UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO.

AVALIAÇÃO DE PLATAFORMAS PARA O DESENVOLVIMENTO DE WEB SERVICES.

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação

Carlos Alberto Furtado
Maurício Botelho.
Acadêmicos.

FLORIANÓPOLIS-SC

2005/1.

AGRADECIMENTOS.

À Deus, por iluminar nosso caminho e nos guiar na direção correta, permitindo que pudéssemos ultrapassar todos os obstáculos que se puseram diante de nós.

À nossas famílias, pelo apoio emocional oferecido, que nos permitiu o equilíbrio e a tranqüilidade necessária, e pelo grande amor que sempre nos deram.

À nossas esposas e filhos, pela sua paciência, amor e, sobretudo, compreensão nos momentos que mais necessitamos.

Ao nosso Orientador, Prof. João Bosco Sobral por ter depositado sua confiança em nosso esforço e pela sua competência na condução de nosso trabalho.

Aos membros de nossa banca, Professores José Leomoar Todesco e Frank Augusto Siqueira, por sua participação e preciosas sugestões a nosso trabalho.

A todos os amigos que, de alguma forma, estiveram ao nosso lado, nos ajudando e dando força para a realização deste trabalho.

A todos que colaboraram, direta ou indiretamente, para a realização e concretização deste trabalho.

RESUMO.

FURTADO, Carlos Alberto; BOTELHO, Maurício. **Avaliação de Plataformas Para Desenvolvimento de Web Services.** 2005. 198 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação). Curso de Sistemas de Informação, Universidade Federal de Santa Catarina, Florianópolis.

Este trabalho teve como objetivo executar a avaliação de algumas plataformas disponíveis para o desenvolvimento de serviços web através da elaboração de uma aplicação simples a qual propiciasse um resumo de cada um dos produtos e oferecesse uma idéia de como é o desenvolvimento com cada uma deles.

Nosso propósito foi apresentar algumas plataformas de desenvolvimento, fornecendo alguns parâmetros e métricas que pudessem ser considerados na escolha e demonstrar algumas vantagens e desvantagens de cada uma das plataformas avaliadas.

Palavras Chave: ETTK, SOAP, WSDL, JWSDP, GLUE, AXIS.

ABSTRACT.

FURTADO, Carlos Alberto; BOTELHO, Maurício. **Evaluation of Plataforms For Web Services Development.** 2005. 198 f. Course Conclusion Work (Bachelor's degree in Information Systems). Information System Course, Federal University of Santa Catarina, Florianópolis.

This academic work had as objective to execute an evaluation of some available platforms for the development of "web services" through the elaboration of a simple application that could provide a summary of each one of the products and offered an idea of how the development is with each one of them. Our purpose was to introduce some development platforms, supplying some indexes and parameters that could be considered in the choice, and to demonstrate some advantages and disadvantages of each one of the evaluated platforms.

Key Words: ETTK, SOAP, WSDL, JWSDP, GLUE, AXIS.

SUMÁRIO.

LISTA DE SIGLAS E ABREVIATURAS. 9
ÍNDICE DE QUADROS. 10
ÍNDICE DE ANEXOS. 11
ÍNDICE DE FIGURAS. 12
1 APRESENTAÇÃO. 15
1.1 ENTIDADE. 15
1.2 TÍTULO. 15
1.3 AUTORES. 15
1.4 ORIENTADOR/COORDENADOR. 15
1.5 BANCA AVALIADORA. 15
2 INTRODUÇÃO. 16
3 TEMA. 17
4 ESCOPO. 20
5 OBJETIVOS. 21
5.1 OBJETIVO GERAL. 21
5.2 OBJETIVO ESPECÍFICO. 21
6 MOTIVAÇÃO. 22
7 OBJETO. 23
7.1 PROBLEMA. 23
8 UMA VISÃO GERAL SOBRE SERVIÇOS WEB. 24
8.1 INTRODUÇÃO. 24
8.2 A PLATAFORMA DE SERVIÇOS WEB. 25
8.3 SOAP. 26
8.4 UDDI (UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION SERVICE). 26
8.5 WSDL (WEB SERVICES DEFINITION LANGUAGE). 28
9 SEGURANÇA EM WEB SERVICES. 31
9.1 USANDO SSL. 32
9.2 XML DSIG 33
10 AXIS. 35
10.1 ARQUITETURA. 35
10.2 SUBSISTEMAS. 39
10.3 SUBSISTEMA DE FLUXO DE MENSAGENS. 40
10.3.1 Manipuladores e cadeias. 40
10.3.2 Cadeias alvo. 43
10.3.3 Contexto de mensagens. 44
10.3.4 Mecanismo. 44
10.3.5 Configuração do mecanismo. 45
10.4 SUBSISTEMA DE ADMINISTRAÇÃO. 46
10.4.1 Administração baseada em WSDD. 47

10.5 SUBSISTEMA DE MODELO DE MENSAGEM. 47
10.5.1 Modelo de Mensagem SOAP. *47*
10.5.2 Elementos da mensagem. *48*
10.5.3 Deserialization. *49*
10.6 SUBSISTEMA DE CODIFICAÇÃO. 51
10.7 SUBSISTEMA DE FERRAMENTAS WSDL. 55
10.7.1 WSDL2Java e Java2WSDL. *55*

11 JWSDP 56

11.1 INTRODUÇÃO. 56
11.2 HISTÓRIA DO JWSDP. 57
11.3 JAXP. 58
11.4 JAXM. 59
11.5 JAX-RPC. 60
11.6 JAXR. 61
11.7 JSSE. 62
11.8 JSTL. 63
11.9 ANT E TOMCAT. 63
11.10 CONCLUSÃO. 64

12 ETTK. 66

12.1 INTRODUÇÃO. 66
12.2 FUNCIONAMENTO. 66
12.3 CONTEÚDO DO ETTK. 67
12.4 TECNOLOGIAS EXISTENTES NO ETTK. 68

13 GLUE. 71

13.1 INTRODUÇÃO. 71
13.2 EDIÇÕES 72
13.2.1 Standard. *72*
13.2.2 Professional. *73*
13.2.3 Enterprise. *73*
13.2.4 webMethod Fabric™ *74*
13.3 CONNECTION POOLING. 75
13.3.1 Conexões de Entrada (Inbound Connections). *75*
13.3.2 Conexões de Saída (Outbound Conections). *75*
13.3.3 Keep Alive. *75*
13.3.4 Proxies. *76*
13.4 TIPOS DE SERVIÇOS. 76
13.5 CAMADAS DE TRANSPORTE. 77
13.6 REGISTROS. 78
13.7 CONCLUSÃO. 79

