UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

ESCOLA POLITÉCNICA

DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

ESTUDO SOBRE INTEGRAÇÃO DE SISTEMAS COMPUTACIONAIS BASEADA EM ARQUITETURA ORIENTADA A SERVIÇOS

Autor:

Marcel Siqueira Antunes

Orientador:

Prof. Antônio Claudio Gómez de Sousa, M. Sc.

Examinador:

Aloysio de Castro Pinto Pedroza, D. Sc.

Examinador:

Marcelo Luiz Drumond Lanza, M. Sc.

DEL

Junho de 2008

Dedicatória

Aos meus familiares, que sempre estiveram

ao meu lado durante essa longa jornada

de me tornar Engenheiro.

Agradecimentos

Agradeço ao meu professor e orientador Antônio Claudio, que me ajudou durantes os vários meses em que estive desenvolvendo este Projeto de Fim de Curso.

Agradeço também aos meus familiares e amigos pelo apoio e compreensão ao longo desses anos de faculdade.

Agradeço especialmente à minha namorada Érica, por todo o apoio durante os dias difíceis de faculdade, Projeto Final e vida.

Resumo

Esse Projeto Final tem como objetivo mostrar os benefícios que podem ser obtidos através de uma abordagem de Arquitetura Orientada a Serviços (SOA – *Service Oriented Architecture*) através de um estudo e da *implementação* prática de uma integração de sistemas. O objetivo do trabalho é proporcionar uma integração com um banco de dados SQL Server, utilizando-se de *Web Services* e XML, de forma que o sistema origem não necessite do conhecimento de comandos SQL para realizar consultas no banco, mas apenas o envio de mensagens XML especificadas no *Web Service* exposto.

Palavras-Chave

Integração de Sistemas, Arquitetura Orientada a Serviços, SOA, *Web Services*, XML.

Sumário

Dedicatória ii
Agradecimentos iii
Resumo iv
Palavras-Chave v
Sumário vi
Índice de Tabelas xii
Índice de Figuras xiii
Glossário xiv
1. Introdução 1
2. Arquitetura Orientada a Serviços 3
3. Conceitos 7
3.1. Serviços Web (*Web Services*) 7
3.2. SOAP 8
3.3. WSDL – Web Service Description Language 9
3.4. Namespace 11
3.5. Arquitetura Orientada a Serviços 11
3.5.1. Visão Geral 11
3.5.2. Benefícios 12
3.5.2.1. Reutilização de software 12
3.5.2.2. Aumento de produtividade 13
3.5.3. Requerimentos 13
4. Ferramentas 14
4.1. Aqualogic Service Bus (ALSB) 14
4.2. Weblogic Workshop – Weblogic Integration (WLI) 15
5. Plano Para Gerenciamento de Projeto 16
6. Especificação Funcional 17
6.1. Visão Geral 17

6.1.1. Descrição 17
6.2. Premissas 17
6.3. Modelo de Dados do Banco de Dados 17
6.4. Lista de Casos de Uso 18
6.5. Modelo de Casos de Uso 19
6.6. Descrição dos Casos de Uso 20
6.6.1. UC_SOA_01 – BuscarDadosPessoaPorCpf 20
6.6.1.1. Descrição 20
6.6.1.2. Atores 20
6.6.1.3. Fluxo de Eventos 20
6.6.1.4. Regras de Negócio 21
6.6.1.5. Diagrama de Seqüencia 22
6.6.2. UC_SOA_02 – BuscarContatosPessoaPorCpf 22
6.6.2.1. Descrição 22
6.6.2.2. Atores 22
6.6.2.3. Fluxo de Eventos 22
6.6.2.4. Regras de Negócio 23
6.6.2.5. Diagrama de Seqüencia 24
6.6.3. UC_SOA_03 – BuscarFuncaoPessoaPorCpf 25
6.6.3.1. Descrição 25
6.6.3.2. Atores 25
6.6.3.3. Fluxo de Eventos 25
6.6.4. Regras de Negócio 26
6.6.4.1. Informações Enviadas ao Barramento 26
6.6.4.2. Diagrama de Seqüencia 27
6.6.5. UC_SOA_04 – BuscarContatoPessoasPorAreaConhecimento 27
6.6.5.1. Descrição 27
6.6.5.2. Atores 27
6.6.5.3. Fluxo de Eventos 27

6.6.6. Regras de Negócio 28
6.6.6.1. Informações Enviadas ao Barramento 28
6.6.6.2. Diagrama de Seqüencia 29
6.6.7. UC_SOA_05 – BuscarContatoPessoaPorNomeSobrenome 29
6.6.7.1. Descrição 29
6.6.7.2. Atores 29
6.6.7.3. Fluxo de Eventos 30
6.6.8. Regras de Negócio 30
6.6.8.1. Informações Enviadas ao Barramento 30
6.6.8.2. Diagrama de Seqüencia 31
6.6.9. UC_SOA_06 – BuscarTodasInformacoesPessoaPorCpf 32
6.6.9.1. Descrição 32
6.6.9.2. Atores 32
6.6.9.3. Fluxo de Eventos 32
6.6.10. Regras de Negócio 33
6.6.10.1. Informações Enviadas ao Barramento 33
6.6.10.2. Diagrama de Seqüencia 34
7. Especificação Técnica 35
7.1. Visão Geral 35
7.2. Fluxo Básico dos Casos de Uso 35
7.3. Lista de Interfaces 36
7.4. Descrição dos Serviços no ALSB 37
7.4.1. Premissas 37
7.4.2. Propósito 38
7.4.3. Endereço 38
7.4.4. Particularidades 38
7.4.5. Diagrama do Serviço 39
7.4.6. Descrição dos Estágios 39
7.4.6.1. Receber Requisição 39

7.4.6.2. Armazenar Dados da Requisição 39
7.4.6.3. Operational Branch 39
7.5. Organização do Projeto no ALSB 49
7.6. Descrição dos Processos no WLI 49
7.6.1. Premissas 49
7.6.2. Propósito 50
7.6.3. Endereço 50
7.6.4. Desenho da Solução 50
7.6.4.1. JDBC Control 50
7.6.4.2. Processos no WLI 52
7.7. Organização de Projeto no WLI 54
8. Resultados do Desenvolvimento 55
9. Conclusão 58
Bibliografia 59
Apêndice I – PGPS 60
11.1. Apresentação 60
11.1.1. Sumário do Projeto 60
11.1.1.1. Finalidade, Escopo e Objetivos 60
11.1.1.2. Postulados e Restrições 60
11.1.1.3. Liberações Parciais 60
11.1.1.4. Sumário de Cronograma 60
11.1.2. Evolução do Plano 60
11.2. Organização do Projeto 61
11.2.1. Interfaces Externas 61
11.2.2. Estrutura Interna 61
11.2.3. Papéis e Responsabilidades 61
11.3. Processos de Gerenciamento 61
11.3.1. Partida no Projeto 61
11.3.1.1. Previsões 61

11.3.1.2. Equipe 61
11.3.1.3. Plano de Aquisição de Recursos 61
11.3.1.4. Plano de Treinamento da Equipe 61
11.3.2. Plano de Trabalho 61
11.3.2.1. Atividades 61
11.3.2.2. Prazos 61
11.3.2.3. Alocação de Recursos 62
11.3.2.4. Alocação de Orçamento 62
11.3.3. Planos de Controle 62
11.3.3.1. Controle dos Requisitos 62
11.3.3.2. Controle dos Prazos 62
11.3.3.3. Controle do Orçamento 62
11.3.3.4. Controle de Qualidade 62
11.3.3.5. Plano de Relatórios 62
11.3.3.6. Plano de Medidas 62
11.3.4. Plano de Gerenciamento de Riscos 62
11.3.5. Plano de Encerramento 64
11.4. Processos Técnicos 64
11.4.1. Modelo dos Processos 64
11.4.2. Métodos, Ferramentas e Técnicas 64
11.4.3. Infra-estrutura 64
11.4.4. Plano para a Aceitação do Produto 64
11.5. Planos para os processos de Suporte 64
11.5.1. Gerenciamento de Configuração 64
11.5.2. Plano de Verificação e de Validação 65
11.5.3. Documentação 65
11.5.4. Plano para Assegurar a Qualidade 65
11.5.5. Revisões e Auditorias 65
11.5.6. Plano para a Resolução de Problemas 65

11.5.7. Gerenciamento de Subcontratações 65
11.5.8. Plano de Aperfeiçoamento 65

Índice de Tabelas

Tabela 1 - Tabela de Casos de Uso 18
Tabela 2 - Lista de Interfaces ALSB 37
Tabela 3 - Padrão VETO 38
Tabela 4 - Organização do Projeto no ALSB 49
Tabela 5 - Organização de Projeto no WLI 54
Tabela 6 - Total de Linhas de Código por Objeto 57
Tabela 7 - Total de Horas por Objeto 57
Tabela 8 - Planilha RMMM 63

Índice de Figuras

Figura 1 - Integração Ponto a Ponto 3
Figura 2 - Integração SOA 4
Figura 3 - Curva de Custo de Software 5
Figura 4 - Estrutura da Mensagem SOAP 9
Figura 5 - Estrutura Básica de Comunicação 10
Figura 6 - Estrutura de um Arquivo WSDL 10
Figura 7 - Modelo de Dados 18
Figura 8 - Casos de Uso 19
Figura 9 - Diagrama de Seqüencia UC_SOA_01 22
Figura 10 - Diagrama de Seqüencia UC_SOA_02 24
Figura 11 - Diagrama de Seqüencia UC_SOA_03 27
Figura 12 - Diagrama de Seqüencia UC_SOA_04 29
Figura 13 - Diagrama de Seqüencia UC_SOA_05 31
Figura 14 - Diagrama de Seqüencia UC_SOA_06 34
Figura 15 - Fluxo Básico dos Casos de Uso 36
Figura 16 – Diagrama do Proxy Service ALSB 39
Figura 17 – Comportamento do *Branch Default* 40
Figura 18 - Diagrama Geral do Fluxo das Operações 41
Figura 19 - Visão geral de um Processo no WLI 53
Figura 20 - Interface SOAPUI 55

Glossário

Branch	Ramificação - No ALSB
Business Service	Entidade no ALSB para chamada a um serviço interno ou externo
CORBA	Common Object Request Broker Architecture
Data Source	Entidade no WLI que identifica a fonte de dados
DCOM	Distributed Component Object Model
EJB	Enterprise JavaBeans
Endpoint	Endereço físico de determinado serviço
ESB	Enterprise Service Bus
HTTP	Hypertext Transfer Protocol
IMAP	Internet Message Access Protocol
IP	Internet Protocol
JDBC	Java DataBase Control
JMS	Java Message Service
JPD	Java Process Definitions - Processos no WLI
POP	Post Office Protocol
Proxy Service	Entidade no ALSB para um serviço construído
Query	Consulta ao Banco de Dados por comando
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
Stage	Nome dado aos estágios do ALSB
SVN	SubVersioN - Programa para versionamento
WLI	WebLogic Integration
XML	eXtended Markup Language
Xquery	Transformação entre XMLs
XSD ou Schema	XML Schema Definition

1. Introdução

A interconexão de sistemas computacionais é objeto de pesquisa e *implementação* há alguns anos. A história dessas integrações de sistemas de software é grande e das mais variadas. A computação distribuída gerou a possibilidade de processos e aplicações em uma localidade acessarem funcionalidades (operações e dados) disponíveis em outra localidade (geralmente física). Desde então, houve a necessidade de integrar sistemas computacionais distintos de forma a fazer com que diferentes plataformas pudessem se comunicar e aproveitar os serviços já existentes.

Muitas tecnologias tentaram solucionar os problemas que surgiram com a computação distribuída, dentre elas:

- RPC - Chamada de Procedimento Remoto (*Remote Procedure Call*) - Que é um protocolo para comunicação entre aplicativos. Existem diversos modelos e implementações de RPC;
- CORBA – *Common Object Request Broker Architecture* – Uma especificação para criação, distribuição, gerenciamento e chamada a componentes de software (Objetos C++);
- RMI – Chamada Remota de Método (*Remote Method Invocation*) – Um mecanismo Java que permite aplicações nesta linguagem a invocar métodos num objeto Java em execução numa máquina virtual diferente de forma a esconder as complicações de rede;
- EDI e Outras Tecnologias de troca de Documento – Tecnologias utilizadas para trocas de documentos inteiros de forma orquestrada e coordenada. Troca de documentos é crítica para a troca de informações do negócio.

Um dos problemas com os modelos de computação distribuída existentes é a plataforma utilizada e o vendedor (distribuidor) do software utilizado. Serviços Web (*Web Services*) oferecem a capacidade de realizar operações de computação distribuída com grande interoperabilidade entre plataformas e sistemas. A Arquitetura Orientada a Serviços visa disponibilizar de forma fácil, genérica e eficiente serviços de determinado sistema para quaisquer outros sistemas computacionais, evitando assim uma topologia de interconexão em estrela.

Para exemplificar as vantagens da Arquitetura Orientada a Serviços, será apresentada uma aplicação que recupera dados de um Sistema de Banco de Dados utilizando essa arquitetura. Essa aplicação é um sistema síncrono e com múltiplas operações que podem ser chamadas através de um único endereço no barramento SOA. Cada uma dessas operações terá seu tipo peculiar de mensagem de requisição e retorno e será responsável pelo retorno de diferentes tipos de informações do Sistema de Banco de Dados. Mais do que isso, através dessa aplicação,

será possível que qualquer sistema que seja capaz de se comunicar através do protocolo SOAP sobre HTTP com a aplicação seja integrado ao Sistema de Banco de Dados e receba informações do mesmo sem haver a necessidade de conhecimento de comandos SQL.

No Capítulo 2 será apresentada uma visão geral sobre Arquitetura Orientada a Serviços, que motivou este trabalho. No Capítulo 3 serão abordados alguns dos conceitos fundamentais para a realização e compreensão deste projeto. No Capítulo 4 haverá uma descrição das ferramentas utilizadas ao longo deste projeto. No Capítulo 5 é mostrado o Plano para Gerenciamento de Projeto. No Capítulo 6 mostra-se a Especificação Funcional do Projeto. No Capítulo 7 mostra-se a Especificação Técnica do Projeto. No Capítulo 8 serão exibidos os Resultados do Desenvolvimento. Por fim, no Capítulo 9 haverá a Conclusão.

2. Arquitetura Orientada a Serviços

A integração de sistemas computacionais é utilizada de forma a compartilhar informações entre sistemas e desenvolver novas aplicações através da integração das informações contidas em sistemas já existentes. O desenvolvimento dessas novas aplicações e a integração de sistemas tradicionais não têm sido flexíveis e baseadas em padrões para facilitar e suportar as necessidades de mudanças de certos setores da Tecnologia da Informação.

Em grandes empresas, o desenvolvimento de software significa interagir com dados do negócio da empresa, espalhados por um ou mais sistemas e outros aplicativos. Em outras palavras, desenvolvimento de aplicativos para muitas empresas significa integração de sistemas e integração de sistemas significa desenvolvimento de aplicativos. Em muitos dos casos, a abordagem adotada ainda é uma abordagem de integração ponto a ponto, em que um software novo é desenvolvido a cada necessidade de se consumir dados de um ou mais sistemas da empresa.

Na metodologia de integração ponto a ponto (ou *peer-to-peer*), as aplicações são integradas com outras aplicações conforme a necessidade. As interconexões como mostradas na figura abaixo mostram a falta de compartilhamento de infra-estrutura, assim como a multiplicação de esforços e aumento do custo.

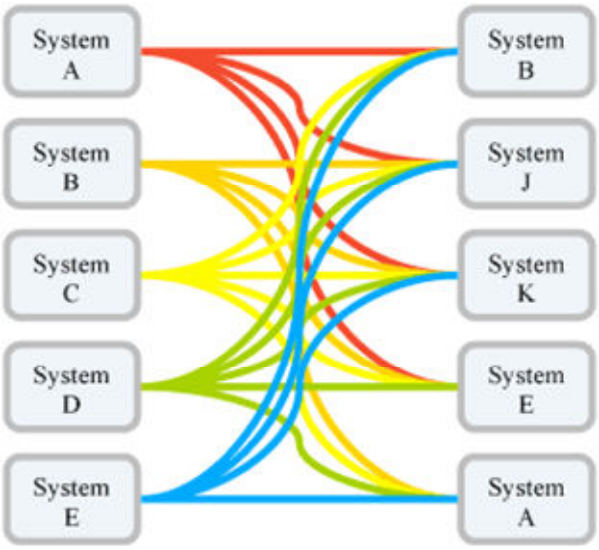

Figura 1 - Integração Ponto a Ponto

A Arquitetura Orientada a Serviços é uma estratégia de Tecnologia da Informação que organiza funções discretas contidas em aplicações de forma a transformá-las em serviços baseados em padrões que podem ser combinados e reutilizados rapidamente de forma a solucionar com maior sucesso determinadas necessidades de um negócio. Serviços são de fácil acoplamento, possuem interfaces bem definidas, independentes de plataforma e são reutilizáveis.

A maior parte dos *middlewares* de comunicação de sistemas, como CORBA, RPC, DCOM, EJB e RMI dependem de padrões similares. Entretanto, existem diferenças entre cada tipo de *implementação* e fraquezas entre elas. O primeiro ponto detectado quando da criação da Arquitetura Orientada a Serviços foi sobre os padrões aceitáveis e a interoperabilidade. SOA foi criada com base nessas tecnologias e tenta eliminar suas aparentes fraquezas. Cada uma dessas tecnologias tem uma granularidade fixa, funções para RPC, objetos para CORBA, e assim por diante. Serviços não possuem essa granularidade fixa. Ao invés disso, serviços podem ser funções, objetos, aplicativos e assim por diante. Sendo assim, SOA é adaptável a qualquer sistema existente e não força o sistema a se limitar a qualquer tipo de granularidade.

