o local dos documentos *XML* de entrada quer para os de resultados.		
		Resultado:	OK	**Defeito nr.:**	

<table>
<tr><td>Identificador de Teste:</td><td>TS18</td><td colspan="4"></td></tr>
<tr><td>Requisito a testar:</td><td>RF18</td><td colspan="4">Sessão para manutenção de parâmetros do módulo IMS</td></tr>
<tr><td>ID do Caso de Teste:</td><td colspan="5">TS18_1</td></tr>
<tr><td>Objectivo:</td><td colspan="5">Exercitar a funcionalidade da sessão de manutenção dos parâmetros do módulo IMS.</td></tr>
<tr><td>Procedimento:</td><td colspan="5">Iniciar a sessão amims0100m000, alterar e guardar os parâmetros disponíveis e consultar os restantes.</td></tr>
<tr><td>Valores de Entrada:</td><td colspan="5">Registo de parâmetros existente na sessão.</td></tr>
<tr><td rowspan="2">Resultado Esperado:</td><td rowspan="2">Deve ser possível alterar o tempo máximo pré-definido para a execução dos pedidos de integração e o número máximo de processos a executar em simultâneo pela sessão de tratamento de pedidos pendentes. Os restantes campos devem ser apenas de consulta.</td><td>Resultado Obtido:</td><td colspan="3">Os parâmetros referidos em resultado esperado existem e podem ser mantidos. Os restantes apenas podem ser consultados.</td></tr>
<tr><td>Resultado:</td><td>OK</td><td>Defeito nr.:</td><td></td></tr>
</table>