14 PLATAFORMA AXIS. 81

14.1 INSTALAÇÃO. 81
14.2 CONFIGURANDO DO SERVIDOR WEB. 81
14.3 BIBLIOTECAS. 82
14.4 VERIFICAÇÃO DA INSTALAÇÃO. 83
14.4.1 Página Inicial. *83*
14.4.2 Validação da instalação. *84*
14.5 DESENVOLVIMENTO DE UM NOVO WEB SERVICE. 88

14.5.1 Variáveis de Ambiente no Cliente. 89
14.6 ESTUDO DE CASO CALCULADORA. 90
14.6.1 Criação e Publicação do Serviço Web. 90
14.7 TESTE DA APLICAÇÃO. 91
14.8 CRIAÇÃO DA APLICAÇÃO CLIENTE. 92

15 PLATAFORMA DE DESENVOLVIMENTO DE WEB SERVICE – GLUE. 100

15.1 INSTALAÇÃO. 100
15.2 BIBLIOTECAS. 101
15.3 CONFIGURANDO DO SERVIDOR WEB. 102
15.4 VERIFICAÇÃO DA INSTALAÇÃO. 104
15.4.1 Página Inicial. 104
15.4.2 Validação da instalação. 105
15.5 DESENVOLVIMENTO DE UM NOVO WEB SERVICE. 106
15.5.1 Variáveis de Ambiente no Cliente. 106
15.6 ESTUDO DE CASO DE UMA CALCULADORA. 107
15.6.1 Criação e Publicação do Serviço Web. 107

16 PLATAFORMA JWSDP 113

16.1 INSTALAÇÃO 113
16.2 CONTAINERS JWSDP 114
16.2.1 Sun Java System Application Server Edition 8. 114
16.2.2 Tomcat 5.0 - JWSDP 116
16.3 ESTUDO DE CASO CALCULADORA. 117
16.3.1 Registro do Servidor Web. 117
16.3.2 Criando um Novo Projeto Para a Aplicação Web. 118
16.3.3 Definição das Bibliotecas. 118
16.3.4 Criação do Web Service. 119
16.3.5 Definindo os Métodos do Serviço Web. 119
16.3.6 Geração e Configuração de Um Manipulador de Mensagens SOAP. 124
16.3.7 Tornando o Serviço Web Disponível. 125
16.3.8 Deploy do Serviço Web. 125
16.3.9 Registrando e Testando o Servidor Web. 126
16.3.10 Utilizando o Web Service. 128
16.3.11 Descobrindo Informações do Web Service. 128
16.3.12 Criando o Cliente do Web Service. 129
16.3.13 "Empacotando" a Aplicação. 130
16.3.14 Criando uma Aplicação J2EE. 131
16.3.15 Deploy da Aplicação J2EE. 131

17 PLATAFORMA EMERGING TECHNOLOGIES TOOLKIT (ETTK) 133

17.1 INSTALAÇÃO. 133
17.1.1 ETTK 133
17.1.2 RAD 134
17.2 CRIANDO UM NOVO WEB SERVICE 135
17.2.1 Um novo projeto 135
17.3 CRIANDO O SERVIDOR ETTK 141
17.4 ADICIONANDO AS CLASSES 145
17.5 CRIANDO O WEB SERVICE 148

18 CONCLUSÕES E TRABALHOS FUTUROS. 158

18.1 CONCLUSÕES. 158
18.2 TRABALHOS FUTUROS. 164
19 BIBLIOGRAFIA. 165

LISTA DE SIGLAS E ABREVIATURAS.

API	*Application Program Interface*
EAR	*Externar Access Register*
HTTP	*Hipertex Transfer Protocol*
JSP	*Java Server Pages*
JVM	*Java Virtual Machine*
JWSDP	*Java Web Services Developer Pack*
RAD	*Rational Software Development Plataform*
SJS	*Sun Java System*
SOAP	*Simple Object Access Protocol*
TCK	*Text Compatibility Kit*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
WSDD	*Web Service Deployment Descriptor*
WSDL	*Web Service Description Language*
XML	*Extensible Markup Language*

ÍNDICE DE QUADROS.

QUADRO 1 - CÓDIGO DA CLASSE CALCULADORA.JAVA – AXIS. 90
QUADRO 2 - RESULTADO DA EXECUÇÃO DA CLASSE CALCULADORA – AXIS. 92
QUADRO 3 - CÓDIGO DA CLASSE CLIENTE.JAVA – AXIS. 94
QUADRO 4 - COMANDO WSDL2JAVA. 95
QUADRO 5 - ESPECIFICAÇÃO XML DA IMPLEMENTAÇÃO DA INTERFACE-AXIS. 95
QUADRO 6 - GERAÇÃO DO CÓDIGO JAVA A PARTIR DO ARQUIVO WSDL - GLUE. 107
QUADRO 7 - CÓDIGO DA CLASSE CALCULADORA.JAVA - GLUE. 108
QUADRO 8 - CÓDIGO DA INTERFACE ICALCULADORA.JAVA - GLUE. 109
QUADRO 9 - CÓDIGO DA CLASSE CALCSERVER - GLUE. 109
QUADRO 10 - INICIALIZAÇÃO DO SERVIDOR - GLUE. 110
QUADRO 11 - INVOCAÇÃO DO WEB SERVICE PELO CLIENTE - GLUE. 111
QUADRO 12 - TESTE DA CLASSE CALCCLIENT1 - GLUE. 112
QUADRO 13 - CÓDIGO DA CLASSE INTERFACE - JWSDP. 120
QUADRO 14 - CÓDIGO DA CLASSE CALCULADORAIMPL.JAVA GERADO PELA IDE - JWSDP. 121
QUADRO 15 - CÓDIGO DA CLASSE CALCULADORAWSIMPL.JAVA ALTERADO - JWSDP. 123
QUADRO 16 - MENSAGEM DE ERRO NO DEPLOY DO SERVIÇO WEB - JWSDP. 125
QUADRO 17 - LOG DO SERVIDOR SJS - JWSDP. 127
QUADRO 18 - CÓDIGO DO MÉTODO "PROCESSREQUEST" DO SERVLET GERADO PELA IDE - JWSDP. 130
QUADRO 19 - CÓDIGO ALTERADO DO MÉTODO "PROCESSREQUEST" DO SERVLET - JWSDP. 130
QUADRO 20 - MENSAGEM DE ERRO NO DEPLOY DA APLICAÇÃO J2EE - JWSDP. 132
QUADRO 21 - CÓDIGO DA CLASSE CALCULADORA - ETTK. 147
QUADRO 22 - CÓDIGO DA CLASSE CALCCLIENTE - ETTK. 147
QUADRO 23 - COMPLEXIDADE DA INSTALAÇÃO. 162
QUADRO 24 - QUALIDADE DA DOCUMENTAÇÃO. 162
QUADRO 25 - DIFICULDADE DE APRENDIZADO. 163
QUADRO 26 - VELOCIDADE DE DESENVOLVIMENTO. 163
QUADRO 27 - CÓDIGO DE INICIALIZAÇÃO DAS VARIÁVEIS DE AMBIENTE AXIS. 168
QUADRO 28 - CÓDIGO DO PROXY - ETTK. 173
QUADRO 29 - CÓDIGO DO ARQUIVO WSDL - ETTK. 176

ÍNDICE DE ANEXOS.