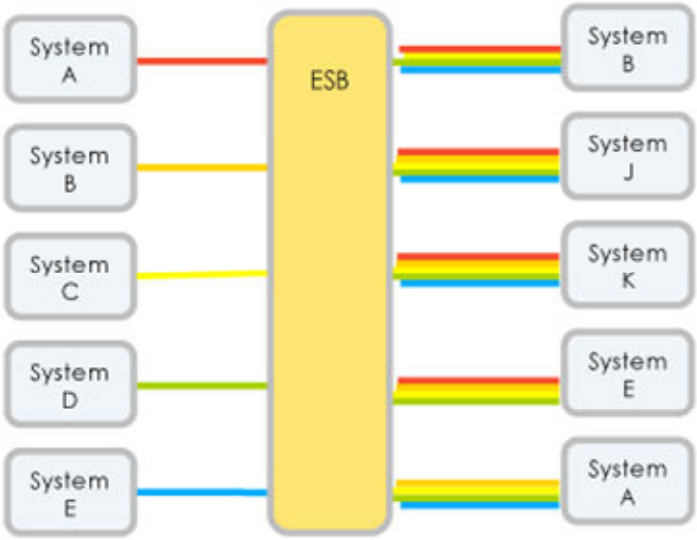

Figura 2 - Integração SOA

A idéia principal da arquitetura SOA é concentrar em um Barramento, comumente chamado de ESB, (da sigla em inglês *Enterprise Service Bus*) todos os serviços que realizem alguma operação em um sistema. Isso garante uma topologia muito melhor do que a topologia gerada numa integração ponto a ponto. Isto pode ser visto na figura acima. Além disso, no caso

particular deste Projeto Final, será utilizada a ferramenta da BEA chamada *Aqualogic Service Bus*, que possibilita a *implementação* deste barramento de serviços idealizado na arquitetura SOA. Mais do que isso, como pode ser visto na figura abaixo, através do barramento de serviços é possível uma reutilização de serviços, de forma que a curva de custo para desenvolvimento de uma aplicação decaia muito com o tempo, economizando várias horas de retrabalho. Pode-se perceber que para as aplicações posteriores a letra D há uma grande reutilização de serviços, permitindo uma grande economia de tempo e maior agilidade.

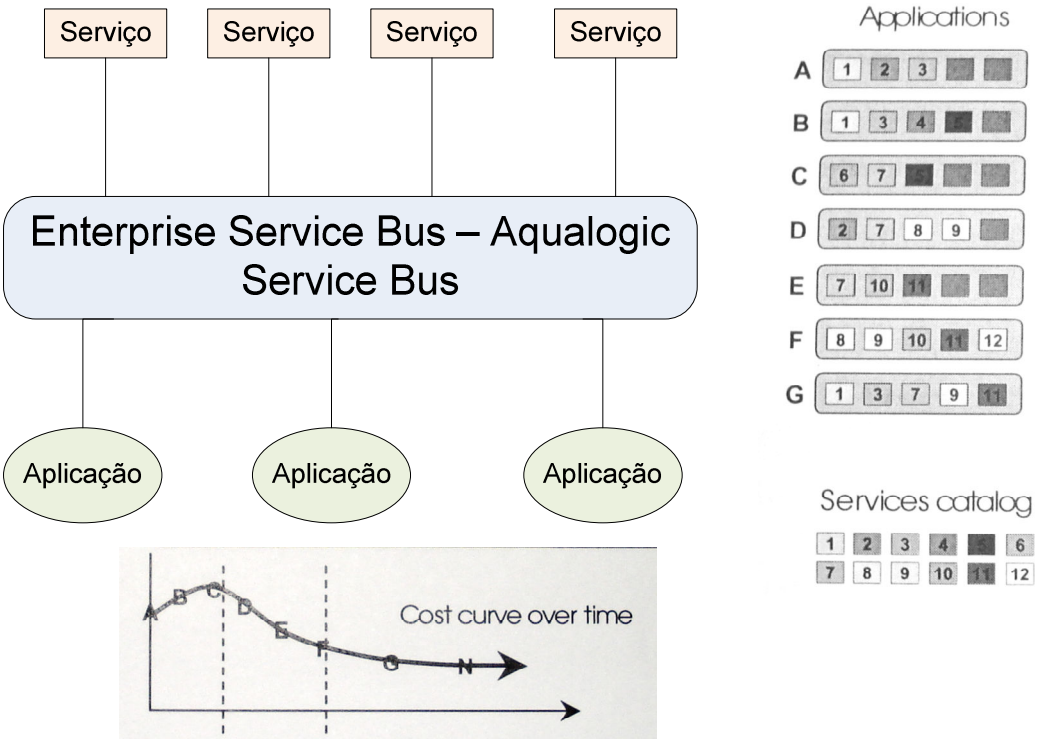

Figura 3 - Curva de Custo de Software

Outro principal objetivo de SOA é permitir que cliente e serviços se comuniquem e se compreendam independentemente da plataforma em que eles são executados. Isto pode ser realizado através de *Web Services* e padrões de comunicação já especificados. Como *Web Services* podem ser orientados a Documentos, podendo receber um XML, que é muito mais flexível do que uma variável numérica ou texto, o serviço disponibilizado pode servir para múltiplos clientes, desde que estes entendam que um metadado trasferido através de um XML equivale aos seus parâmetros internos de sistema.

A padronização dos *Web Services* se torna uma realidade uma vez que já existe uma organização criada com o intuito de aumentar a interoperabilidade entre *Web Services* – a *Web Services Interoperability Organization*. Esta organização já criou um padrão chamado *WS-I Basic Profile*, que define parâmetros a serem seguidos de forma a melhorar a integração de *Web Services*.

Este novo conceito de Arquitetura Orientada a Serviços que permite esta agilidade, assim como uma melhora na qualidade da integração de sistemas, foi o que motivou a criação deste Projeto.

3. Conceitos

3.1. Serviços Web (*Web Services*)

Um serviço web é um software com uma interface XML que pode ser acessado através de um protocolo padrão e aberto. Um serviço web pode enviar e receber mensagens XML em um formato especificado.

Os mecanismos de transporte para um Serviço Web variam, mas, tipicamente, eles operam sobre HTTP, FTP, SMTP ou outro protocolo Internet aberto. Um Serviço Web pode ser descrito de forma padrão, também num documento XML. Por serem baseados em XML, eles propiciam descrições detalhadas de suas interfaces e podem ser descritos de forma abstrata.

A comunicação entre componentes é dada através de XML, padrões abertos e protocolos de comunicação conhecidos. Como resultado, qualquer componente de software numa plataforma que opere com essa construção pode participar de uma operação de Serviço Web. Isso permite a integração entre plataformas (*Web Services* baseados em Java podem ser acessados por clientes baseados em Visual Basic, Perl, .NET), o que aumenta a base de clientes dos serviços. Por não exigir requisitos na especificação dos *Web Services* para o componente que realiza o serviço, é simples estender componentes e adicionar uma interface de Serviço Web.

A capacidade de integração dos *Web Services* é imensa, englobando a possibilidade de estender aplicativos já existentes, desenvolvimento de novos aplicativos, desenvolvimento de aplicativos compostos, integração de sistemas legados e bancos de dados. Em resumo, quaisquer dois sistemas que precisem ser conectados podem ser integrados utilizando-se de Serviços Web.

Esta utopia de integração não se dará da noite para o dia, porém grandes apostas são feitas neste caminho. Antes que todas as promessas dos *Web Services* sejam realizadas, uma grande gama de sistemas ao redor do mundo precisam ser capazes de trabalhar com *Web Services*. Para que isso aconteça, os dados dos sistemas precisam ser transformados para XML ou possuírem uma ponte capaz de expor seus dados através de XML.

Como a interoperabilidade entre plataformas pode ser garantida? Apenas pelo apoio de toda comunidade tecnológica. Toda grande empresa do ramo de tecnologia provê ou espera prover algum tipo de suporte a *Web Services*, como uma forma de promover a interação de seus sistemas. Líderes nessa área são: BEA Systems, IBM, Microsoft e SUN. Essas empresas,

assim como muitas outras, contribuem para Organização de Interoperabilidade de Serviços Web (*Web Services Interoperability Organization* – WS-I: http://www.ws-i.org) que é formada para garantir que os padrões de *Web services* sejam criados, funcionalidades sejam desenvolvidas respeitando sempre o fato de que as especificações dos Serviços Web sejam mantidas abertas e não proprietárias. Os padrões atuais de mensagens baseadas em XML são definidos pelo *World Wide Web Consortium* (W3C – http://www.w3c.org/2002/ws) e o grupo OASIS (http://www.oasis-open.org).

Um *Web Service* é dito síncrono quando durante o processamento de sua operação, ou regra de negócio, o cliente fica bloqueado aguardando resposta. Um *Web Service* síncrono imita uma chamada no estilo RPC.

Um *Web Service* é dito assíncrono quando não há a necessidade do cliente ficar bloqueado aguardando a resposta do processamento da operação. Geralmente uma resposta simples de recebimento (muitas vezes chamada de "*ack*") é enviada para informar o cliente de que a requisição foi recebida com sucesso e será processada para posterior envio da resposta.

3.2. SOAP

O SOAP (*Simple Object Access Protocol*) é um protocolo que descreve o formato da mensagem e a comunicação entre as partes envolvidas em um *Web Service*. O SOAP é um protocolo baseado em XML para executar chamadas estilo RPC e troca de mensagens. As mensagens SOAP são formatadas em XML e são contidas em um envelope SOAP. Mensagens SOAP são destinadas a ficar embutidas em outros protocolos como HTTP, JMS, SMTP e outros. O formato da mensagem XML num envelope SOAP pode variar, mas tipicamente ele é definido de forma a conter as informações necessárias para a chamada de um método ou conter as informações de roteamento necessárias para a entrega do documento.

O SOAP não é um protocolo de comunicação, é apenas um padrão de formato de mensagem. Outro protocolo de transporte, como HTTP/HTTPS ou SMTP é necessário para a entrega da mensagem. SOAP é a estrutura de mensagem que predomina em quase todos os *Web Services*. Em particular, requisições e respostas SOAP são o âmago da comunicação entre cliente e provedor de *Web Services*.

A mensagem SOAP consiste de um cabeçalho (*header*) com informações, um envelope para conter os dados de requisição ou resposta, assim como a operação em particular que está sendo invocada. A especificação por si só define as regras para a estrutura da mensagem e como os dados devem ser codificados dentro da mensagem SOAP.

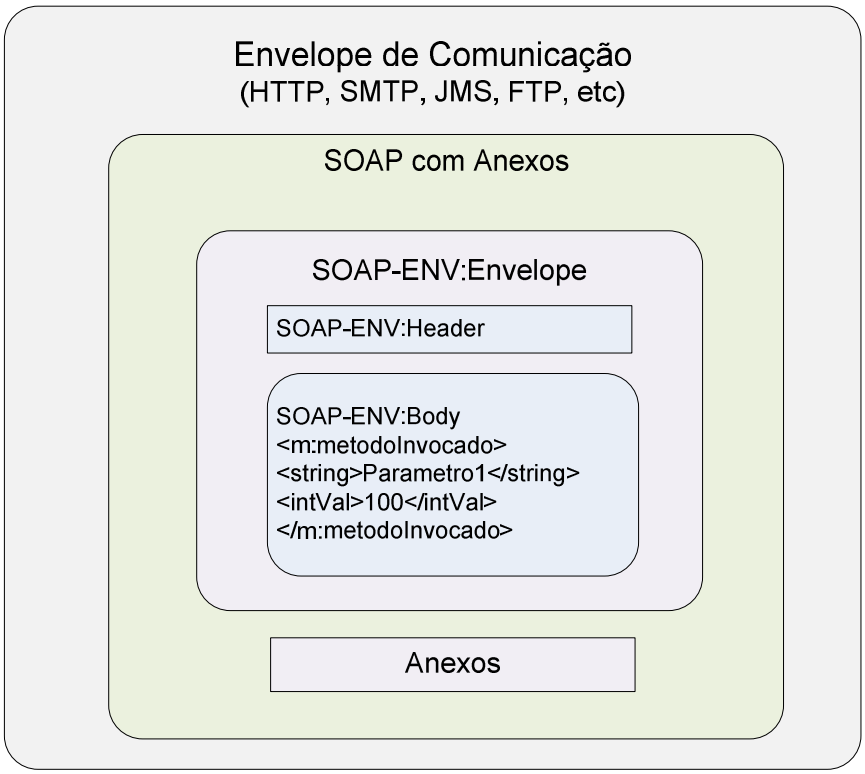

Figura 4 - Estrutura da Mensagem SOAP

3.3. WSDL – Web Service Description Language

WSDL é um formato baseado em XML para descrever *Web Services*. Os elementos declarados num WSDL tornam possível descrever cada operação de um *Web Service*, seus parâmetros, dados de envio e retorno esperados, e dados para invocar a operação, incluindo o endereço de rede do serviço. Essa informação é utilizada para gerar ou formatar a requisição e resposta SOAP apropriadas à interação cliente-provedor de serviço.

Para cada *Web Service*, o provedor do serviço deve criar um arquivo WSDL que o descreva. O arquivo WSDL é utilizado para o cliente entender a sintaxe de comunicação entre ele e o *Web Service*. Um WSDL descreve principalmente: endereço (*endpoint)*, localização, protocolo, operações e tipos de dados para requisição e resposta do serviço.

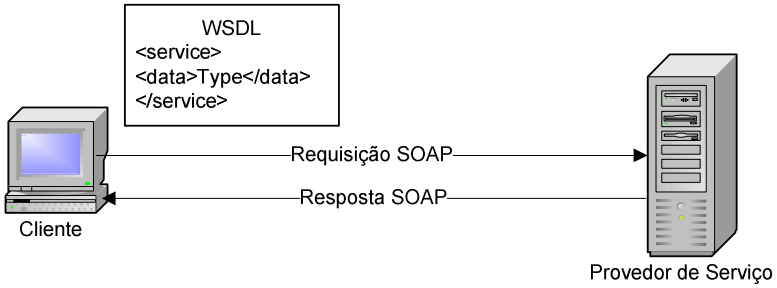

Figura 5 - Estrutura Básica de Comunicação

As informações contidas num WSDL permitem descrever:

- Onde o serviço está localizado;
- Que operações o *Web Service* provê;
- Como invocar cada operação;
- Qual estrutura esperada para requisição e resposta de uma dada operação;

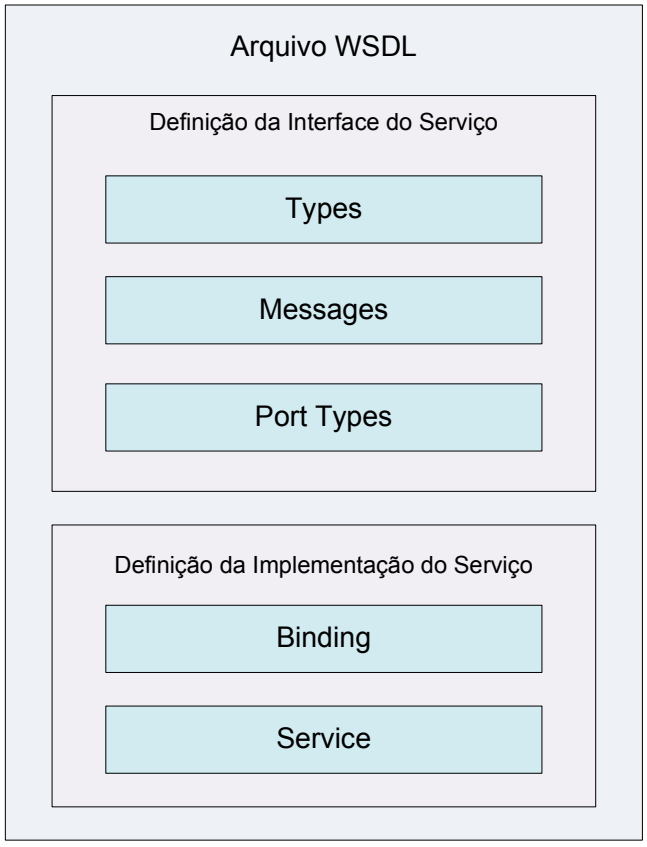

Figura 6 - Estrutura de um Arquivo WSDL

A sintaxe de um WSDL define algumas *tags* para descrever um *Web Service*. Dentre elas:

- <types> - Descreve os diferentes tipos de dados necessários para usar um determinado *Web Service*;
- <message> - Descreve a sintaxe SOAP para os parâmetros de requisição e resposta;
- <portType> - Define a associação entre uma operação de um *Web Service* e parâmetros descritos em <message>;
- <binding> - Define a associação entre uma operação, seu <portType> e o mecanismo de transporte apropriado;
- <port> - Define o endereço usado para acessar o serviço;
- <service> - define a lista completa de operações do *Web Service* e descrições textuais usadas para classificação.

3.4. Namespace

As definições de *namespace* são um mecanismo para assegurar que elementos XML sejam definidos de forma única para evitar conflito de nomes em interações entre sistemas. O conceito de *namespaces* é comum entre desenvolvedores, principalmente Java. Com *Web Services* utilizando muitos XML, garantir que elementos de um XML não entrem em conflito com os de outro é uma responsabilidade importante.

Exemplo: xmlns:tns="http://www.w3.org/2001/XMLSchema"

3.5. Arquitetura Orientada a Serviços

3.5.1. Visão Geral

Service Oriented Architecture (SOA) – ou, em português, Arquitetura Orientada a Serviços – é um termo que descreve duas coisas muito diferentes. As duas primeiras palavras expressam uma metodologia para desenvolvimento de *software*. A terceira palavra é um panorama de todos os ativos de *software* de uma empresa, assim como uma planta arquitetônica é uma representação de todas as peças que, juntas, formam uma construção. Portanto, "*service-oriented architecture*" é uma estratégia que proclama a criação de todos os ativos de software de uma empresa via metodologia de programação orientada a serviços.

O que é um serviço?

Serviços são porções - ou componentes - de *software* construídas de tal modo que possam ser facilmente vinculadas a outros componentes de software. A idéia por trás destes serviços é

simples: a tecnologia expressa de forma que o pessoal de negócio possa entender, e não como um aplicativo enigmático.

No centro do conceito de serviços está a idéia de que é possível definir partes dos códigos de *software* em porções significativas o suficiente para serem compartilhadas e reutilizadas em diversas áreas da empresa. Com isso, algumas tarefas passam a ser automatizadas – por exemplo, enviar uma *query* para um *website* de relatório de crédito para descobrir se um cliente se qualifica para um empréstimo. Se os programadores em um banco puderem abstrair todo este código em um nível mais alto (isto é, pegar todo o código que foi escrito para realizar a verificação de classificação de crédito e reuni-lo em uma única unidade chamada "obter classificação de crédito"), eles poderão reutilizar esta porção da próxima vez que o banco decidir lançar um novo produto de empréstimo que requeira a mesma informação, ao invés de ter que escrever o código a partir do zero.

Para chegar a isto, os desenvolvedores criam um invólucro complexo em torno do código empacotado. Este invólucro é uma interface que descreve o que a porção faz e como conectar a ele. É um conceito antigo, que data dos anos 80, quando a programação orientada a objetos surgiu. A única diferença é a demanda atual por objetos de software muito maiores e mais sofisticados.

3.5.2. Benefícios

3.5.2.1. Reutilização de software

Se o pacote de códigos que constitui um serviço tiver o tamanho e o escopo certos (um grande "se", dizem os veteranos em SOA), então ele poderá ser reutilizado da próxima vez que a equipe de desenvolvimento precisar de uma função específica para um novo aplicativo que queira desenvolver. Tome-se como exemplo uma empresa de telecomunicações que tenha quatro divisões diferentes, cada qual com seu próprio sistema para processar pedidos. Todos estes sistemas executam determinadas funções similares, como verificações de crédito e buscas de registros de clientes. Mas, tendo em vista que cada sistema é altamente integrado, nenhuma destas funções redundantes pode ser compartilhada. O desenvolvimento orientado a serviços coleta o código necessário para criar uma versão de "verificação de crédito" que possa ser compartilhada pelos quatro sistemas. O serviço pode ser uma porção de software totalmente nova ou um aplicativo composto, consistindo de código de alguns dos sistemas ou de todos eles. De qualquer forma, o aplicativo composto é envolto por uma interface que oculta sua complexidade. Da próxima vez que os desenvolvedores quiserem criar um aplicativo que exija verificação de crédito, vão criar um link simples para o novo aplicativo. Eles não precisam se preocupar em conectar aos sistemas individuais — na realidade, nem precisam saber como o código foi incluído ou de onde ele vem. Só precisam criar uma conexão para ele.

3.5.2.2. **Aumento de produtividade**

Se os desenvolvedores reutilizam serviços, os projetos de *software* podem andar mais rápido e a mesma equipe de desenvolvimento pode trabalhar em mais projetos. A integração se torna mais barata e mais rápida, eliminando alguns meses dos ciclos de desenvolvimento de novos projetos.

3.5.3. **Requerimentos**

A Arquitetura Orientada a Serviços é uma abstração da coleção das especificações de *Web Services*. A Arquitetura Orientada a Serviços requer dois padrões para invocação e descrição de serviços, respectivamente. No caso de *Web Services*, o padrão mais comum para invocar e interagir é o protocolo SOAP (abordado anteriormente), que descreve como uma aplicação se comunica com outras aplicações. O padrão mais comum para descrição de *Web Services* é o WSDL (também abordado anteriormente), que provê um mecanismo capaz de gerar metadados ou informação descritiva sobre o serviço que trata.

4. Ferramentas

4.1. Aqualogic Service Bus (ALSB)

Para tornar a Arquitetura Orientada a Serviços uma realidade, os profissionais de TI necessitam uma infra-estrutura inteligente para os serviços, que organize e simplifique a reutilização de serviços e que ainda seja confiável, heterogênea e multiplataforma. Para isto este projeto se utilizará do *Aqualogic Service Bus*, que permite a criação, monitoramento e administração de serviços web num único produto, de forma a permitir a *implementação* e a disponibilização da sua Arquitetura Orientada a Serviços. O ALSB possibilita que não ocorra o espalhamento desordenado de serviços e um maior controle sobre os mesmos, sem perder a flexibilidade e autonomia de cada um.

O ALSB é baseado num servidor WEB (*Weblogic Server*) e permite a criação de *Web Services* sendo o elemento central dos serviços distribuídos. O ALSB permite que seja feito o acoplamento dos clientes de serviços num único ponto de fácil controle e monitoramento.

O ALSB é um intermediário que recebe mensagens, as processa e determina para onde deve roteá-las, além de transformá-las, conforme seja especificado. Ele recebe mensagens através dos protocolos HTTP(S), JMS, *File*, FTP e E-mail (SMTP, POP, IMAP), além de poder enviar suas mensagens através dos mesmos protocolos. O ALSB permite o controle e gerenciamento de diversos *endpoints* distintos dos serviços distribuídos.

Mais do que isso, ele ainda permite que serviços sejam alterados, sem parar os serviços que já estão em produção e seu uso. Isto é possível através de um banco de dados do servidor Web que guarda a última versão da ativação do serviço disponibilizado, mantendo seu uso de forma normal e alterando o serviço apenas quando houver uma nova ativação. Não havendo mudanças na forma de comunicação ou na mensagem esperada pelo *proxy* do serviço, a alteração será imperceptível para o cliente em termos de indisponibilidade de serviço. Não somente os outros serviços que não estão sofrendo alteração, como o próprio serviço sendo alterado estarão disponíveis por todo o tempo de alteração do serviço na última versão presente no servidor. Isso se torna muito útil para sistemas que necessitem de uma disponibilidade constante e não contem com janelas para realizar alterações de forma a indisponibilizar o serviço para os clientes.

4.2. Weblogic Workshop – Weblogic Integration (WLI)

O *Weblogic Integration* é um framework baseado no Eclipse que permite a geração de *Web Services* e *Business Services*, além de todas as possibilidades de controles Java e transformações (*Xqueries*). Através dele serão gerados todos os códigos que farão a lógica de comunicação com o Sistema de Banco de Dados (SQL Server 2005 Express) através de um JDBC Control.

O WLI se caracteriza por ser uma ferramenta de codificação de Quarta Geração, permitindo a inserção de diversos blocos já desenvolvidos e chamados de controles de forma a facilitar a programação. Através desses controles há uma abstração de várias funções que necessitam ser *implementadas* em outras linguagens como C, C++ e Java. Essa abstração é traduzida para código Java, que pode ser visto e alterado através da aba de código fonte. Os controles existentes são bastante simplificados e na maioria das vezes necessitam de inserção de lógica própria em código Java, o que será realizado para a construção do JDBC *Control* e toda a lógica dos *Web Services* que serão disponibilizados.

Ainda no WLI, existe uma ferramenta chamada *Xquery Mapper* que permite o mapeamento de XML *Schemas* de forma a transformar dados de um sistema para outro de forma bastante intuitiva. Mais uma vez, além do modo gráfico há a possibilidade de inserção de código *hardcoded* de forma a realizar mais do que as operações oferecidas.

Por ser baseado em Eclipse, o WLI aceita uma boa personalização, além de possuir *plugins* como o do SVN de forma a facilitar o controle de versão dos softwares gerados. Além disso, o WLI possui um modo gráfico que permite a utilização de métodos e controles como objetos em diversos serviços com uma abordagem de *Business Process Management*.

5. Plano Para Gerenciamento de Projeto

O Plano para Gerenciamento de Projeto de Software (PGPS) busca prever o que seria desenvolvido na cadeira de Projeto Final de forma a orientar melhor a produção do projeto. É também a oportunidade de mostrar em um primeiro momento o que se pretende desenvolver, com quais ferramentas, sob quais critérios e em que prazo de tempo.

O processo criativo precisa ser gerenciado de forma a realizar com a maior eficiência possível o desenvolvimento de um sistema de informação conciso e funcional. O intuito deste Plano foi obter um guia para um melhor desenvolvimento, diante das boas práticas de Engenharia de Software, entregando um produto final de qualidade e dentro do prazo estipulado para o mesmo.

Este Plano para Gerenciamento de Projeto de Software foi elaborado numa fase que precedeu a escrita desta monografia e por isso encontra-se ao fim deste documento como Apêndice I.

6. Especificação Funcional

6.1. Visão Geral

6.1.1. Descrição

Este projeto está direcionado à disponibilização de consultas a pessoas por CPF, nome e sobrenome ou área de Conhecimento. Os dados das pessoas estão em um Banco de Dados SQL Server 2005 Express. Para acesso aos dados será utilizada uma Arquitetura Orientada a Serviços. Desta forma, será demonstrado como qualquer sistema capaz de importar um WSDL do Barramento de Serviços poderá realizar consultas sem a necessidade do conhecimento de comandos SQL específicos.

6.2. Premissas

- A partir do Banco de Dados gerado, foram elaborados os casos de uso para cada tipo de consulta a ser realizada no mesmo, de forma que seja inteligível para cada operação o que será consultado;
- As consultas realizadas no Banco de Dados foram realizadas através de um JDBC Control gerado a partir da necessidade do modelo de dados;
- A Comunicação com o Barramento SOA é através de SOAP/HTTP, sendo que o Barramento fornece os WSDLs e XSDs necessários para cada serviço e regra de negócio;
- Estes Serviços estão disponibilizados no Aqualogic Service Bus, que através de *Proxy Services* e *Business Services* encaminha as solicitações ao sistema de destino;
- Toda a comunicação entre os sistemas é realizada de forma síncrona;
- Em caso de ocorrência de erro técnico o Barramento retornará um erro padrão em formato SOAP Fault. Para caso de erros funcionais, o erro será mapeado para os campos de controle previstos nas mensagens de retorno.

6.3. Modelo de Dados do Banco de Dados

A figura abaixo ilustra o modelo de dados do Banco SQL Server 2005. A partir deste modelo de dados foram levantados todos os casos de uso necessários para este projeto.

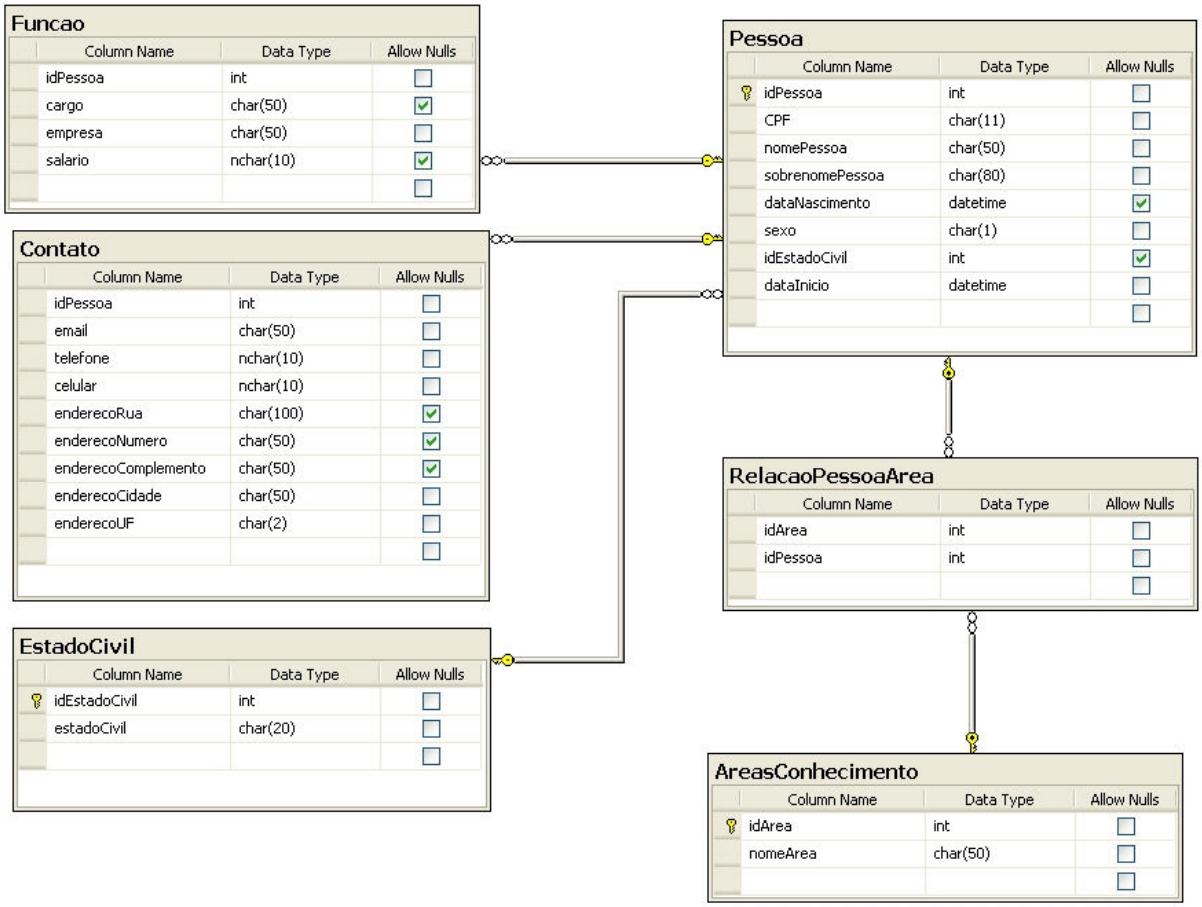

Figura 7 - Modelo de Dados

6.4. Lista de Casos de Uso

Tabela 1 - Tabela de Casos de Uso

Código	Nome
UC_SOA_01	BuscarDadosPessoaPorCpf
UC_SOA_02	BuscarContatosPessoaPorCpf
UC_SOA_03	BuscarFuncaoPessoaPorCpf
UC_SOA_04	BuscarContatoPessoasPorAreaConhecimento
UC_SOA_05	BuscarContatoPessoaPorNomeSobrenome
UC_SOA_06	BuscarTodasInformacoesPessoaPorCpf

6.5. Modelo de Casos de Uso

Na figura abaixo são ilustrados os casos de uso levantados a partir da modelagem dos dados do Banco de Dados.

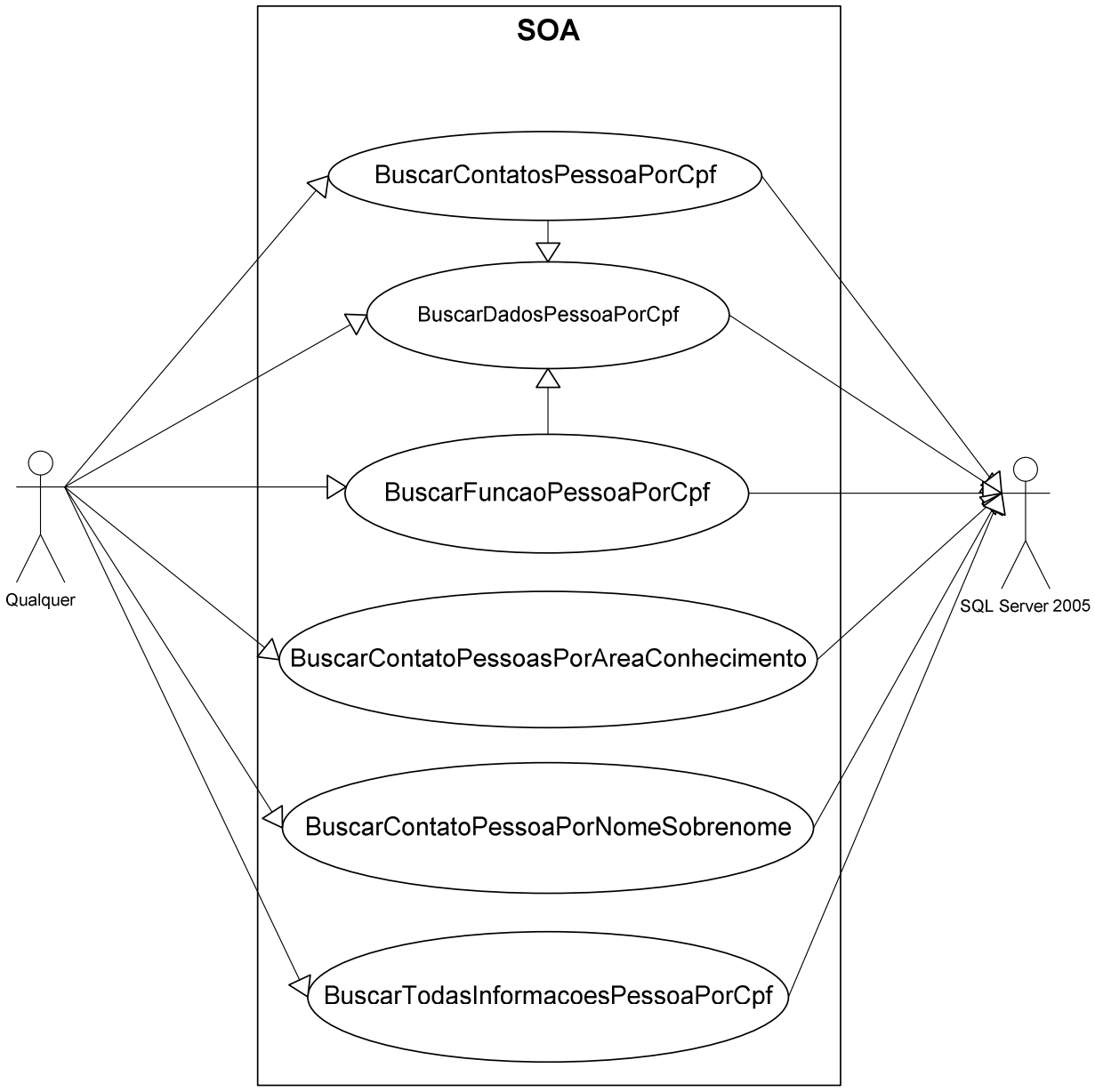

Figura 8 - Casos de Uso

6.6. Descrição dos Casos de Uso

6.6.1. UC_SOA_01 – BuscarDadosPessoaPorCpf

6.6.1.1. Descrição

Este caso de uso é responsável por permitir a visualização de informações sobre determinada pessoa a partir de seu CPF, isto é, receber suas informações como o id da pessoa no banco, seu nome, sobrenome, data de nascimento, sexo e estado civil.