ANEXO 1 – CONFIGURAÇÃO DAS VARIÁVEIS DE AMBIENTE DA PLATAFORMA AXIS..... 166
ANEXO 2 – CONFIGURAÇÃO DAS VARIÁVEIS DE AMBIENTE DA PLATAFORMA GLUE..... 170
ANEXO 3 – PROXY ETTK..... 173
ANEXO 4 - ARQUIVO WSDL – ETTK. 176
ANEXO 5 - ARTIGO..... 180

ÍNDICE DE FIGURAS.

FIGURA 1 - VISÃO GERAL DA ESTRUTURA DE IMPLEMENTAÇÃO DE WEB SERVICE. 25
FIGURA 2 - MECANISMO DO AXIS – SERVIDOR. 36
FIGURA 3 - MECANISMO AXIS – CLIENTE 38
FIGURA 4 – SUBSISTEMAS 39
FIGURA 5 - FLUXO DE MENSAGENS. 41
FIGURA 6 – CONJUNTO DE MANIPULADORES. 42
FIGURA 7 – TARGETED CHAIN 43
FIGURA 8 - CONTEXTO DE MENSAGENS. 44
FIGURA 9 – RELACIONAMENTO AXISCLIENT E AXIASERVER. 45
FIGURA 10 - SUBSISTEMA DE ADMINISTRAÇÃO. 46
FIGURA 11 - ADMINISTRAÇÃO BASEADA EM WSDD. 47
FIGURA 12 - MENSAGEM SOAP. 48
FIGURA 13 – ELEMENTOS DAS MENSAGENS. 49
FIGURA 14 - MANIPULADORES SAX 50
FIGURA 15 - EMPILHAMENTO DE INSTÂNCIAS. 50
FIGURA 16 - CLASSES SERIALIZERS E DESERIALIZERS. 51
FIGURA 17 - MECANISMO DE PROCESSAMENTO XML. 52
FIGURA 18 - TYPE MAPPING. 53
FIGURA 19 – CODIFICAÇÃO DO TYPE MAPPING. 54
FIGURA 20 - ARQUITETURA DA PLATAFORMA WEBMETHODS GLUE 71
FIGURA 21 - ESTRUTURA DA EDIÇÃO ENTERPRISE. 74
FIGURA 22 - RELACIONAMENTO ENTRE REGISTRO E ISERVER. 78
FIGURA 23 – ESTRUTURA DE DIRETÓRIOS DO SERVIDOR TOMCAT-AXIS. 82
FIGURA 24 – DIRETÓRIO DE BIBLIOTECAS DO SERVIDOR WEB. 83
FIGURA 25 – PÁGINA INICIAL DA PLATAFORMA AXIS. 84
FIGURA 26 – AXIS HAPPINES PAGE. 85
FIGURA 27 – OPTIONAL COMPONENTS. 86
FIGURA 28 – OUTROS SERVIÇOS. 87
FIGURA 29 – ESPECIFICAÇÃO DO ENVELOPE SOAP. 88
FIGURA 30 – ESTRUTURA DE DIRETÓRIOS DA PLATAFORMA AXIS. 89
FIGURA 31 – ESTRUTURA DE DIRETÓRIOS DO SERVIDOR TOMCAT DO LADO CLIENTE 93
FIGURA 32 – DIRETÓRIO DE BIBLIOTECAS DO GLUE. 101
FIGURA 33 – ESTRUTURA DE DIRETÓRIOS DO SERVIDOR TOMCAT-GLUE. 103
FIGURA 34 – PÁGINA INICIAL DA PLATAFORMA GLUE. 104
FIGURA 35 105
FIGURA 36 106
FIGURA 37 – INSTALAÇÃO DO SERVIDOR SJS. 114

FIGURA 38 – PARÂMETROS DE INSTALAÇÃO DO SJS. 115
FIGURA 39 – ESTRUTURA DE DIRETÓRIOS DO SERVIDOR TOMCAT-JWSDP. 116
FIGURA 40 – NETBEANS 4.1RC2. 117
FIGURA 41 – PROJETO CALCULADORAWS. 118
FIGURA 42 – MANIPULADOR DE MENSAGENS SOAP. 125
FIGURA 43 126
FIGURA 44 - URL WSDL. 126
FIGURA 45 - TESTE DE OPERAÇÃO DO WEB SERVICE. 127
FIGURA 46 - TESTE DE OPERAÇÃO DO WEB SERVICE CLIENTE. 129
FIGURA 47 - APLICAÇÃO J2EE. 131
FIGURA 48 - CRIAÇÃO DE NOVO PROJETO - ETTK. 136
FIGURA 49 - DEFINIÇÃO DO TIPO DE APLICAÇÃO - ETTK. 137
FIGURA 50 - DEFINIÇÃO DO PROJETO EAR - ETTK. 138
FIGURA 51 - DISPLAY DOS RECURSOS DO PROJETO - ETTK. 139
FIGURA 52 - DEFINIÇÃO DO GABARITO DA INTERFACE DA APLICAÇÃO WEB - ETTK. 140
FIGURA 53 - DEFINIÇÃO DAS BIBLIOTECAS - ETTK. 141
FIGURA 54 - INICIANDO O SERVIDOR WEB - ETTK. 142
FIGURA 55 - INCLUSÃO DO SERVIDOR WEB AO PROJETO - ETTK. 142
FIGURA 56 - DEFINIÇÃO DO SERVIDOR WEB - ETTK. 143
FIGURA 57 - DEFINIÇÃO DO DIRETÓRIO DO NOVO SERVIDOR - ETTK. 144
FIGURA 58 - INICIANDO O SERVIDOR WEB ATRAVÉS DO RAD - ETTK. 144
FIGURA 59 – DEFINIÇÃO DA PERSPECTIVA DAS CLASSES DA APLICAÇÃO WEB - ETTK. 145
FIGURA 60 - CRIAÇÃO DAS CLASSES DA APLICAÇÃO WEB - ETTK. 146
FIGURA 61 - CRIAÇÃO DO WEB SERVICE - ETTK. 149
FIGURA 62 - SELEÇÃO DOS OBJETO DO WEB SERVICE - ETTK. 150
FIGURA 63 - CONFIGURAÇÃO DA IMPLEMENTAÇÃO DO WEB SERVICE - ETTK. 151
FIGURA 64 - SELEÇÃO DA INTERFACE DO NÓ DE EXTREMIDADE DO WEB SERVICE - ETTK. 152
FIGURA 65 - CRIAÇÃO DO ARQUIVO WSDL - ETTK. 153
FIGURA 66 - GERAÇÃO DO PROXY DO WEB SERVICE - ETTK. 154
FIGURA 67 - TESTE DO CLIENTE DO WEB SERVICE - ETTK 155
FIGURA 68 - PUBLICAÇÃO DO WEB SERVICE - ETTK. 156
FIGURA 69 - TESTE DO WEB SERVICE - ETTK. 157
FIGURA 70 – PROPRIEDADES DO SISTEMA. 166
FIGURA 71 – AXIS_HOME. 167
FIGURA 72 – AXIS_LIB. 167
FIGURA 73 – AXISCLASSPATH. 167
FIGURA 74 – VARIÁVEIS DE AMBIENTE. 168
FIGURA 75 170
FIGURA 76 171
FIGURA 77 171