6.6.1.2. Atores

- Origem Qualquer
- SQL Server 2005

6.6.1.3. Fluxo de Eventos

6.6.1.3.1. *Fluxo Básico*

- O caso de uso se inicia quando o sistema consumidor deseja obter informações sobre determinada pessoa;
- O Barramento SOA recebe as informações necessárias;
- O Barramento SOA solicita ao SQL Server 2005 as informações de determinada pessoa;
- O SQL Server 2005 retorna as informações da pessoa solicitada;
- O Barramento SOA retorna as informações ao sistema consumidor;
- O caso de uso é encerrado.

6.6.1.3.2. *Fluxo Alternativo*

Falha no envio de solicitação ao SQL Server 2005

Caso ocorra erro durante a chamada ao SQL Server 2005 o Barramento retornará um código de erro. O Caso de uso é encerrado.

Erro Interno no SQL Server 2005

Caso ocorra erro interno no SQL Server 2005, o Barramento retornará um código de erro. O caso de uso é encerrado.

6.6.1.3.3. *Fluxo de Exceção*

Informações necessárias para consulta não foram fornecidas

O Barramento retorna código de erro e descrição para o sistema cliente. O caso de uso é encerrado.

Erro Interno no Barramento

Caso ocorra erro interno, o Barramento retorna código de erro e descrição para o sistema cliente. O caso de uso é encerrado.

6.6.1.4. Regras de Negócio

6.6.1.4.1. *Informações Enviadas ao Barramento*

Devem ser enviadas as seguintes informações ao Barramento para que seja possível a consulta aos dados de uma pessoa:

- CPF

6.6.1.4.2. *Informações Enviadas ao SQL Server 2005*

Devem ser enviadas as seguintes informações ao SQL Server 2005 para que seja possível a consulta de contatos de uma pessoa:

- CPF

6.6.1.4.3. *Informações Retornadas pelo SQL Server 2005*

As informações retornadas pelo SQL Server 2005 são:

- idPessoa
- CPF
- nomePessoa
- sobrenomePessoa
- dataNascimento
- sexo
- idEstadoCivil

6.6.1.4.4. *Informações Retornadas pelo Barramento*

O Barramento retornará as seguintes informações estruturadas em um arquivo XML: idPessoa, CPF, nomePessoa, sobrenomePessoa, dataNascimento, sexo, estadoCivil

6.6.1.4.5. *Mensagem de erro quando campo obrigatório não é fornecido*

O sistema retorna código de erro, além da mensagem “Mensagem XML de Entrada Inválida. Verifique obrigatoriedade de campos e tipagem de dados.”

6.6.1.4.6. *Mensagem de erro técnico ao consultar SQL Server 2005*

O sistema retorna código de erro, além da mensagem “Erro ao consultar SQL Server 2005:” + mensagem interna do SQL Server 2005.

6.6.1.5. Diagrama de Seqüencia

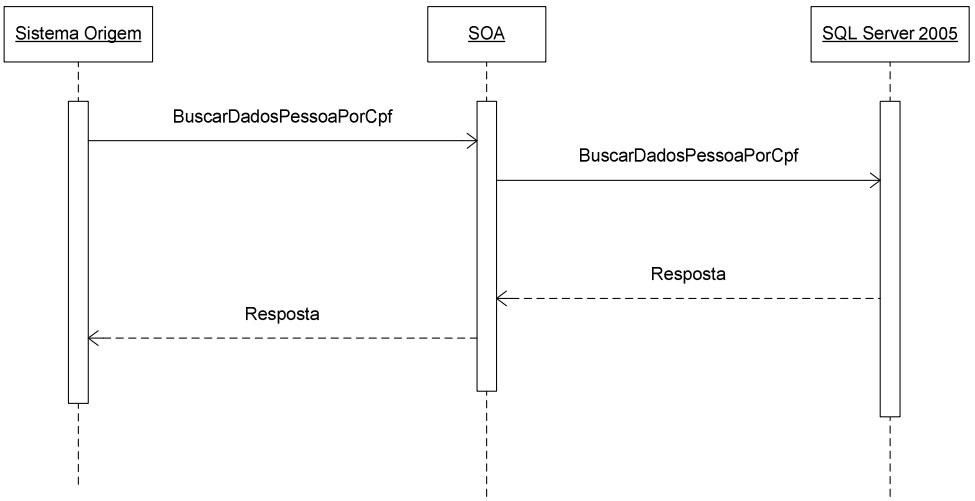

Figura 9 - Diagrama de Seqüencia UC_SOA_01

6.6.2. UC_SOA_02 – BuscarContatosPessoaPorCpf

6.6.2.1. Descrição

Este caso de uso é responsável por permitir a visualização de informações sobre as formas de contato com determinada pessoa a partir de seu CPF, isto é, seu e-mail, telefone, celulare e endereço.

6.6.2.2. Atores

- Origem Qualquer
- SQL Server 2005

6.6.2.3. Fluxo de Eventos

6.6.2.3.1. *Fluxo Básico*

- O caso de uso se inicia quando o sistema consumidor deseja obter informações sobre as formas de contato de uma determinada pessoa a partir de seu CPF;
- O Barramento SOA recebe as informações necessárias;
- O Barramento SOA solicita ao SQL Server 2005 as informações de contato de determinada pessoa;
- O SQL Server 2005 retorna as informações de contato da pessoa solicitada;

- O Barramento SOA retorna as informações ao sistema consumidor;
- O caso de uso é encerrado.

6.6.2.3.2. *Fluxo Alternativo*

Falha no envio de solicitação ao SQL Server 2005

Caso ocorra erro durante a chamada ao SQL Server 2005 o Barramento retornará um código de erro. O Caso de uso é encerrado.

Erro Interno no SQL Server 2005

Caso ocorra erro interno no SQL Server 2005, o Barramento retornará um código de erro. O caso de uso é encerrado.

6.6.2.3.3. *Fluxo de Exceção*

Informações necessárias para consulta não foram fornecidas

O Barramento retorna código de erro e descrição para o sistema cliente. O caso de uso é encerrado.

Erro Interno no Barramento

Caso ocorra erro interno, o Barramento retorna código de erro e descrição para o sistema cliente. O caso de uso é encerrado.

6.6.2.4. Regras de Negócio

6.6.2.4.1. *Informações Enviadas ao Barramento*

Devem ser enviadas as seguintes informações ao Barramento para que seja possível a consulta de contatos de uma pessoa:

- CPF

6.6.2.4.2. *Informações Enviadas ao SQL Server 2005*

Devem ser enviadas as seguintes informações ao SQL Server 2005 para que seja possível a consulta de contatos de um grupo de pessoas:

- CPF

6.6.2.4.3. *Informações Retornadas pelo SQL Server 2005*

- Nome
- Sobrenome

- Email
- Telefone
- Celular
- Endereço

6.6.2.4.4. *Informações Retornadas pelo Barramento*

- Nome
- Sobrenome
- Email
- Telefone
- Celular
- Endereço

6.6.2.4.5. *Mensagem de erro quando campo obrigatório não é fornecido*

O sistema retorna código de erro, além da mensagem "Mensagem XML de Entrada Inválida. Verifique obrigatoriedade de campos e tipagem de dados."

6.6.2.4.6. *Mensagem de erro técnico ao consultar SQL Server 2005*

O sistema retorna código de erro, além da mensagem "Erro ao consultar SQL Server 2005:" + mensagem interna do SQL Server 2005.

6.6.2.5. Diagrama de Seqüencia

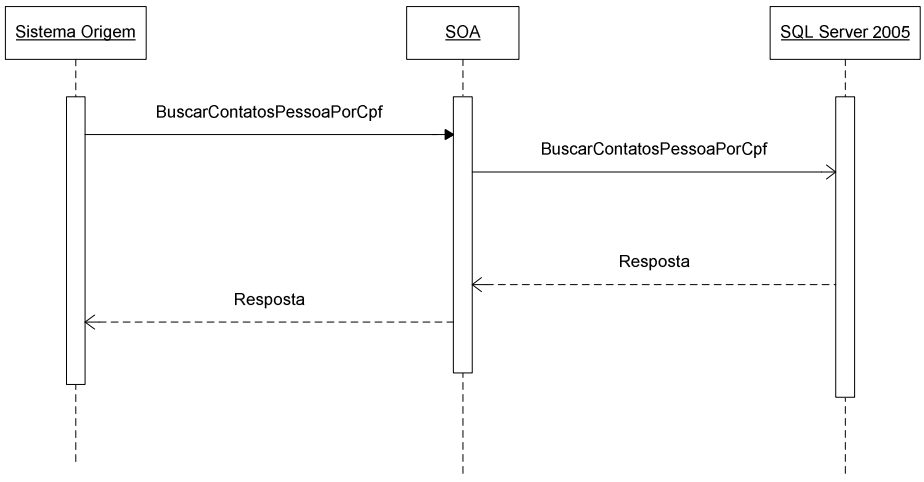

Figura 10 - Diagrama de Seqüencia UC_SOA_02

6.6.3. UC_SOA_03 – BuscarFuncaoPessoaPorCpf

6.6.3.1. Descrição

Este caso de uso é responsável por permitir a visualização de informações sobre a função de determinada pessoa, isto é, seus dados a respeito de área de atuação, cargo, função, empresa e salário.

6.6.3.2. Atores

- Origem Qualquer
- SQL Server 2005

6.6.3.3. Fluxo de Eventos

6.6.3.3.1. *Fluxo Básico*

- O caso de uso se inicia quando o sistema consumidor deseja obter informações sobre a função de determinada pessoa;
- O Barramento SOA recebe as informações necessárias;
- O Barramento SOA solicita ao SQL Server 2005 as informações da função de determinada pessoa;
- O SQL Server 2005 retorna as informações da pessoa solicitada;
- O Barramento SOA retorna as informações ao sistema consumidor;
- O caso de uso é encerrado.

6.6.3.3.2. *Fluxo Alternativo*

Falha no envio de solicitação ao SQL Server 2005

Caso ocorra erro durante a chamada ao SQL Server 2005 o Barramento retornará um código de erro. O Caso de uso é encerrado.

Erro Interno no SQL Server 2005

Caso ocorra erro interno no SQL Server 2005, o Barramento retornará um código de erro. O caso de uso é encerrado.

6.6.3.3.3. *Fluxo de Exceção*

Informações necessárias para consulta não foram fornecidas

O Barramento retorna código de erro e descrição para o sistema cliente. O caso de uso é encerrado.

Erro Interno no Barramento

Caso ocorra erro interno, o Barramento retorna código de erro e descrição para o sistema cliente. O caso de uso é encerrado.

6.6.4. Regras de Negócio

6.6.4.1. Informações Enviadas ao Barramento

Devem ser enviadas as seguintes informações ao Barramento para que seja possível a consulta de informações da função de uma pessoa:

- CPF

6.6.4.1.1. *Informações Enviadas ao SQL Server 2005*

Devem ser enviadas as seguintes informações ao SQL Server 2005 para que seja possível a consulta de função de determinada pessoa:

- CPF

6.6.4.1.2. *Informações Retornadas pelo SQL Server 2005*

As informações retornadas pelo SQL Server 2005 são: nome, sobrenome, função, cargo, área de atuação, empresa, salário e data de início na empresa.

6.6.4.1.3. *Informações Retornadas pelo Barramento*

As informações retornadas pelo Barramento são: nome, sobrenome, função, cargo, área de atuação, empresa, salário e data de início na empresa, formatadas num arquivo XML.

6.6.4.1.4. *Mensagem de erro quando campo obrigatório não é fornecido*

O sistema retorna código de erro, além da mensagem “Mensagem XML de Entrada Inválida. Verifique obrigatoriedade de campos e tipagem de dados.”

6.6.4.1.5. *Mensagem de erro técnico ao consultar SQL Server 2005*

O sistema retorna código de erro, além da mensagem “Erro ao consultar SQL Server 2005:” + mensagem interna do SQL Server 2005.

6.6.4.2. Diagrama de Seqüencia

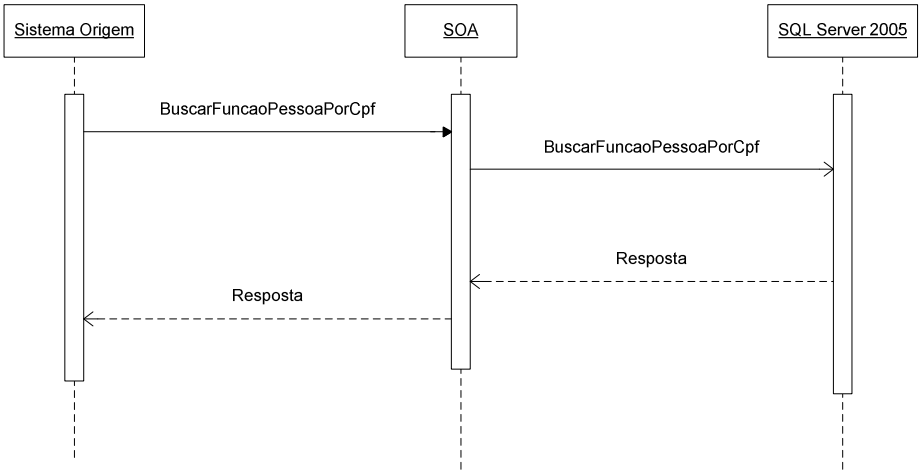

Figura 11 - Diagrama de Seqüencia UC_SOA_03

6.6.5. UC_SOA_04 – BuscarContatoPessoasPorAreaConhecimento

6.6.5.1. Descrição

Este caso de uso é responsável por permitir a visualização de informações de contato de determinado grupo de pessoas baseado em sua Área de Conhecimento, isto é, todos os dados de contato de todas as pessoas pertencentes a uma determinada área de conhecimento.

6.6.5.2. Atores

- Origem Qualquer
- SQL Server 2005

6.6.5.3. Fluxo de Eventos

6.6.5.3.1. *Fluxo Básico*

- O caso de uso se inicia quando o sistema consumidor deseja obter informações para contato de determinado grupo de pessoas pertencente a uma área de conhecimento;
- O Barramento SOA recebe as informações necessárias;
- O Barramento SOA solicita ao SQL Server 2005 as informações de contato de determinado grupo de pessoas de uma área de conhecimento;
- O SQL Server 2005 retorna as informações de contato do grupo de pessoas solicitado;
- O Barramento SOA retorna as informações ao sistema consumidor;
- O caso de uso é encerrado.

6.6.5.3.2. *Fluxo Alternativo*

Falha no envio de solicitação ao SQL Server 2005

Caso ocorra erro durante a chamada ao SQL Server 2005 o Barramento retornará um código de erro. O Caso de uso é encerrado.

Erro Interno no SQL Server 2005

Caso ocorra erro interno no SQL Server 2005, o Barramento retornará um código de erro. O caso de uso é encerrado.

6.6.5.3.3. *Fluxo de Exceção*

Informações necessárias para consulta não foram fornecidas

O Barramento retorna código de erro e descrição para o sistema cliente. O caso de uso é encerrado.

Erro Interno no Barramento

Caso ocorra erro interno, o Barramento retorna código de erro e descrição para o sistema cliente. O caso de uso é encerrado.

6.6.6. Regras de Negócio

6.6.6.1. Informações Enviadas ao Barramento

Devem ser enviadas as seguintes informações ao Barramento para que seja possível a consulta de informações da função de uma pessoa:

- Área de Atuação

6.6.6.1.1. *Informações Enviadas ao SQL Server 2005*

Devem ser enviadas as seguintes informações ao SQL Server 2005 para que seja possível a consulta de função de determinada pessoa:

- areaAtuacao

6.6.6.1.2. *Informações Retornadas pelo SQL Server 2005*

As informações retornadas pelo SQL Server 2005 são: nome, sobrenome, telefone, celular, endereço de cada pessoa pertencente a uma determinada área de atuação.

6.6.6.1.3. *Informações Retornadas pelo Barramento*

As informações retornadas pelo Barramento são: nome, sobrenome, telefone, celular, endereço de cada pessoa pertencente a uma determinada área de atuação, formatadas num arquivo XML.

6.6.6.1.4. *Mensagem de erro quando campo obrigatório não é fornecido*

O sistema retorna código de erro, além da mensagem “Mensagem XML de Entrada Inválida. Verifique obrigatoriedade de campos e tipagem de dados.”

6.6.6.1.5. *Mensagem de erro técnico ao consultar SQL Server 2005*

O sistema retorna código de erro, além da mensagem “Erro ao consultar SQL Server 2005:” + mensagem interna do SQL Server 2005.

6.6.6.2. Diagrama de Seqüencia

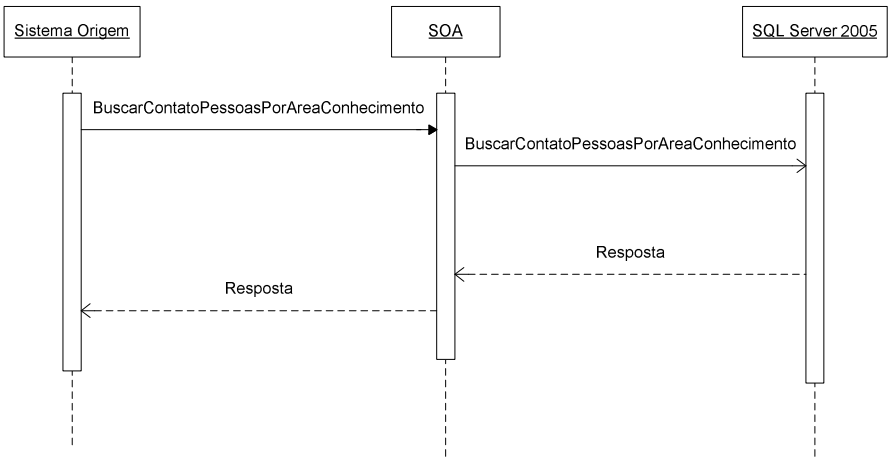

Figura 12 - Diagrama de Seqüencia UC_SOA_04

6.6.7. UC_SOA_05 – BuscarContatoPessoaPorNomeSobrenome

6.6.7.1. Descrição

Este caso de uso é responsável por permitir a visualização de informações de contato de uma determinada pessoa baseado em seu nome e sobrenome, isto é, todos os dados de contato de uma pessoa a partir de seu nome e sobrenome.

6.6.7.2. Atores

- Origem Qualquer
- SQL Server 2005

6.6.7.3. Fluxo de Eventos

6.6.7.3.1. *Fluxo Básico*

- O caso de uso se inicia quando o sistema consumidor deseja obter informações para contato de determinada pessoa a partir de seu nome e sobrenome;
- O Barramento SOA recebe as informações necessárias;
- O Barramento SOA solicita ao SQL Server 2005 as informações de contato de uma pessoa baseado em seu nome e sobrenome;
- O SQL Server 2005 retorna as informações de contato da pessoa solicitada;
- O Barramento SOA retorna as informações ao sistema consumidor;
- O caso de uso é encerrado.

6.6.7.3.2. *Fluxo Alternativo*

Falha no envio de solicitação ao SQL Server 2005

Caso ocorra erro durante a chamada ao SQL Server 2005 o Barramento retornará um código de erro. O Caso de uso é encerrado.

Erro Interno no SQL Server 2005

Caso ocorra erro interno no SQL Server 2005, o Barramento retornará um código de erro. O caso de uso é encerrado.

6.6.7.3.3. *Fluxo de Exceção*

Informações necessárias para consulta não foram fornecidas

O Barramento retorna código de erro e descrição para o sistema cliente. O caso de uso é encerrado.

Erro Interno no Barramento

Caso ocorra erro interno, o Barramento retorna código de erro e descrição para o sistema cliente. O caso de uso é encerrado.

6.6.8. Regras de Negócio

6.6.8.1. Informações Enviadas ao Barramento

Devem ser enviadas as seguintes informações ao Barramento para que seja possível a consulta de informações da função de uma pessoa:

- Nome
- Sobrenome

6.6.8.1.1. *Informações Enviadas ao SQL Server 2005*

Devem ser enviadas as seguintes informações ao SQL Server 2005 para que seja possível a consulta de função de determinada pessoa:

- Nome
- Sobrenome

6.6.8.1.2. *Informações Retornadas pelo SQL Server 2005*

As informações retornadas pelo SQL Server 2005 são: nome, sobrenome, cpf, telefone, celular, endereço da pessoa solicitada.

6.6.8.1.3. *Informações Retornadas pelo Barramento*

As informações retornadas pelo Barramento são: nome, sobrenome, cpf, telefone, celular, endereço da pessoa solicitada, formatadas num arquivo XML.

6.6.8.1.4. *Mensagem de erro quando campo obrigatório não é fornecido*

Sistema retorna código de erro, além da mensagem “Mensagem XML de Entrada Inválida. Verifique obrigatoriedade de campos e tipagem de dados.”

6.6.8.1.5. *Mensagem de erro técnico ao consultar SQL Server 2005*

Sistema retorna código de erro, além da mensagem “Erro ao consultar SQL Server 2005:” + mensagem interna do SQL Server 2005.

6.6.8.2. Diagrama de Seqüencia

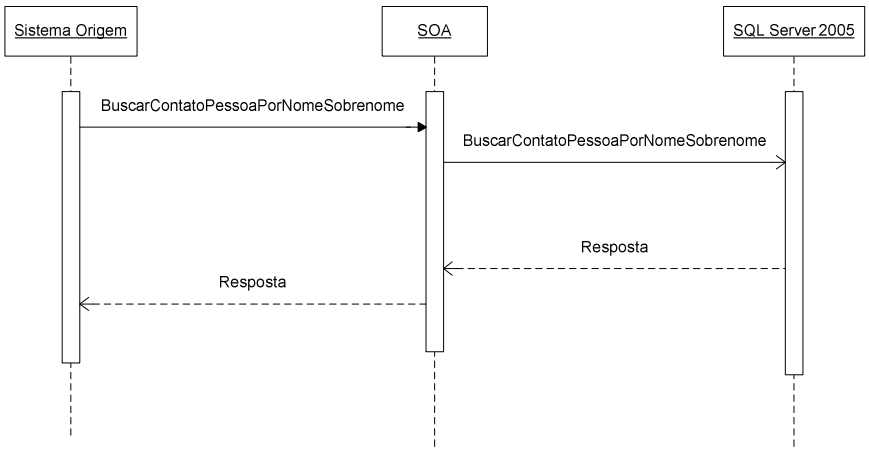

Figura 13 - Diagrama de Seqüencia UC_SOA_05

6.6.9. UC_SOA_06 – BuscarTodasInformacoesPessoaPorCpf

6.6.9.1. Descrição

Este caso de uso é responsável por permitir a visualização de todas as informações cadastradas no banco de uma determinada pessoa baseado em seu CPF, isto é, todos os dados das tabelas do Banco de Dados para o CPF indicado.

6.6.9.2. Atores

- Origem Qualquer
- SQL Server 2005

6.6.9.3. Fluxo de Eventos

6.6.9.3.1. *Fluxo Básico*

- O caso de uso se inicia quando o sistema consumidor deseja obter todas as informações de determinada pessoa a partir de seu CPF;
- O Barramento SOA recebe as informações necessárias;
- O Barramento SOA solicita ao SQL Server 2005 as informações de uma pessoa baseado em seu CPF;
- O SQL Server 2005 retorna as informações da pessoa solicitada;
- O Barramento SOA retorna as informações ao sistema consumidor;
- O caso de uso é encerrado.

6.6.9.3.2. *Fluxo Alternativo*

Falha no envio de solicitação ao SQL Server 2005

Caso ocorra erro durante a chamada ao SQL Server 2005 o Barramento retornará um código de erro. O Caso de uso é encerrado.

Erro Interno no SQL Server 2005

Caso ocorra erro interno no SQL Server 2005, o Barramento retornará um código de erro. O caso de uso é encerrado.

6.6.9.3.3. *Fluxo de Exceção*

Informações necessárias para consulta não foram fornecidas

O Barramento retorna código de erro e descrição para o sistema cliente. O caso de uso é encerrado.

Erro Interno no Barramento

Caso ocorra erro interno, o Barramento retorna código de erro e descrição para o sistema cliente. O caso de uso é encerrado.

6.6.10. Regras de Negócio

6.6.10.1. Informações Enviadas ao Barramento

Devem ser enviadas as seguintes informações ao Barramento para que seja possível a consulta de informações da função de uma pessoa:

- CPF

6.6.10.1.1. *Informações Enviadas ao SQL Server 2005*

Devem ser enviadas as seguintes informações ao SQL Server 2005 para que seja possível a consulta de função de determinada pessoa:

- CPF

6.6.10.1.2. *Informações Retornadas pelo SQL Server 2005*

As informações retornadas pelo SQL Server 2005 são: todos os dados da pessoa solicitada presentes em todas as tabelas.

6.6.10.1.3. *Informações Retornadas pelo Barramento*

As informações retornadas pelo Barramento são: todos os dados de todas as tabelas do Banco de Dados da pessoa solicitada, formatadas num arquivo XML.

6.6.10.1.4. *Mensagem de erro quando campo obrigatório não é fornecido*

O sistema retorna código de erro, além da mensagem “Mensagem XML de Entrada Inválida. Verifique obrigatoriedade de campos e tipagem de dados.”

6.6.10.1.5. *Mensagem de erro técnico ao consultar SQL Server 2005*

O sistema retorna código de erro, além da mensagem “Erro ao consultar SQL Server 2005:” + mensagem interna do SQL Server 2005.

6.6.10.2. **Diagrama de Seqüencia**

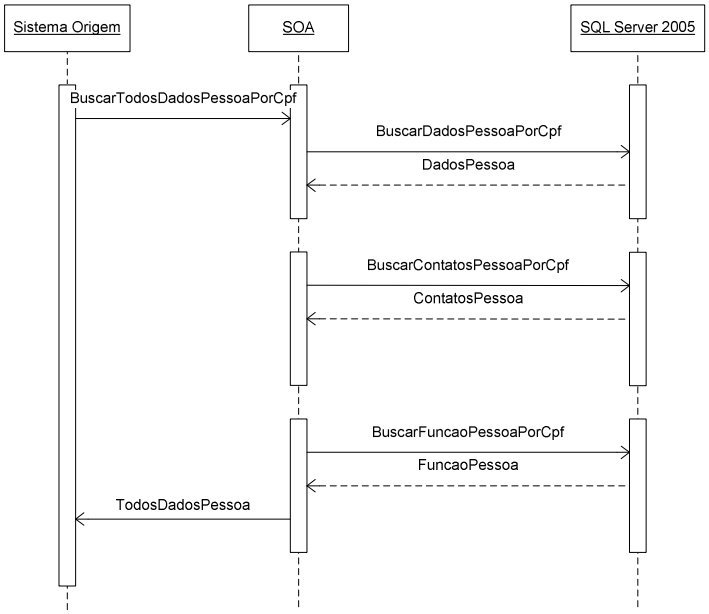

Figura 14 - Diagrama de Seqüencia UC_SOA_06

7. Especificação Técnica

7.1. Visão Geral

As operações possíveis de serem realizadas no Banco de Dados especificado serão disponibilizadas através de um *Proxy Service* no *Aqualogic Service Bus*. Através de um único endereço será possível requisitar qualquer uma das operações previstas no caso de uso através da chamada e mensagem correta.

O ALSB será o responsável pelo armazenamento das requisições, *log* e tratamento de erro das mesmas durante as operações. A partir do reconhecimento da operação solicitada o ALSB chamará o processo JPD disponível no *Weblogic Integration* capaz de realizar aquela solicitação. O acesso ao Banco de Dados e a formatação dos dados obtidos do banco em XML fica a cargo dos processos do *Weblogic Integration*. O *Weblogic Integration* se faz necessário devido às chamadas ao Banco de Dados serem através de um JDBC *Control* que não é realizável no ALSB.

Esta organização é necessária para se obter uma maior facilidade de acesso por qualquer sistema aos serviços oferecidos e manter uma topologia de barramento para este serviço disponibilizado a partir do Banco de Dados MS SQL Server 2005 Express a qualquer sistema que acesse o barramento SOA.

7.2. Fluxo Básico dos Casos de Uso

A figura abaixo ilustra o fluxo básico e genérico de informações para os casos de uso levantados na fase de análise a partir do modelo de dados. Este fluxo ajuda a entender o esboço da solução pensada para o atendimento de todas as operações.

De forma geral, uma requisição chegará ao barramento SOA no *endpoint* especificado do *Aqualogic Service Bus*. O ALSB irá armazenar, validar e formatar a requisição para o formato especificado pelos processos JPD do WLI. Feito isso, a mensagem será processada pelo processo específico do WLI de forma a invocar os métodos do JDBC Control criado de forma a realizar as consultas necessárias para cada operação. De posse dos dados consultados, o WLI devolverá as informações obtidas ao ALSB, que se encarregará de formatar a resposta para o formato esperado.

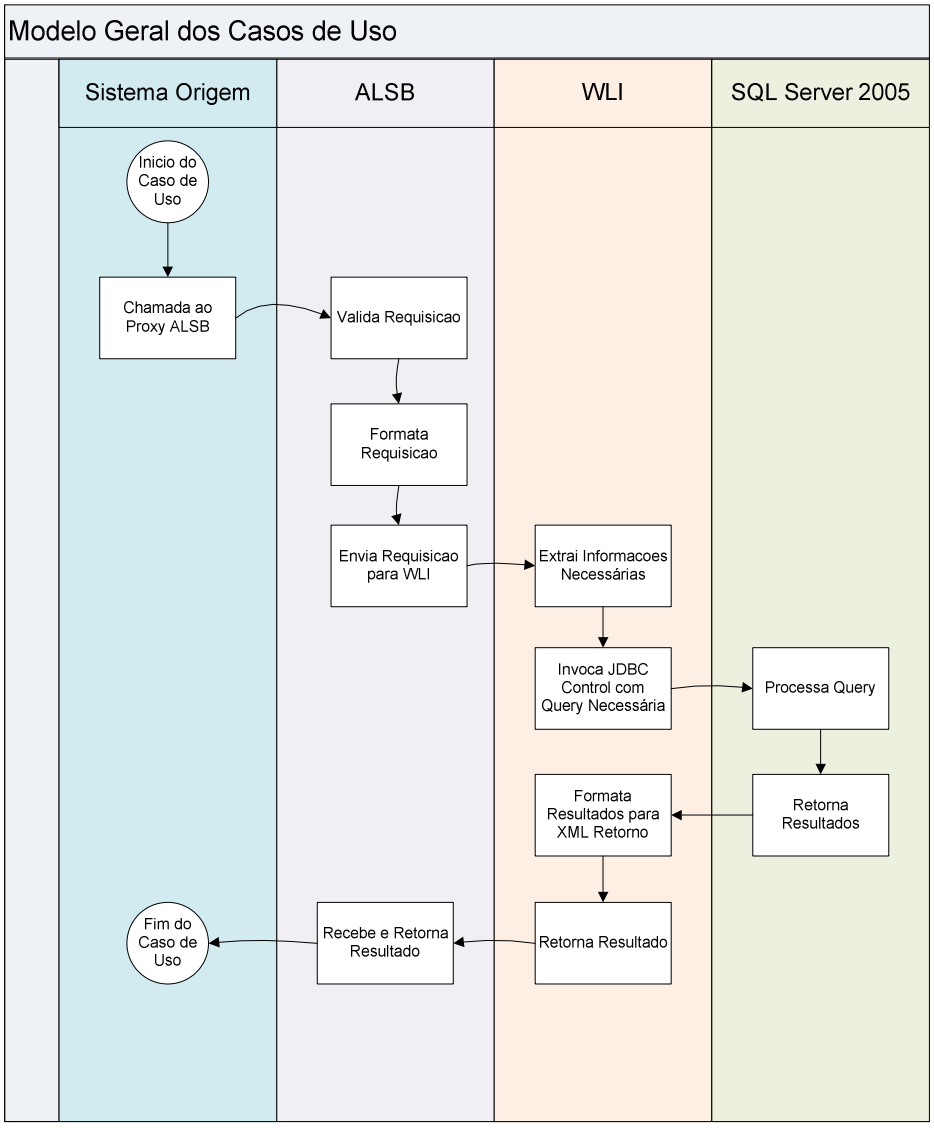

Figura 15 - Fluxo Básico dos Casos de Uso

7.3. Lista de Interfaces

A tabela a seguir mostra a lista de interfaces e sua organização no ALSB. Haverá um projeto criado no ALSB, com um *Proxy Service* que será a porta de entrada para a chamada de qualquer operação de consulta ao Banco de Dados PF contido no SQL Server 2005 Express. Este Projeto se chamará ConsultaSQL, já pensando no conceito de reaproveitamento da Arquitetura Orientada a Serviços, de forma que seja possível no futuro a inserção de novos *Proxies* para realizar consultas em outros bancos do mesmo sistema.