FIGURA 78 172
FIGURA 79 172

1 APRESENTAÇÃO.

1.1 Entidade.

Universidade Federal de Santa Catarina

Curso de Sistemas de Informação

1.2 Título.

Avaliação de Plataformas para o Desenvolvimento de Web Services.

1.3 Autores.

Carlos Alberto Furtado;

Maurício Botelho.

1.4 Orientador/Coordenador.

Prof. João Bosco Mangueira Sobral, Doutor.

1.5 Banca Avaliadora.

Prof. Frank Augusto Siqueira, Doutor.

Prof. José Leomar Todesco, Doutor.

2 INTRODUÇÃO.

O presente documento tem por objetivo implementar as metas definidas na proposta de projeto de conclusão de final do curso de Sistemas de Informação, apresentado no semestre 2004/1. Nele serão descritos os objetivos gerais e específicos do trabalho proposto como também parte da fundamentação teórica do trabalho, como sugerido na proposta inicial.

Será apresentada uma visão geral sobre os serviços web, bem como uma descrição das plataformas em estudo.

Este documento é, portanto, um rascunho inicial do trabalho final, que irá apresentar parte da fundamentação teórica do estudo bem como a bibliografia pesquisada.

3 TEMA.

A maioria das aplicações distribuídas e dos sistemas distribuídos é baseada no paradigma cliente/servidor, aonde o servidor anuncia um conjunto de serviços ao qual ele provê acesso para alguns recursos. O código que executa estes serviços é hospedado localmente no servidor que executa cada serviço e pode utilizar a capacidade de seu processador, ou seja, o servidor possui os recursos o know-how e o processador.

Se o cliente quiser usar algum recurso oferecido pelo servidor o cliente escolhe um ou mais serviços oferecidos pelo servidor.

A arquitetura cliente/servidor pode ser classificada como: cliente/servidor com processos, cliente/servidor com objetos distribuídos (RMI, CORBA, DCOM,...), cliente/servidor com objetos para web (XML, RPC/XML, WSDL, SOAP/XML, UDDI).

A estrutura cliente/servidor, com objetos distribuídos, pospõe limitações para funcionar na web. A web foi concebida para tirar vantagem da Internet e a computação distribuída como foi concebida, não foi pensado para a web.

A web é basicamente desconectada, isto é, as conexões são temporárias.
Os serviços de computação distribuída, como segurança e transações, que necessitam de conexões ao nível de transporte, precisam ser projetadas para oferecer tais funcionalidades para as características da web desconectada.

Na web as partes podem se conectar sem conhecimento uma da outra, seguindo os links URL e observando poucas regras básicas. Nas tecnologias de computação distribuída o acoplamento cliente/servidor se dá de forma mais firme e não pode, portanto, tirar vantagem da web hoje existente.

Na computação distribuída convencional, as partes precisam conhecer as outras seguindo as referências de objetos-servidores passadas por esses para o

lado cliente, ou providas por um servidor de nomes remoto (que não está no lado cliente), o qual fornece referências, e através dos quais, objetos de aplicação são localizados e podem então ser acessados.

Os Web Services adotam o modelo de publicação na web, sendo então possível um ponto específico (endereço do serviço), usando uma definição de interface de serviços para a web, não sendo necessário um tipo específico de cliente para acessar este serviço.

Para os Web Services qualquer cliente pode acessar qualquer serviço disponível na web, desde que as informações sobre o serviço estejam disponíveis e sejam compreensíveis e possam ser geradas mensagens, através de processadores XML.

Este paradigma possui a vantagem de permitir a integração cooperativa de aplicações. Os clientes podem desenvolver e integrar aplicações posteriormente.

A próxima geração da web é voltada para a busca de informações exatas sobre strings de texto embutidas em páginas web. Soluções para a comunicação entre aplicações devem derivar das tecnologias atuais da Internet. Os pontos finais da web, os endereços URL, irão proporcionar serviços para o processamento de dados XML, da mesma maneira que os browser atuais processam texto html.

Os serviços web devem fazer referência dinâmica a URL's e também ler e gerar esquemas XML automaticamente.

Grandes empresas já se dedicam a desenvolver os tipos de serviços padrões que são acessíveis para qualquer programa. O objetivo é a integração de aplicações que utilizam serviços pré-definidos na web.

Para isto é necessária uma padronização no desenvolvimento destes serviços. Já existem iniciativas, tais como do W3C para web.

Os web services necessitam de diversas tecnologias baseadas em XML, tais como:

- XML (extended markup language) – Representa uma série de especificações publicadas e mantidas pelo W3C;
- WSDL (web services description language) – Tecnologia baseada em XML para especificação de interfaces de web services, dados e tipos de mensagens, modelos de interação e mapeamento de protocolos;
- SOAP (simple object access protocol) – Tecnologias baseadas em XML que define um envelope para a comunicação de serviços na web;
- UDDI (universal description, Discovery and integration) – Permite o registro e descoberta de serviços na web.