O nome do *Proxy Service* escolhido denota o escopo deste projeto que é a consulta a informações do banco de dados. Sua escolha é propícia e permite o entendimento que as operações que podem ser realizadas neste Proxy são de consulta de dados e que são baseadas no protocolo SOAP sobre HTTP. Inserções e remoções no Banco de Dados estão fora do escopo deste projeto.

O nome de cada operação permite que seja entendida de forma clara e objetiva a consulta realizada no banco, de forma que qualquer sistema poderá reaproveitar os serviços oferecidos.

Tabela 2 - Lista de Interfaces ALSB

Projeto	Proxy Services	Operação	Sistema de Origem	Sistema Destino
ConsultaSQL	BuscarInformacoesSQLProxySoap	BuscarDadosPessoaPorCpf	Qualquer	SQL Server 2005
		BuscarContatosPessoaPorCpf	Qualquer	SQL Server 2005
		BuscarFuncaoPessoaPorCpf	Qualquer	SQL Server 2005
		BuscarContatoPessoasPorAreaAtuacao	Qualquer	SQL Server 2005
		BuscarContatoPessoaPorNomeSobrenome	Qualquer	SQL Server 2005
		BuscarTodasInformacoesPessoaPorCpf	Qualquer	SQL Server 2005

7.4. Descrição dos Serviços no ALSB

7.4.1. Premissas

Será disponibilizada para todos os serviços uma única interface no protocolo SOAP com múltiplas operações. Todas as requisições chegarão ao *Proxy Service* BuscarInformacoesSQLProxySoap no ALSB. Para a construção do *Proxy Service* no ALSB será levado em conta o padrão recomendado internacionalmente pela BEA que é o VETO (iniciais para Validar, Enriquecer, Transformar e Operar), conforme descrito a seguir:

Tabela 3 - Padrão VETO

Padrão	Significado	Descrição
V	*Validate*	Validação do XML de entrada contra um WSDL ou XSD através de action do tipo Validate.
E	*Enrich*	Enriquecimento da mensagem XML com dados obtidos de serviços externos por meio de actions do tipo Service Callout e Java Callout.
T	*Transform*	Transformações na mensagem XML com o objetivo de adaptá-la ao formato de mensagem esperado pelo serviço configurado no business service através de actions como Insert, Delete, Rename e Replace.
O	*Operate*	Chamada de uma operação disponibilizada pelo serviço configurado no business service através de action de Route.

7.4.2. Propósito

Este *Proxy Service* será responsável por expor no Barramento SOA serviços que envolvam consulta à dados no SQL Server 2005 Express.

7.4.3. Endereço

O endereço do Proxy Service será:
http://<servidor>:<porta>/ConsultaSQL/BuscarInformacoesSQLProxySoap

7.4.4. Particularidades

O comportamento geral do *proxy service* é:

- Receber Requisição
- Armazenar os dados da requisição;
- Determinar operação a ser realizada;
- Validar a requisição;
- Formatar a requisição para o formato esperado pelo destino;
- Invocar Business Service vinculados à operação;
- Formatar a resposta para origem;
- Enviar Resposta.

7.4.5. Diagrama do Serviço

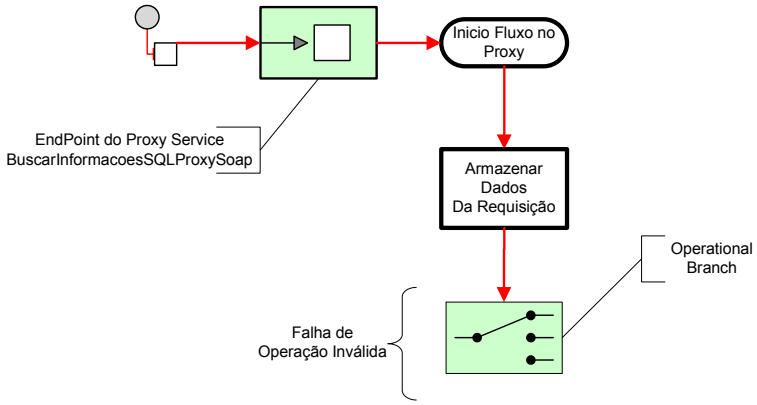

Figura 16 – Diagrama do Proxy Service ALSB

7.4.6. Descrição dos Estágios

7.4.6.1. Receber Requisição

O *Proxy Service* recebe em uma mensagem de requisição do sistema de origem no protocolo SOAP.

7.4.6.2. Armazenar Dados da Requisição

Nesse *stage* é armazenado o conteúdo da mensagem de requisição e as informações técnicas contidas na variável de sistema $inbound do ALSB. Essas informações armazenadas serão utilizadas para efetuar *logs* e para acrescentar informações no *report* de erro caso estes ocorram no Proxy Service.

Para armazenar essas informações serão utilizadas variáveis *customizadas*. As seguintes ações deverão ser *implementadas* nesse stage:

Tipo de Ação	Ação	Variável *Customizada*
Assign	$body	requestBody
Assign	$inbound	requestInbound

7.4.6.3. Operational Branch

Após terem sido armazenados os dados da mensagem de requisição, a próxima etapa no fluxo de processamento do *proxy service* é selecionar qual fluxo seguir. Desta forma, de acordo com a mensagem de requisição o *proxy* vai selecionar automaticamente qual fluxo seguir.

Quando se cria um *operational branch* no ALSB, obrigatoriamente será criado um *branch Default*. Nesse *branch Default* será feito um tratamento para o caso do sistema de origem enviar uma mensagem de requisição invocando alguma operação diferente das definidas no WSDL. O tratamento vai consistir em criar um XML de erro, lançar a exceção para o Top Level Error Handler do Proxy.

Operational Branch
BuscarDadosPessoaPorCpf
BuscarContatosPessoaPorCpf
BuscarFuncaoPessoaPorCpf
BuscarContatoPessoasPorAreaAtuacao
BuscarContatoPessoaPorNomeSobrenome
BuscarTodasInformacoesPessoaPorCpf
Default

7.4.6.3.1. *Default*

O comportamento geral do *branch* é lançar erro de operação inexistente.

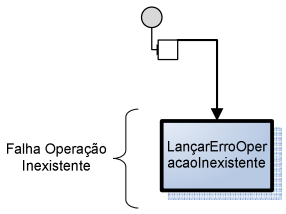

Figura 17 – Comportamento do *Branch Default*

LançarErroOperacaoInexistente

Nesse *stage* é lançado um erro de operação inexistente.

Tipo de Ação	Error Code	Mensagem
Raise Error	ESB-3002	Operacao Inexistente Solicitada no XML de entrada

7.4.6.3.2. *Diagrama geral das Operações*

Todas as operações terão um diagrama de fluxo bastante parecido no ALSB, com mudança e peculiaridades apenas dentro de cada estágio de validação de mensagem para o tipo esperado por cada operação e também para a formatação de requisição e resposta de cada operação.

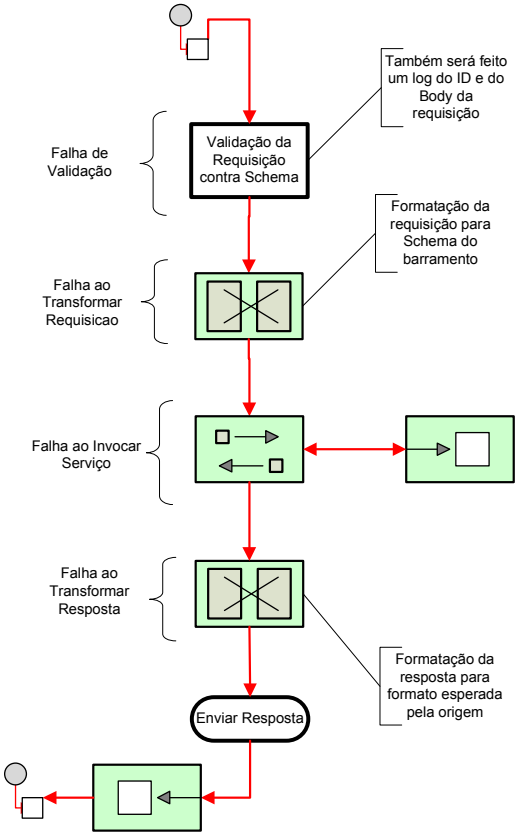

Figura 18 - Diagrama Geral do Fluxo das Operações

7.4.6.3.3. *BuscarDadosPessoaPorCpf*

O comportamento geral do branch é:

- Validar a requisição;
- Formatar a requisição para o formato esperado pelo destino;
- Invocar Business Service vinculados à operação;
- Formatar a resposta para origem;
- Enviar Resposta para o Proxy Service.

Validação da requisição contra schema

Nesse *stage* a mensagem de requisição enviada pela origem para o *proxy service* será validada para verificar se os parâmetros recebidos estão de acordo com o elemento BuscarDadosPessoaPorCpfRequest definidos no Schema BuscarDadosPessoaPorCpf.xsd.

Formatação da requisição

Será feita a formatação da mensagem de requisição enviada pela origem para o formato esperado pelo WLI.

Invoca Business Service

Invoca o *business service* BuscarDadosPessoaPorCpfProcessBizSoap passando a mensagem de requisição, invocando a operação "BuscarDadosPessoaPorCpfRequest". O *business service* receberá resposta síncrona.

Também armazenará o conteúdo da mensagem de resposta retornada, assim como as informações técnicas contidas na variável de sistema $outbound do ALSB. Essas informações armazenadas serão utilizadas para efetuar *logs* e para acrescentar informações nos *reports* de erros que ocorram no proxy service.

Para armazenar essas informações serão utilizadas variáveis *customizadas*. As seguintes ações deverão ser *implementadas* nesse *stage*:

Tipo de Ação	Ação	Variável *Customizada*
Assign	$body	responseBody
Assign	$outbound	responseOutbound

Formatação da resposta para formato esperado pela origem

Nesse *stage* será feito o mapeamento da mensagem XML de resposta do formato do WLI para o formato esperado pela origem.

Enviar Resposta

Envia a mensagem de resposta formatada para o sistema de origem.

7.4.6.3.4. *BuscarContatosPessoaPorCpf*

O comportamento geral do branch é:

- Validar a requisição;
- Formatar a requisição para o formato esperado pelo destino;
- Invocar Business Service vinculados à operação;
- Formatar a resposta para origem;
- Enviar Resposta para o Proxy Service.

Validação da requisição contra schema

Nesse *stage* a mensagem de requisição enviada pela origem para o *proxy service* será validada para verificar se os parâmetros recebidos estão de acordo com o elemento BuscarContatosPessoaPorCpfRequest definidos no Schema BuscarContatosPessoaPorCpf.xsd.

Formatação da requisição

Será feita a formatação da mensagem de requisição enviada pela origem para o formato esperado pelo WLI.

Invoca Business Service

Invoca o *business service* BuscarContatosPessoaPorCpfProcessBizSoap passando a mensagem de requisição, invocando a operação "BuscarContatosPessoaPorCpfRequest". O *business service* receberá resposta síncrona.

Também armazenará o conteúdo da mensagem de resposta retornada, assim como as informações técnicas contidas na variável de sistema $outbound do ALSB. Essas informações armazenadas serão utilizadas para efetuar *logs* e para acrescentar informações nos *reports* de erros que ocorram no *proxy service*.

Para armazenar essas informações serão utilizadas variáveis *customizadas*. As seguintes ações deverão ser *implementadas* nesse *stage*:

Tipo de Ação	Ação	Variável *Customizada*
Assign	$body	responseBody
Assign	$outbound	responseOutbound

Formatação da resposta para formato esperado pela origem

Nesse *stage* será feito o mapeamento da mensagem XML de resposta do formato do WLI para o formato esperado pela origem.

Enviar Resposta

Envia a mensagem de resposta formatada para o sistema de origem.

7.4.6.3.5. *BuscarFuncaoPessoaPorCpf*

O comportamento geral do *branch* é:

- Validar a requisição;
- Formatar a requisição para o formato esperado pelo destino;
- Invocar *Business Service* vinculados à operação;
- Formatar a resposta para origem;
- Enviar Resposta para o *Proxy Service*.

Validação da requisição contra schema

Nesse *stage* a mensagem de requisição enviada pela origem para o *proxy service* será validada para verificar se os parâmetros recebidos estão de acordo com o elemento BuscarFuncaoPessoaPorCpfRequest definidos no Schema BuscarFuncaoPessoaPorCpf.xsd.

Formatação da requisição

Será feita a formatação da mensagem de requisição enviada pela origem para o formato esperado pelo WLI.

Invoca Business Service

Invoca o *business service* BuscarFuncaoPessoaPorCpfProcessBizSoap passando a mensagem de requisição, invocando a operação "BuscarFuncaoPessoaPorCpfRequest". O *business service* receberá resposta síncrona.

Também armazenará o conteúdo da mensagem de resposta retornada, assim como as informações técnicas contidas na variável de sistema $outbound do ALSB. Essas informações armazenadas serão utilizadas para efetuar *logs* e para acrescentar informações nos *reports* de erros que ocorram no proxy service.

Para armazenar essas informações serão utilizadas variáveis *customizadas*. As seguintes ações deverão ser *implementadas* nesse stage:

Tipo de Ação	Ação	Variável ***Customizada***
Assign	$body	responseBody
Assign	$outbound	responseOutbound

Formatação da resposta para formato esperado pela origem

Nesse *stage* será feito o mapeamento da mensagem XML de resposta do formato do WLI para o formato esperado pela origem.

Enviar Resposta

Envia a mensagem de resposta formatada para o sistema de origem.

7.4.6.3.6. *BuscarContatoPessoasPorAreaAtuacao*

O comportamento geral do branch é:

- Validar a requisição;
- Formatar a requisição para o formato esperado pelo destino;
- Invocar *Business Service* vinculados à operação;
- Formatar a resposta para origem;
- Enviar Resposta para o *Proxy Service*.

Validação da requisição contra schema

Nesse *stage* a mensagem de requisição enviada pela origem para o *proxy service* será validada para verificar se os parâmetros recebidos estão de acordo com o elemento BuscarContatoPessoasPorAreaAtuacaoRequest definidos no Schema BuscarContatoPessoasPorAreaAtuacao.xsd.

Formatação da requisição

Será feita a formatação da mensagem de requisição enviada pela origem para o formato esperado pelo WLI.

Invoca Business Service

Invoca o *business service* BuscarContatoPessoasPorAreaConhecimentoProcessBizSoap passando a mensagem de requisição, invocando a operação "BuscarContatoPessoasPorAreaConhecimentoRequest". O *business service* receberá resposta síncrona.

Também armazenará o conteúdo da mensagem de resposta retornada, assim como as informações técnicas contidas na variável de sistema $outbound do ALSB. Essas informações armazenadas serão utilizadas para efetuar *logs* e para acrescentar informações nos *reports* de erros que ocorram no *proxy service*.

Para armazenar essas informações serão utilizadas variáveis *customizadas*. As seguintes ações deverão ser *implementadas* nesse *stage*:

Tipo de Ação	Ação	Variável *Customizada*
Assign	$body	responseBody
Assign	$outbound	responseOutbound

Formatação da resposta para formato esperado pela origem

Nesse *stage* será feito o mapeamento da mensagem XML de resposta do formato do WLI para o formato esperado pela origem.

Enviar Resposta

Envia a mensagem de resposta formatada para o sistema de origem.

7.4.6.3.7. *BuscarContatoPessoaPorNomeSobrenome*

O comportamento geral do *branch* é:

- Validar a requisição;
- Formatar a requisição para o formato esperado pelo destino;
- Invocar *Business Service* vinculados à operação;
- Formatar a resposta para origem;
- Enviar Resposta para o *Proxy Service*.

Validação da requisição contra schema

Nesse *stage* a mensagem de requisição enviada pela origem para o *proxy service* será validada para verificar se os parâmetros recebidos estão de acordo com o elemento BuscarContatoPessoaPorNomeSobrenomeRequest definidos no *Schema* BuscarContatoPessoaPorNomeSobrenome.xsd.

Formatação da requisição

Será feita a formatação da mensagem de requisição enviada pela origem para o formato esperado pelo WLI.

Invoca Business Service

Invoca o *business service* BuscarContatoPessoaPorNomeSobrenomeProcessBizSoap passando a mensagem de requisição, invocando a operação "BuscarContatoPessoaPorNomeSobrenomeRequest". O business service receberá resposta síncrona.

Também armazenará o conteúdo da mensagem de resposta retornada, assim como as informações técnicas contidas na variável de sistema $outbound do ALSB. Essas informações armazenadas serão utilizadas para efetuar *logs* e para acrescentar informações nos *reports* de erros que ocorram no *proxy service*.

Para armazenar essas informações serão utilizadas variáveis *customizadas*. As seguintes ações deverão ser *implementadas* nesse *stage*:

Tipo de Ação	Ação	Variável *Customizada*
Assign	$body	responseBody
Assign	$outbound	responseOutbound

Formatação da resposta para formato esperado pela origem

Nesse *stage* será feito o mapeamento da mensagem XML de resposta do formato do WLI para o formato esperado pela origem.

Enviar Resposta

Envia a mensagem de resposta formatada para o sistema de origem.

7.4.6.3.8. *BuscarTodasInformacoesPessoaPorCpf*

O comportamento geral do branch é:

- Validar a requisição;
- Formatar a requisição para o formato esperado pelo destino;
- Invocar *Business Service* vinculados à operação;
- Formatar a resposta para origem;
- Enviar Resposta para o *Proxy Service*.

Validação da requisição contra schema

Nesse *stage* a mensagem de requisição enviada pela origem para o *proxy service* será validada para verificar se os parâmetros recebidos estão de acordo com o elemento

BuscarTodasInformacoesPessoaPorCpfRequest definidos no Schema BuscarTodasInformacoesPessoaPorCpf.xsd.

Formatação da requisição

Será feita a formatação da mensagem de requisição enviada pela origem para o formato esperado pelo WLI.

Invoca Business Service

Invoca o business service BuscarTodasInformacoesPessoaPorCpfProcessBizSoap passando a mensagem de requisição, invocando a operação "BuscarTodasInformacoesPessoaPorCpfRequest". O *business service* receberá resposta síncrona.

Também armazenará o conteúdo da mensagem de resposta retornada, assim como as informações técnicas contidas na variável de sistema $outbound do ALSB. Essas informações armazenadas serão utilizadas para efetuar *logs* e para acrescentar informações nos *reports* de erros que ocorram no *proxy service*.

Para armazenar essas informações serão utilizadas variáveis *customizadas*. As seguintes ações deverão ser *implementadas* nesse *stage*:

Tipo de Ação	Ação	Variável ***Customizada***
Assign	$body	responseBody
Assign	$outbound	responseOutbound

Formatação da resposta para formato esperado pela origem

Nesse *stage* será feito o mapeamento da mensagem XML de resposta do formato do WLI para o formato esperado pela origem.

Enviar Resposta

Envia a mensagem de resposta formatada para o sistema de origem.

7.5. Organização do Projeto no ALSB

Tabela 4 - Organização do Projeto no ALSB

Projeto	Pasta		Objeto
ConsultaSQL	ProxyServices		BuscarInformacoesSQLProxySoap
	BusinessServices		BuscarDadosPessoaPorCpfBizSoap BuscarContatosPessoaPorCpfBizSoap BuscarFuncaoPessoaPorCpfBizSoap BuscarContatoPessoasPorAreaConhecimentoBizSoap BuscarContatoPessoaPorNomeSobrenomeBizSoap BuscarTodasInformacoesPessoaPorCpfBizSoap
	WSDLs		BuscarInformacoesSQL.wsdl
	Schemas		BuscarDadosPessoaPorCpf.xsd BuscarContatosPessoaPorCpf.xsd BuscarFuncaoPessoaPorCpf.xsd BuscarContatoPessoasPorAreaConhecimento.xsd BuscarContatoPessoaPorNomeSobrenome.xsd BuscarTodasInformacoesPessoaPorCpf.xsd
	Xqueries	Commons	XQueries comuns que podem ser usadas por mais de uma interface do projeto
		BuscarDadosPessoaPorCpf	XQueries relacionadas a operação BuscarDadosPessoaPorCpf
		BuscarContatosPessoaPorCpf	XQueries relacionadas a operação BuscarContatosPessoaPorCpf
		BuscarFuncaoPessoaPorCpf	XQueries relacionadas a operação BuscarFuncaoPessoaPorCpf
		BuscarContatoPessoasPorAreaAtuacao	XQueries relacionadas a operação BuscarContatoPessoasPorAreaAtuacao
		BuscarContatoPessoaPorNomeSobrenome	XQueries relacionadas a operação BuscarDadosPessoaPorNomeSobrenome
		BuscarTodasInformacoesPessoaPorCpf	XQueries relacionadas a operação BuscarTodasInformacoesPessoaPorCpf

7.6. Descrição dos Processos no WLI

7.6.1. Premissas

Será criado um JDBC *Control* que fará as consultas ao banco de dados de acordo com a necessidade de cada operação. Será criado um processo para cada operação presente no ALSB, sendo que os processos poderão e deverão ser reutilizados de forma a *otimizar* ao máximo a construção e diminuir o tempo para finalização do projeto.

Neste momento poderiam ser adotadas duas soluções. A primeira delas seria colocar o controle das consultas no Banco de Dados sob o controle de uma equipe de administradores de Banco de Dados, que realizassem *Stored Procedures* e as disponibilizassem para o barramento SOA descrevendo os parâmetros de entrada e de saída de cada *Stored Procedure*. Isto traria o

benefício de uma equipe especializada tratar das consultas ao Banco de forma que estas seriam as mais *otimizadas* possíveis e garantir que a integridade dos dados estaria preservada. A segunda e escolhida solução é a de colocar a cargo do barramento SOA a consulta ao Banco de Dados. Esta opção foi escolhida, pois o Administrador do Banco de Dados é o próprio desenvolvedor do Barramento SOA. Além disso, como serão efetuadas apenas consultas, sem haver o perigo de alterar a integridade dos dados do Banco, não há uma grande preocupação neste sentido. Esta solução é bastante favorável para o caso de consultas que não alterem os estados das tabelas do Banco de Dados, pois não há a necessidade de se contactar o administrador do Banco de Dados para gerar uma nova *Stored Procedure* para uma nova consulta a ser realizada. A mesma pode ser criada através de um novo método no JDBC *Control* e disponibilizada para os processos desenvolvidos.

7.6.2. Propósito

Cada processo no WLI permite a inserção de código Java, reutilização de componentes e a disponibilização dos mesmos como *Web Services* de forma a aumentar a possibilidade de reutilização futura.

7.6.3. Endereço

O endereço de cada processo JPD no WLI será:
http://<servidor>:<porta>/ConsultaSQL/src/processess/<NomeDoProcesso>

7.6.4. Desenho da Solução

7.6.4.1. JDBC Control

Um *JDBC Data Source* será criado a partir do servidor do WLI, com o *driver* de conexão com o Banco de Dados SQL Server 2005 Express mais as configurações de servidor, porta, usuário, senha e database a ser consultado. A partir da criação deste *Data Source*, na IDE do WLI será possível criar um JDBC *Control* que é um controle que abstrai as funções de conexão com o Banco de Dados de forma que seja possível e relativamente mais fácil realizar as consultas aos dados do Banco.

Neste JDBC Control serão criadas classes com elementos com os mesmos nomes esperados das consultas a serem realizadas no Banco de Dados. Isso implica em mapear todos os dados já expostos anteriormente nas tabelas do Banco de Dados em classes Java. Ou seja, para cada tabela do Banco de Dados haverá uma classe cujo nome será o mesmo da tabela e os seus elementos terão os mesmos nomes das colunas das tabelas do Banco. Essa é uma limitação da ferramenta *Weblogic Integration* e para o funcionamento correto do JDBC *Control* há a

necessidade de se criar classes isoladas para cada Tabela do Banco de Dados a ser consultada. Isso será necessário para que o resultado de uma *query* seja mapeado para uma classe a ser utilizada no processo.

Métodos serão criados de forma que seja possível realizar as *queries* no Banco de Dados de forma genérica com passagem de parâmetros para o método e também permitir a reutilização dos métodos por todo o projeto.

Por exemplo, para o caso da consulta mais simples a ser realizada no Banco de Dados, a consulta de um estado civil a partir do id do estado civil, a seguinte classe deve ser criada (mimetizando a tabela do Banco de Dados já mostrada anteriormente):

```
static public class EstadoCivil {
          private int idEstadoCivil;
          private String estadoCivil;
          public String getEstadoCivil() {
                 return estadoCivil;
          }
          public void setEstadoCivil(String estadoCivil) {
                 this.estadoCivil = estadoCivil;
          }
          public int getIdEstadoCivil() {
                 return idEstadoCivil;
          }
          public void setIdEstadoCivil(int idEstadoCivil) {
                 this.idEstadoCivil = idEstadoCivil;
          }
    }
```

A partir desta classe, o seguinte método pode ser criado:

@JdbcControl.SQL(statement="SELECT * *FROM [PF].[dbo].[EstadoCivil] WHERE (idEstadoCivil={idEstadoCivil})")*

EstadoCivil buscarEstadoCivilPorId(int idEstadoCivil) throws SQLException;

7.6.4.2. **Processos no WLI**

A partir da criação de métodos no JDBC *Control*, como foi citado anteriormente, é possível importar o Controle gerado de forma que seus métodos apareçam disponíveis na interface gráfica do WLI. Juntamente com o *Control*, serão importados os *schemas* XML das requisições e respostas a serem tratadas pelo WLI de forma que seja possível seguir um padrão e que exista uma maior facilidade de realização de transformações entre os dados.

Cada processo no WLI chamará um ou mais métodos necessários do JDBC *Control* de forma a consultar o Banco de Dados e enriquecer a mensagem da forma como foi proposta para cada operação. Por exemplo, na consulta de dados por CPF será utilizado um método que ao receber um CPF realiza uma consulta ao Banco de Dados e em caso de sucesso, uma classe Java que mapeia os campos da tabela Pessoa no Banco de Dados é devolvida. Esta classe é mapeada para a resposta esperada pelo ALSB e devolvida na chamada síncrona ao processo. Em caso de falha na consulta ao SQL, o erro é mapeado dentro da mensagem de retorno nos Parâmetros de Controle e é retornado para o ALSB de forma a notificar que houve uma falha na consulta. Por exemplo, para um caso de consulta que retorne valor NULL para o CPF solicitado, serão preenchidos os campos de Parâmetros de Controle da mensagem retornada de forma a notificar que a pessoa solicitada não está cadastrada no Banco de Dados.

Sempre haverá preocupação com o tratamento de erro de forma que seja possível informar a cada estágio o erro ocorrido. Os principais erros possíveis são de falha de transformação dos dados, falha de comunicação com o Banco de Dados no caso de uma interrupção de serviços e falha de dados inválidos retornados, pois a consulta feita é de dados não cadastrados no Banco de Dados. Todos esses erros serão mapeados dentro da estrutura ParametrosControle presente em todas as respostas do sistema.

Em seguida há uma figura que mostra a forma geral de um processo no ambiente gráfico do *Weblogic Integration*. Nela é possível ver a criação do controle e sua posterior importação e utilização no processo.

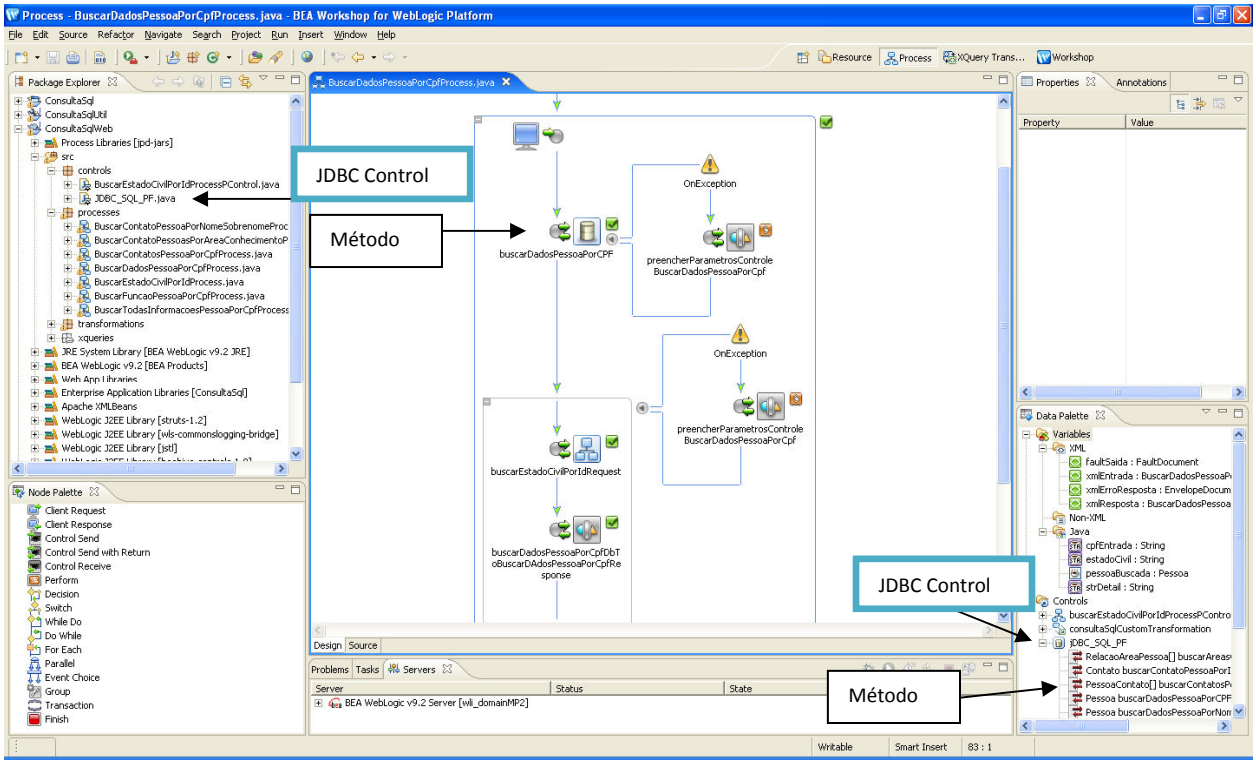

Figura 19 - Visão geral de um Processo no WLI

7.7. Organização de Projeto no WLI

Tabela 5 - Organização de Projeto no WLI

Projeto	Localização		Objeto
ConsultaSql			
ConsultaSqlUtil	schemas		Schemas necessários para o Projeto
ConsultaSqlWeb	src	controls	JDBC_SQL_PF.java (Controle criado para executar consultas no BD)
		processes	BuscarDadosPessoaPorCpfProcess BuscarContatosPessoaPorCpfProcess BuscarFuncaoPessoaPorCpfProcess BuscarContatoPessoasPorAreaAtuacaoProcess BuscarContatoPessoaPorNomeSobrenomeProcess BuscarTodasInformacoesPessoaPorCpfProcess
		transformations	ConsultaSqlCustomTransformation.java Outras Transformações Necessárias
		xqueries	Xqueries necessárias para o projeto

8. Resultados do Desenvolvimento

Os resultados obtidos na fase de construção atenderam plenamente ao que havia sido planejado e especificado. Isto pode ser comprovado pela bateria de testes executada com o serviço através de um programa chamado SOAPUI. Este programa é de licença GPL e é líder em testes de sistemas baseados em *Web Services* permitindo o envio de qualquer requisição SOAP para um determinado endereço. Através do SOAPUI foram feitos testes em cada um dos serviços criados de forma a verificar se o funcionamento atendia às especificações de projeto.

A interface do SOAPUI pode ser vista na figura seguinte, em que uma requisição de consultarDadosPorCpf foi enviada para o endereço do servidor do ALSB e foi recebido um retorno com as informações provenientes do Banco de Dados já mapeadas para XML. Com isso, fica demonstrado o potencial da Integração de Sistemas baseada em Arquitetura SOA, pois qualquer Sistema de Informação capaz de mandar uma requisição válida para o Proxy disponibilizado terá a resposta proveniente do Banco de Dados sem haver a necessidade de conhecer comandos de Bancos de Dados. Demonstra-se assim que múltiplos sistemas podem acessar o mesmo serviço para consultar o Banco de Dados, reduzindo assim ao longo do tempo o retrabalho de diversas integrações diferentes com o mesmo Banco de Dados para fazer a mesma coisa.

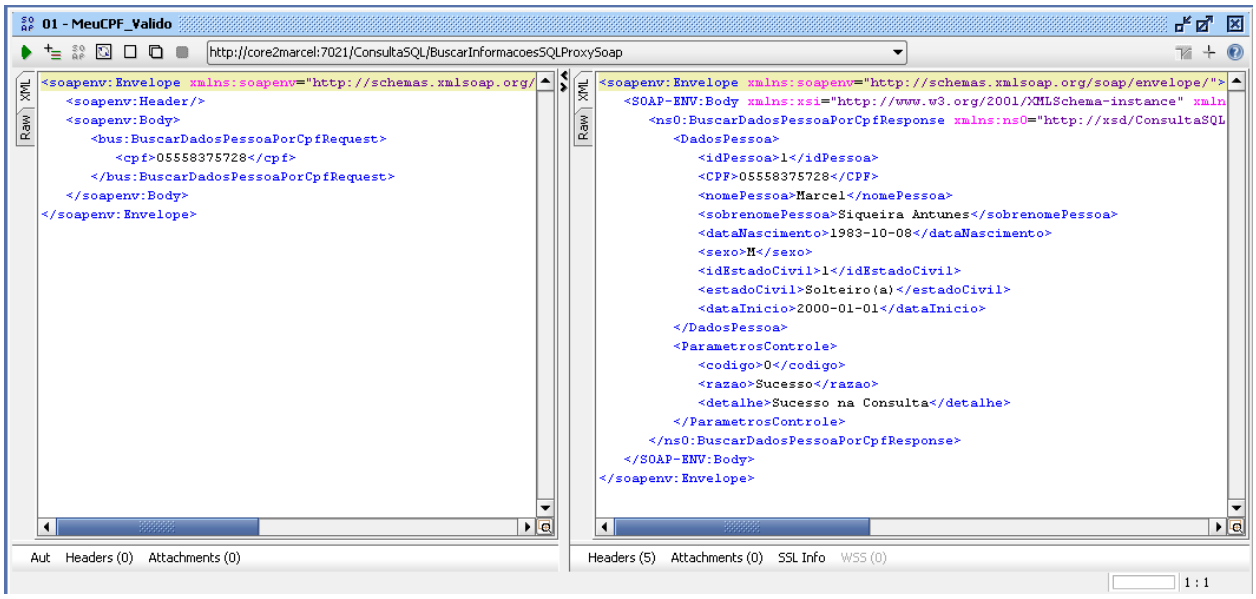

Figura 20 - Interface SOAPUI

Cada possibilidade de cada uma das operações no Proxy do ALSB e de cada um dos Processos no WLI foi testada de forma que o tratamento de erro fosse verificado. Foram feitos testes de falha de comunicação entre ALSB e WLI e também entre WLI e MS SQL Server 2005 Express. Em ambos os casos o cliente que faz uma requisição no ALSB recebe uma mensagem informando o erro que ocorreu de forma clara e precisa.