4 ESCOPO.

Tanto o avanço da Internet como a utilização de web services, fez com que várias ferramentas e plataformas de desenvolvimento destes serviços surgissem no mercado.

O projeto proposto visa analisar algumas ferramentas para o desenvolvimento de web services, estabelecendo um comparativo entre elas.

As ferramentas e plataformas analisadas serão as seguintes:

- ETTK (Emerging Techlogies ToolKit – IBM);
- AXIS – Projeto Jakarta (Apache);
- GLUE – The Mind Eletric;
- JWSDP – Java Web Services Developer Pack.

5 OBJETIVOS.

5.1 Objetivo Geral.

Avaliar várias plataformas de desenvolvimento de web services, através de parâmetros pré definidos.

5.2 Objetivo Específico.

Criar uma metodologia para uma análise comparativa das plataformas citadas, avaliando como principais aspectos:

- . Desempenho;
- Desenvolvimento de aplicações específicas tais como B2B;
- . Utilização;
- . Instalação;
- . Ambiente de Desenvolvimento.

6 MOTIVAÇÃO.

A necessidade, cada vez maior, na busca de soluções rápidas para o desenvolvimento de web services, nem sempre permitem que sejam escolhidas as ferramentas e plataformas mais apropriadas para sua implementação.

Um dos objetivos deste projeto é proporcionar medidas de comparação entre as plataformas, que possam auxiliar os desenvolvedores no momento da escolha de uma plataforma.

7 OBJETO.

7.1 Problema.

Como as plataformas de desenvolvimento de web services são fornecidas por diversas empresas, nem sempre é uma tarefa fácil fazer uma escolha sobre qual utilizar.

O problema principal está focado na falta de métricas de comparação entre elas, tornado a escolha, muitas vezes, baseada em parâmetros abstratos ou também na experiência de utilização dos desenvolvedores, quando são ignoradas as características benéficas e as vantagens técnicas de cada plataforma.

8 UMA VISÃO GERAL SOBRE SERVIÇOS WEB.

8.1 Introdução.

Nos dias de hoje seria difícil imaginar a computação em rede sem o uso da internet. O sucesso da difusão da internet se baseia em dois pontos básicos: sua simplicidade e presença em todo o mundo. Sob a visão de um fornecedor de serviços, com a criação de uma página na web ele pode ser visto em todo o mundo. Do ponto de vista do usuário, basta possuir um computador conectado à web para acessar uma vasta gama de serviços com facilidade.

Pelo ponto de vista de uma API de serviços todo o trabalho executado na internet pode ser feito utilizando 3 métodos (GET, POST e PUT) junto com uma linguagem de marcação. O aumento do uso de serviços web baseia-se no fato de que a internet vai além do fornecimento de informações, mas se estende no fornecimento de serviços.

Por serviços web pode-se entender a união e reutilização de vários serviços que estão disponíveis na internet que executam tarefas específicas. Podemos citar como exemplo um site de comparação de preços de produtos que se utiliza dos serviços web de fornecedores, obtendo informações de suas bases de dados corporativas e concentrando o resultado em uma única página.

Os serviços web são uma nova geração de aplicações. São aplicações modulares, independentes e auto-explicativas, as quais podem ser publicadas, localizadas e invocadas através da internet. Os serviços web executam funções que podem ir de uma simples até complexos processos de negócios. Uma vez que um serviço web é distribuído, outras aplicações (e outros serviços web) podem encontrá-lo, invocá-lo e utilizar seus recursos.

Uma das questões levantadas sobre os serviços web, é porque preocupar-se em desenvolver serviços para web se existem outras plataformas de objetos distribuídos (RMI, Jini, CORBA, DCOM, etc.) A grande vantagem da web está em sua ampla divulgação e distribuição e facilidade de acesso. Além disso, a internet fornece uma interface padrão e amplamente conhecida de acesso a estes serviços, através dos navegadores web.

Observada do ponto de vista de sua arquitetura, um serviço web é implementado de forma semelhante às aplicações tradicionais:

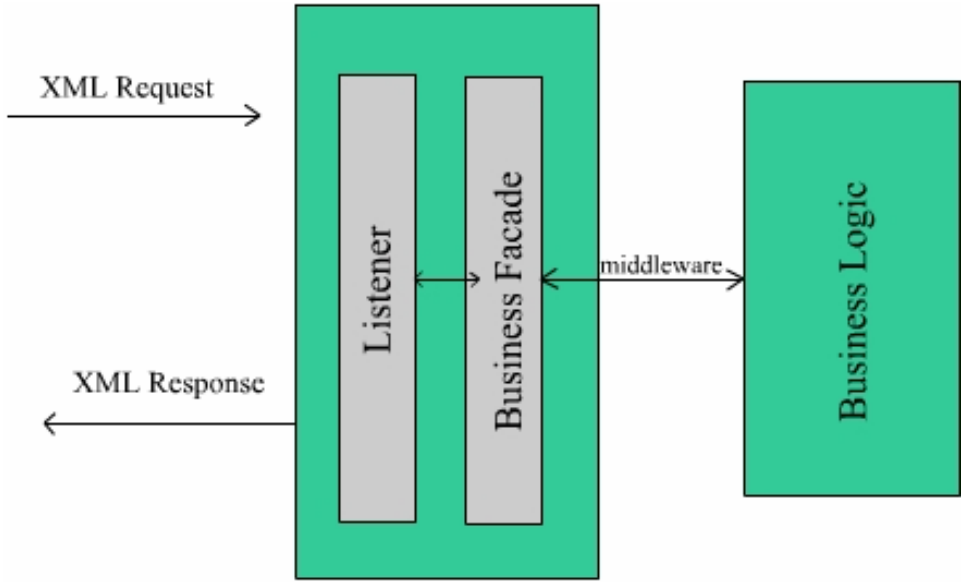


Figura 1 - Visão Geral da Estrutura de Implementação de Web Service.

8.2 A plataforma de Serviços Web.

A plataforma básica dos serviços web é composta pelo XML mais o protocolo HTTP. O protocolo HTTP se encontra em qualquer lugar, executando praticamente tudo na internet. O XML fornece uma meta linguagem na qual podemos escrever códigos especializados para representar iterações complexas entre clientes e serviços ou entre componentes que formam um serviço. Por trás da fachada de um servidor web, uma mensagem XML é convertida para uma requisição que conhecida pelo servidor e o resultado convertido novamente para XML.

É claro que o processo de requisição e resposta vai além do procedimento descrito. A plataforma de serviços web completa envolve outras tecnologias e procedimentos, tais como:

- SOAP – Chamadas Remotas;
- WSDL – Definição das características do serviço;
- UDDI. – Serviço de publicação e procura de serviços.

8.3 SOAP.

O SOAP é a especificação de um protocolo que define uma maneira para a passagem de dados codificados em XML (Ethan, 2002). Também define maneiras para facilitar a chamada remota de funções via Internet, permitindo que dois programas se comuniquem de forma muito semelhante à invocação de páginas Web. Submetido em 2003 ao W3C pela IBM, Microsoft, Userland e DevelopMentor, o futuro do desenvolvimento do SOAP agora esta entregue aos cuidados do grupo de trabalho do protocolo XML do W3C (W3C's Protocols Working Group). Efetivamente, isto significa que a atual versão do SOAP está parada e estável até que o grupo de trabalho do W3C forneça uma nova especificação.

8.4 UDDI (Universal Description, Discovery and Integration Service).

O UDDI fornece um mecanismo para a descoberta dinâmica de serviços web. Utilizando a interface do UDDI, negócios podem se conectar dinamicamente a outros serviços fornecidos por parceiros externos. Um registro UDDI é similar a um registro CORBA, ou pode ser imaginado com um serviço DNS para uma aplicação de negócios. Um registro UDDI possui dois tipos de clientes: negócios que querem publicar um serviço (e sua interface), e clientes que querem obter serviços de certo

tipo. O UDDI é desenvolvido sobre o SOAP e assume que as requisições e respostas são objetos UDDI enviados como mensagens SOAP. Abaixo é oferecida uma visão geral do que é fornecido pelo UDDI.

Tabela 1 - Serviços UDDI. (Ethan, 2002, p. 135-136).

Informação	**Operação**	**Informação Detalhada** (suportada por API de baixo nível)
Páginas Brancas: Fornece o nome, endereço, telefone e outras informações de contato de um determinado negócio.	**Publicação:** Como um fornecedor de serviço registra suas informações.	**Informação do negócio:** Armazenada em um objeto *BusinessEntity*, o qual contém informações sobre os serviços, categorias, contatos, URLs e outros dados necessários para interagir com a empresa.
Páginas amarelas: Informação que determina a categoria do negócio.	**Descoberta:** Como uma aplicação descobre um serviço web em particular.	**Informação do Serviço:** Descreve um grupo de serviços web. Estes estão contidos em um objeto *BusinessService.*
Páginas verdes: Informações técnicas sobre um serviço web.	**Conexão:** Como uma aplicação conecta e interage com um serviço web após sua	**Informação de conexão:** Os detalhes técnicos necessários para invocar um serviço web. Isto

	descoberta.	inclui URLs, informações sobre os nomes dos métodos, tipos de argumentos e assim por diante. Estes dados são representados no objeto *BindingTemplate.* **Detalhes de Especificação de Serviço:** São os meta dados sobre as várias especificações implementadas por um serviço web. São chamados *tModels* na especificação UDDI.

Fonte: Adaptado de Ethan (2002).

8.5 WSDL (Web Services Definition Language).

O WSDL fornece uma maneira para que os provedores de serviços descrevam o formato básico dos serviços requisitados sobre diferentes protocolos e codificações (Ethan, 2002). O WSDL é usado para descrever o que um serviço web é capaz de fazer, onde ele está hospedado e como invocá-lo. Enquanto a

declaração do SOAP/HTTP pode ser feita utilizando várias especificações, o WSDL faz mais sentido por assumir SOAP/HTTP/MIME como mecanismo de invocação de objetos remotos. Os registros UDDI descrevem vários aspectos dos serviços web, incluindo detalhes obrigatórios do serviço. O WSDL se ajusta como um subsistema da descrição de serviço UDDI.

O WSDL define os serviços com coleções de pontos finais ou portas na rede. Em WSDL a definição abstrata de pontos finais e mensagens são separadas pela distribuição concreta da rede ou pelo formato obrigatório dos dados. Isto permite a reutilização da definição abstrata de mensagens, as quais são descrições abstratas dos dados que estão sendo trocados, e dos tipos de portas, as quais são coleções abstratas de operações. O protocolo concreto e especificação do formato dos dados constituem uma ligação reutilizável. Uma porta é definida pela associação de um endereço na rede com uma ligação reutilizável; uma coleção de porta define um serviço. E, assim, um documento WSDL utiliza os seguintes elementos para a definição de serviços:

- Tipos – um repositório para definição de tipos de dados utilizando alguns tipos de sistemas (tal como XSD);
- Mensagem – Um tipo abstrato de definição dos dados que estão sendo comunicados;
- Operação – Uma descrição abstrata de uma ação suportada pelo serviço;
- Tipo de Porta – Um conjunto abstrato de operações suportadas por um ou mais pontos finais da rede;
- Ligação – Um protocolo concreto e especificação de formato de dado para um tipo particular de porta;

- Porta – Um ponto final simples definido como uma combinação de uma ligação e um endereço na rede;
- Serviço – Uma coleção de pontos finais relacionados.

Em resumo, o WSDL é um modelo para mostrar como os serviços podem ser descritos e conectados pelos clientes.

9 SEGURANÇA EM WEB SERVICES.

É opinião comum que os web services só poderão ser considerados bem sucedidos quando estiverem equacionadas e consolidadas todas as questões relativas à segurança.

Dentro do contexto de segurança, podemos definir um web service seguro, quando satisfizer as seguintes condições:

- Verificação da integridade de uma mensagem, isto é, não houver alteração entre a fonte emissora e receptora, e vice-versa;
- Uma mensagem classificada como confidencial possa ser transmitida, sem que terceiros possam interceptá-la e decodificá-la;
- Determinar e autenticar a identidade do remetente;
- Determinar se o remetente tem autorização para efetuar a operação solicitada (implicitamente ou explicitamente) pela mensagem.

Em um ambiente distribuído, estas exigências são satisfeitas com a utilização de criptografia, com o uso de chaves para a codificação e decodificação de mensagens.