As evidências de teste ficam coletadas no SOAPUI de forma que servem de parâmetros de indicação de uma boa construção, assim como fontes para testes futuros no caso de alguma atualização e a necessidade de se executar novamente os testes para averiguação do funcionamento do sistema.

Como medida do bom funcionamento do sistema como um todo, foi obtido que o tempo de resposta de uma requisição da maior das operações, que é a de Consultar Contatos das Pessoas por Área de Conhecimento, é em média de 100 milissegundos com todos os servidores rodando na mesma máquina. Vale salientar que as versões utilizadas tanto do ALSB quanto do WLI são livres para desenvolvedores de aplicações de pesquisa ou não lucrativas. Esta versão possui uma limitação quanto ao número de IPs que pode acessar o servidor do ALSB e do WLI, sendo esse número um total de 5 IPs por servidor. Há também um desempenho menor, pois há uma limitação no número máximo de threads possíveis de serem executadas. Para aplicações de larga escala e que exijam muito desempenho é necessário recorrer à versão comercial de tais aplicações.

A única restrição encontrada na construção do projeto foi a criação de classes com herança e relacionamento e seu posterior funcionamento com o JDBC Control. Por uma limitação da ferramenta (WLI) não é possível executar métodos no JDBC Control sem que a Classe que receberá os resultados da consulta possua única e exclusivamente os campos presentes nas tabelas consultadas. Com classes que possuam campos diferentes das tabelas consultadas parece haver uma perda de referência e ocasionamento de erro no processo. Por isso não foi criado um diagrama de entidade e relacionamento das classes Java, pois elas foram criadas sem relacionamento, espelhando apenas o que constava nas tabelas físicas do Banco de Dados.

O valor estimado de linhas de código pelo software de métrica (COCOMO) na fase de Planejamento foi de cerca de cinco mil (5.000) linhas de código, levando-se em conta os seis casos de uso propostos. Este valor ficou próximo do valor real obtido, que aparece demonstrado na tabela abaixo. O valor real foi de aproximadamente seis mil (6.000) linhas de código, ultrapassando a estimativa prevista. A estimativa de tempo para finalização do projeto acabou sendo ultrapassada, com seu término previsto para Março de 2008 sendo estendido até Maio de 2008 principalmente por conta da construção, isto é, devido à complexidade e novidade dos serviços gerados.

Tabela 6 - Total de Linhas de Código por Objeto

Objeto	Quantidade	Linhas Por Objeto	SubTotal
Xqueries	20	60	1200
Processos	7	200	1400
Schemas	12	50	600
Control	1	500	500
Codigo ALSB	8	300	2400
Total			6100

A estimativa de horas utilizada previa uma carga horária diária de três horas ao dia de dedicação exclusiva ao Projeto Final, de forma que seriam necessárias por volta de duzentas horas para completar todo o Projeto. Essas duzentas horas aproximadas demandariam aproximadamente sessenta e sete dias de construção. Como a impossibilidade de dedicação de três horas diárias às vezes se apresentou, aliando-se ainda a subestimativa do tamanho e complexidade do projeto em termos de linhas de código, o que se obteve realmente foi um total de aproximadamente noventa dias de construção. Os tempos de construção de cada um dos objetos podem ser vistos na tabela abaixo.

Tabela 7 - Total de Horas por Objeto

Objeto	Quantidade	Tempo Por Objeto (h)	SubTotal (h)
Xqueries	20	0,5	10
Processos	7	20	140
Schemas	12	1	12
Control	1	50	50
Codigo ALSB	8	8	64
Total			276

Apesar do tempo a mais que foi necessário para a realização do projeto, o conhecimento trazido junto com ele foi bastante gratificante e o resultado obtido foi de enorme satisfação para o desenvolvedor.

9. Conclusão

Ao final deste projeto de fim de curso é possível perceber que a Integração de Sistemas baseada em Arquitetura Orientada a Serviços tem um grande potencial e cumpre um dos seus objetivos, que é criar uma topologia de sistemas interligados por Barramento e não por estrela, permitindo uma grande reutilização de serviços. Essa tecnologia começa a ser bastante aproveitada no âmbito profissional e acadêmico devido à imensa gama de possibilidades oferecidas.

Com o desenvolvimento deste projeto pode-se observar uma integração com um sistema de Banco de Dados que poderia ser o CRM ou Mainframe de alguma grande companhia disponibilizando várias de suas consultas através de *Web Services* totalmente genéricos, sem a necessidade de conhecimento de comandos SQL por parte do sistema origem e sem limitação de plataforma ou ambiente a ser utilizado. Mais do que isso, através de um único *endpoint* (o do Proxy Service no ALSB) todas as consultas podem ser realizadas de acordo com a operação selecionada através da mensagem de entrada. Isso significa que o sistema que quiser utilizar qualquer uma das consultas ao Banco de Dados, poderá fazê-lo através de um único endereço e não por seis endereços diferentes que seria o caso de uma integração tradicional operação a operação.

Do ponto de vista do aprendizado este projeto foi de suma importância acadêmica, pois fez com que todos os conhecimentos aprendidos ao longo da Engenharia Eletrônica e de Computação da Universidade Federal do Rio de Janeiro servissem de base para o estudo e aprofundamento de uma tecnologia que é nova, possui um grande potencial, tem uma grande visão de mercado e possui poucos profissionais especializados e com bom conhecimento. Este projeto propiciou um aprendizado constante e a colocação em prática de vários conceitos aprendidos ao longo da faculdade.

Como sugestão para projetos futuros fica a possibilidade de utilização de toda a infra-estrutura de barramento sob a Arquitetura Orientada a Serviços, assim como a utilização dos serviços criados neste projeto de forma a gerar novos sistemas com arquitetura SOA que acessem múltiplos destinos. Por exemplo, pode ser gerado um novo sistema que além de consultar dados de uma pessoa na Base de Dados agora use estas informações em algum outro serviço na Web ou em outra plataforma e retorne os dados consolidados. As possibilidades nessa área são muito grandes e trazem muito conhecimento novo que o mercado demonstra bastante interesse.

Bibliografia

[1] WS-I *Web Services Interoperability Organization* < http://www.ws-i.org>. Acessado em 15 de Janeiro de 2008.

[2] OASIS <http://www.oasis-open.org>. Acessado em 15 de Janeiro de 2008.

[3] W3C – *World Wide Web Consortium* <http://w3c.org/2002/ws/>. Acessado em 15 de Janeiro de 2008.

[4] BEA – Fabricante de ALSB e WLI <http://www.bea.com>. Acessado em 08 de Fevereiro de 2008.

[5] *World Wide Web Consortium* - Especificação SOAP 1.1 <http://www.w3.org/TR/SOAP/>. Acessado em 08 de Fevereiro de 2008.

[6] *World Wide Web Consortium* - Especificação SOAP 1.2 <http://www.w3.org/TR/soap12-part0/>. Acessado em 08 de Fevereiro de 2008.

[7] *World Wide Web Consortium* - SOAP com anexos <http://www.w3.org/TR/soap12-af/>. Acessado em 09 de Fevereiro de 2008.

[8] *World Wide Web Consortium* - WSDL <http://www.w3.org/TR/wsdl>. Acessado em 09 de Fevereiro de 2008.

[9] MSDN – Usando *Inner Joins* <http://msdn2.microsoft.com/en-us/library/ms190014.aspx>. Acessado em 15 de Março de 2008.

[10] MSDN – *Subqueries e Alias* <http://msdn2.microsoft.com/en-us/library/ms190410. aspx>. Acessado em 15 de Março de 2008.

[11] SOAPUI – Programa de Teste <http://www.soapui.org/>. Acessado em 16 de Maio de 2008.

[12] Eclipse – IDE que serve de molde para o WLI < http://www.eclipse.org/>. Acessado em 16 de Maio de 2008.

[13] SubVersion - SVN – Programa para Versionamento <http://subversion.tigris.org/>. Acessado em 16 de Maio de 2008.

Apêndice I – PGPS

11.1. Apresentação

11.1.1. Sumário do Projeto

11.1.1.1. Finalidade, Escopo e Objetivos

O Projeto tem como finalidade realizar a integração entre dois sistemas, baseando-se no paradigma da Arquitetura Orientada a Serviços. O objetivo do projeto é disponibilizar um serviço capaz de ser acessado por qualquer aplicativo capaz de se adequar a um WSDL, facilitando e encapsulando o acesso a informações provenientes de outro sistema.

11.1.1.2. Postulados e Restrições

Há responsabilidade profissional, que impede que ocorra dedicação total ao projeto. Como postulado há a utilização softwares livres, ou sem necessidade de pagamento de licenças, devido a restrições orçamentárias. Outro postulado seria a linguagem. Foi estabelecida a utilização de linguagem JAVA para codificar o projeto. Serão utilizadas as ferramentas da BEA, livres para desenvolvimento não comercial, *Aqualogic Service Bus* e Weblogic Integration. Estas ferramentas são baseadas em ambiente Web e Eclipse, respectivamente, e possuem diversas facilidades de desenvolvimento baseado em SOA.

11.1.1.3. Liberações Parciais

Não se aplica. Será liberada apenas uma versão, a versão final.

11.1.1.4. Sumário de Cronograma

O cronograma deste projeto se iniciou com o primeiro contato com o Professor-Orientador Antônio Cláudio e com a decisão do que seria executado no Projeto Final. O término estimado para conclusão do mesmo é para Março de 2008 com entrega de toda a documentação e *implementação* referente ao projeto.

11.1.2. Evolução do Plano

A evolução do projeto será marcada por reuniões periódicas com o Professor-Orientador, de forma a nortear da melhor maneira o andamento do projeto. Estas reuniões serão os marcos úteis para o alinhamento das definições, especificações e verificação do obedecimento deste planejamento.

11.2. Organização do Projeto

11.2.1. Interfaces Externas

11.2.2. Estrutura Interna

O projeto está sendo realizado por um desenvolvedor. O Professor-Orientador atuará no controle de qualidade e validação durante todo o processo.

11.2.3. Papéis e Responsabilidades

Não se aplica. O único integrante do projeto assume todos os papéis e responsabilidades.

11.3. Processos de Gerenciamento

11.3.1. Partida no Projeto

11.3.1.1. Previsões

Há previsão de que o projeto esteja pronto em sua totalidade ao final do mês de Março de 2008.

11.3.1.2. Equipe

Não se aplica. Existe apenas um integrante no projeto.

11.3.1.3. Plano de Aquisição de Recursos

Não se aplica. Este projeto é individual.

11.3.1.4. Plano de Treinamento da Equipe

Haverá estudo durante o processo de planejamento e projeto de forma a consolidar o conhecimento adquirido ao longo dos anos de ensino do curso de Engenharia Eletrônica e de Computação. Isto servirá para solidificar os conceitos e aplicá-los da melhor forma possível à *implementação* prática.

11.3.2. Plano de Trabalho

11.3.2.1. Atividades

Todas as atividades são de responsabilidade do único membro da equipe.

11.3.2.2. Prazos

O primeiro prazo tido como meta é o mês de Março de 2008. Ao longo de Fevereiro e Março de 2008, reuniões com o Professor-Orientador acontecerão de forma a manter o alinhamento de cronograma.

11.3.2.3. Alocação de Recursos

Não se aplica.

11.3.2.4. Alocação de Orçamento

Não se aplica.

11.3.3. Planos de Controle

11.3.3.1. Controle dos Requisitos

Os requisitos serão controlados por meio de relatórios de modificações, caso hajam, no decorrer do desenvolvimento do projeto.

11.3.3.2. Controle dos Prazos

Utilização do software MS-Project para projetar, analisar, estimar e acompanhar o projeto ao longo do cronograma especificado.

11.3.3.3. Controle do Orçamento

Não se aplica.

11.3.3.4. Controle de Qualidade

O controle de qualidade será de responsabilidade do aluno e do Professor-Orientador de Projeto Final, que irá revisar a documentação aqui presente e opinar acerca das decisões tomadas em relação ao produto final.

11.3.3.5. Plano de Relatórios

Consiste na apresentação a cada reunião da documentação completa até a data.

11.3.3.6. Plano de Medidas

Ao longo do projeto, serão colecionadas as medidas do tempo gasto e da quantidade de linhas de código gerada em cada etapa do processo. Assim, ao final, os dados estatísticos de todo o processo estarão disponíveis.

11.3.4. Plano de Gerenciamento de Riscos

Os riscos foram identificados e analisados, montando a tabela de riscos abaixo. Os mais importantes serão gerenciados pelo plano RMMM:

Tabela 8 - Planilha RMMM

Risco	Classificação	Probabilidade	Impacto	Plano RMMM
Complexidade da Ferramenta	Técnico	80%	1	Plano Ferramenta
Prazo	Projeto	50%	2	Plano Prazo
Complexidade do Produto	Técnico	70%	1	Plano Complexidade
Inexperiência	Pessoal	60%	2	Plano Treinamento
Mudança de requisitos	Cliente	5%	2	----------
Custo elevado	Projeto	10%	3	----------
Aceitação do produto	Produto	10%	3	----------

O plano RMMM consiste em Mitigar, Monitorar e gerenciar os riscos do projeto. Abaixo são apresentados os planos RMMM:

- Plano Ferramenta:
  - Mitigar: Estudar a complexidade da Ferramenta a ser utilizada na arquitetura SOA, avaliando suas capacidades e limitações;
  - Monitorar: Monitorar as dificuldades de entender e utilizar a ferramenta;
  - Gerenciar: Ler documentação, consultar colegas com alguma experiência na ferramenta e pesquisar projetos semelhantes que foram *implementados* com a mesma ferramenta.

- Plano Prazo:
  - Mitigar: Estabelecer um cronograma compatível com a realidade;
  - Monitorar: Monitorar o andamento das atividades planejadas em cada etapa;
  - Gerenciar: Aumentar a carga horária trabalhada para suprir os atrasos.

- Plano Complexidade:
  - Mitigar: Estudar todos os requisitos e o melhor meio de *implementação* do projeto antes de começá-lo;
  - Monitorar: Monitorar as dificuldades encontradas na *implementação*;
  - Gerenciar: Estudar outros produtos semelhantes para entender como funcionam e, contornando assim, as dificuldades encontradas.

- Plano Treinamento:
  - Mitigar: Estudar sobre o uso das ferramentas escolhidas;
  - Monitorar: Verificar se o entendimento das ferramentas se aplica praticamente;

- Gerenciar: Aumentar a carga horária para suprir a falta de conhecimento de alguma peculiaridade da ferramenta.

11.3.5. Plano de Encerramento

Ao fim do Projeto Final, em Março de 2008, deverá ser entregue toda a documentação relativa ao software, juntamente com a *implementação* do mesmo em sua versão final.

11.4. Processos Técnicos

11.4.1. Modelo dos Processos

O projeto teve início em Dezembro de 2007 sendo encerrado até Março de 2008. A qualidade será controlada ao longo do projeto através de revisões técnicas feitas pelo Professor-Orientador, Antônio Cláudio.

11.4.2. Métodos, Ferramentas e Técnicas

Para o desenvolvimento deste projeto, ficou decidida a utilização das ferramentas Aqualogic Service Bus e Weblogic Integration da BEA, baseadas em plataforma Web e Eclipse e linguagem Java, por serem largamente empregadas no desenvolvimento de soluções baseadas em SOA. A comunicação entre os sistemas acontecerá baseada em SOAP sobre HTTP com uso de XML. Haverá utilização do software TortoiseSVN para controle de versão.

11.4.3. Infra-estrutura

Os ambientes de desenvolvimento escolhidos foram o Aqualogic Service Bus e Weblogic Integration, disponibilizados gratuitamente para uso não comercial.

11.4.4. Plano para a Aceitação do Produto

Consiste na apresentação, durante as reuniões, da evolução do produto e, ao final, uma sessão de testes com o produto na sua versão final.

11.5. Planos para os processos de Suporte

11.5.1. Gerenciamento de Configuração

Utilização do software TortoiseSVN para gerenciar as configurações do projeto. Este software permite o controle de versões, criação de múltiplas versões, acesso ao repositório via Web (por meio do software SubVersion), entre outros.

11.5.2. Plano de Verificação e de Validação

A verificação e validação do projeto ficarão a cargo do Professor-Orientador Antonio Cláudio, à medida que as reuniões acontecerem e o documento for apresentado. O Professor utilizará suas próprias técnicas para rastrear possíveis problemas com o projeto.

11.5.3. Documentação

A documentação do trabalho será feita à medida que o projeto avança, sendo finalizada na época do encerramento do projeto. Toda a documentação estará presente na monografia de fim de curso. Não estão previstos documentos não - liberáveis neste projeto.

11.5.4. Plano para Assegurar a Qualidade

Serão feitas modificações quando necessário para garantir a qualidade do produto, de acordo com o *feedback* fornecido pelo professor com as modificações propostas.

11.5.5. Revisões e Auditorias

As revisões e auditorias serão efetuadas pelo professor a cada reunião.

11.5.6. Plano para a Resolução de Problemas

O desenvolvedor se responsabiliza pela resolução de problemas.

11.5.7. Gerenciamento de Subcontratações

Não se aplica.

11.5.8. Plano de Aperfeiçoamento

Não ficou definido o aperfeiçoamento do projeto. A versão final será a ultima versão do projeto proposto.