Contudo existem outros métodos de segurança que podem ser utilizados no âmbito dos serviços web, como por exemplo:

- Assinatura Digital XML, um método desenvolvido pelo pela W3C/IETF hospedado pelo W3C. Esta iniciativa está essencialmente consolidada conseguindo definir com relativo sucesso uma especificação para assinatura digital usando XML;
- XML Encriptyon, uma iniciativa W3C que defini como codificar um documento XML;

- Gerenciamento de Chave XML, uma especificação que permite aos clientes obterem informações de criptografia (chaves, certificados, etc...) e efetuar o gerenciamento destas chaves como registros, revogações, etc.
- OASIS Security Service TC, que é conhecido como SAML (Security Authorization Markup Language). SALM é um framework para a troca de informações de identificação;
- OASIS Accesss Control Markup Language TC, é uma framework para definir um conjunto de privilégios requeridos para a execução de uma operação, incluindo a informação da identidade e fatores externos (como a política e a hora do acesso).
- OASIS Digital Signature Services TC, é um novo comitê com o objetivo de definir uma interface para um serviço da geração de assinaturas e verificação de serviços.
- OASIS Web Services Security TC, é construído sobre uma especificação da IBM e da Microsoft que especifica como inserir uma assinatura dentro de uma mensagem SOAP.

9.1 Usando SSL.

O protocolo SSL atua entre as extremidades de uma comunicação, não entre aplicações. Isso funciona muito bem para a maioria das aplicações web, onde um usuário acessa um Web site através de um browser. Porém este tipo de acesso pode incorrer na corrupção ou vazamento dos dados, principalmente por falhas nos navegadores web. O provedor de acesso frequentemente tem problemas para configurar o nível de proteção para seus servidores de maneira que atendam aos muitos usuários através de várias identidades diferentes utilizando vários servidores

de certificados SSL que atendam as várias requisições de acesso, através de listeners instalados em várias portas.

Outra característica do SSL é não poder armazenar a mensagem para verificar posteriormente sua autenticidade e originalidade. Quando uma conexão é estabelecida, as partes trocam entre si uma chave de sessão para codificar a informação. As duas partes possuem a chave, mas somente para ser usada por um breve período de tempo. No caso de dúvida, não é possível a nenhuma das partes garantirem que a mensagem não foi modificada.

As características citadas, não tornam o uso de SSL como o mais adequado para a segurança em web services. Para garantir a segurança dos serviços web, é necessário um método que possa ser encapsulado no próprio envelope SOAP.

9.2 XML DSIG

Uma opção para garantir a segurança na implementação dos web services, é a utilização do padrão XML DSIG (XML Digital Signature).

O padrão XML DSIG especifica como assinar e codificar uma mensagem SOAP. O código abaixo demonstra como colocar uma assinatura XML DSIG em um cabeçalho SOAP.

```
<SOAP:Envelope xmlns:SOAP='http://schemas.xmlsoap.org/soap/envelope/'>
  <SOAP:Header>
    <wsse:Security
xmlns:wsse='http://schemas.xmlsoap.org/ws/2002/07/secext'>
      <Signature xmlns='http://www.w3.org/2000/09/xmldsig#'>
        ...defined below...
```

```
    </Signature>
   </wsse:Security>
  </SOAP:Header>
  <SOAP:Body id='Body'>
     ...message body...
  </SOAP:Body>
</SOAP:Envelope>
```

A assinatura é composta de três partes:

- Uma referência ao que está sendo assinado (podem existir múltiplas referências);
- Um valor para a assinatura que faz a conversão de valores;
- Informações sobre as chaves de criptografia.

A principal vantagem do padrão XML DSIG é o de codificar e assinar a informação com uma chave RSA, o que torna flexível e facilita a implementação de políticas de segurança para o uso de web services.

10 AXIS.

10.1 Arquitetura.

A plataforma Axis consiste basicamente de vários subsistemas funcionando conjuntamente e pretendemos a seguir dar uma visão geral de como isto funciona.

Visto de uma forma simples, Axis consiste basicamente do processamento de mensagens. Quando o processamento central do Axis inicia, uma séria de manipuladores de mensagens são invocados em ordem. A ordem em que são chamados depende de 2 fatores – a configuração em que são desenvolvidos e distribuídos e se a configuração funciona como um cliente ou servidor.

O objeto que é enviado para cada manipulador de mensagens é um “MessageContext”. Um “MessageContext” é uma estrutura que contem várias partes: a) uma mensagem de solicitação, b) uma mensagem de resposta, e c) um conjunto de propriedades.

Há basicamente duas maneiras de invocar o Axis:

1) Como um servidor, um “transport listener”, que irá criar um “MessageContext” e irá chamar o framework do Axis sempre que for solicitado;
2) Como um Cliente, neste caso uma aplicação que irá gerar um “MessageContext” e irá chamar o framework do Axis sempre que for solicitado;

Em qualquer uma das situações, o framework do Axis irá simplesmente enviar o “MessageContext” resultante através dos manipuladores de mensagens configurados, onde cada um deles irá processar as ações determinadas sobre o “MessageContext”.

A figura abaixo mostra o lado servidor. Os cilindros menores representam manipuladores de mensagens e o cilindro maior, com cilindros menores em seu interior, represente uma corrente de manipuladores de mensagens (coleção de manipuladores ordenados de forma pré-determinada).

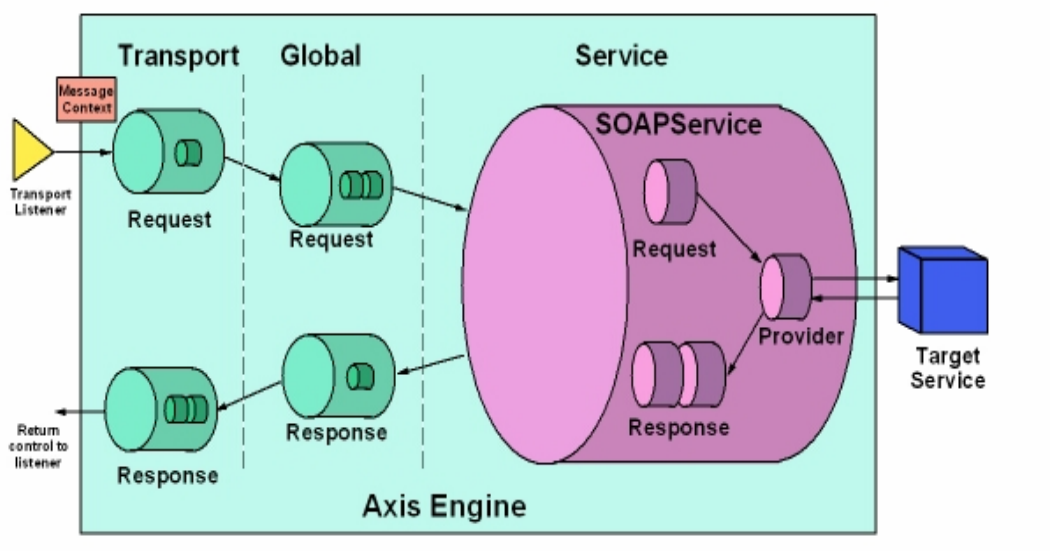


Figura 2 - Mecanismo do Axis – Servidor.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

Uma mensagem chega (em um manipulador específico de protocolo) através de um “Transport Listener”. Neste caso assumiremos que o “Listener” é um servlet http. Este é o processo do “Listener” para “empacotar” os dados de um protocolo específico em um objeto Mensagem (org.apache.axis.Message), e colocar a mensagem em um “MessageContext”. O “MessageContext” é carregado com várias propriedades pelo “Listener” – como por exemplo a propriedade “http.SOAPAction” será atribuída como valor do atributo SOAPAction do cabeçalho http. O “Transport Listener” também atribui a cadeia “transportName” ao “MessageContext”. Uma vez

que o objeto “MessageContext” está pronto para ser enviado o “Listener” o passa ao mecanismo do Axis (Gros, 2003).

O primeiro procedimento do mecanismo do Axis é procurar o transporte pelo nome. O Transporte é um objeto que contem uma cadeia “request” e uma cadeia “response” e, em alguns casos, ambas. Uma cadeia consiste é um manipulador de mensagens formado por uma seqüência de manipuladores. Se uma cadeia “request” de transporte existe, esta será chamada passando o “MessageContext” para um método invoke(). Isto resultará na chamada de todos os manipuladores especificados na cadeia “request”.

Após a manipulação do “request” do transporte, o mecanismo do Axis aloca uma cadeia “request” global e então pode invocar qualquer um dos manipuladores configurados no “request” global.

Dentro deste contexto, um dos manipuladores irá especificar o valor do campo “serviceHandler” no objeto “MessageContext” (normalmente isto é feito pelo manipulador “URLMapper” o qual mapeia uma URL como por exemplo, http://localhost:8080/axis/service/AdminService para o serviço “AdminService”). Este campo determina o manipulador que será chamado para executar uma funcionalidade específica, como por exemplo, fazer uma chamada RPC em um objeto back-end. Na plataforma Axis, serviços são comumente instancias de cadeias “request” e “response” e devem possuir um provedor, que é um manipulador de mensagens responsável pela implementação da lógica do serviço.

Para “requests” RPC, o provedor é uma classe org.apache.axis.provider.Java.RPCProvider. Também se trata de um manipulador de mensagens que, quando invocado, tenta chamar uma classe Java que é determinada pelo parâmetro “className” especificada durante o desenvolvimento. É

utilizado o padrão do SOAP RPC para a especificação do método chamado e para verificar se os tipos dos argumentos de entrada, codificados em XML, são compatíveis com os parâmetros resultantes do método chamado.

O caminho da mensagem do lado cliente é similar ao lado servidor, exceto pelo fato de a ordem de envio e recebimento de mensagens, fluírem em sentido oposto.

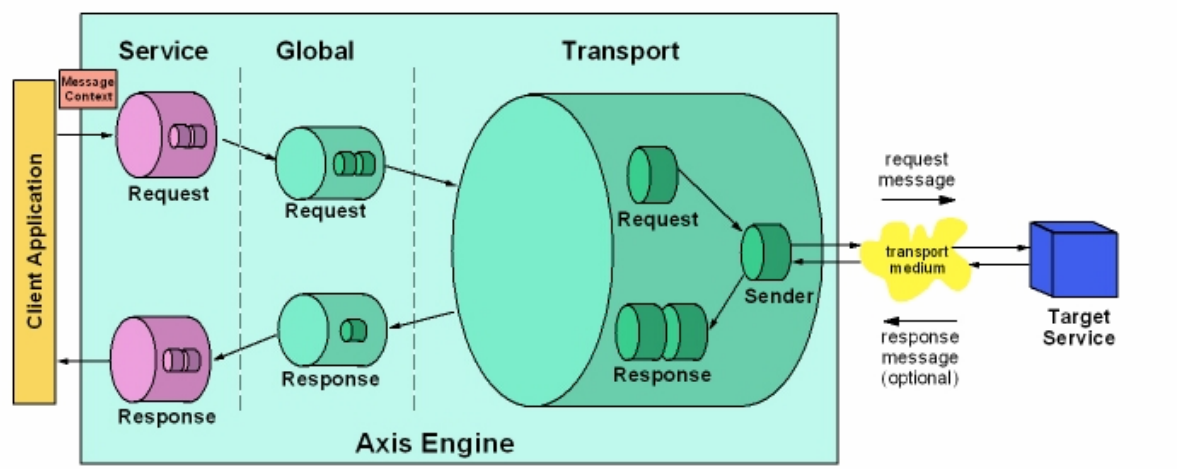


Figura 3 - Mecanismo Axis – Cliente

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

O manipulador “Service” é chamado primeiro – no lado cliente não há um provedor, uma vez que o serviço está sendo disponibilizado por um nó remoto, mas ainda há a possibilidade da existência de cadeias “request” e “response”. As cadeias “request” e “response” existentes na camada “Service” executam qualquer processamento específico na requisição da mensagem para fora do sistema, assim como na mensagem de resposta que retorna ao sistema.

Depois da cadeia “request” da camada “Service”, a cadeia “request” da camada “Global” é chamada, seguida pela camada “Transport”. O “Transport

Sender", um manipulador especial cuja função é obter as mensagens que entram e saem do servidor SOAP, independente da especificação de protocolo. A responsta ("response"), se existir, é colocada no campo "responseMessage" do objeto "MessageContext" e o objeto é propagado através da cadeia "response", passando pelas camadas "Trasnport", "Global" e "Service", nesta ordem.

10.2 Subsistemas.

A plataforma Axis engloba vários subsistemas trabalhando conjuntamente com o objetivo de separar responsabilidades e tornar a plataforma modular. Subsistemas que são adequadamente especificados, permitem que partes de um sistema possam ser usadas, sem ser necessário invocar todos seus módulos.

A figura a seguir mostra a divisão de subsistemas em camadas. As camadas mais baixas são independentes das camadas mais altas. As caixas empilhadas são mutuamente independentes embora não sejam mutuamente exclusivas. Por exemplo, os transportes HTTP, SMTP e JMS são independentes entre si, mas podem ser utilizados juntos.

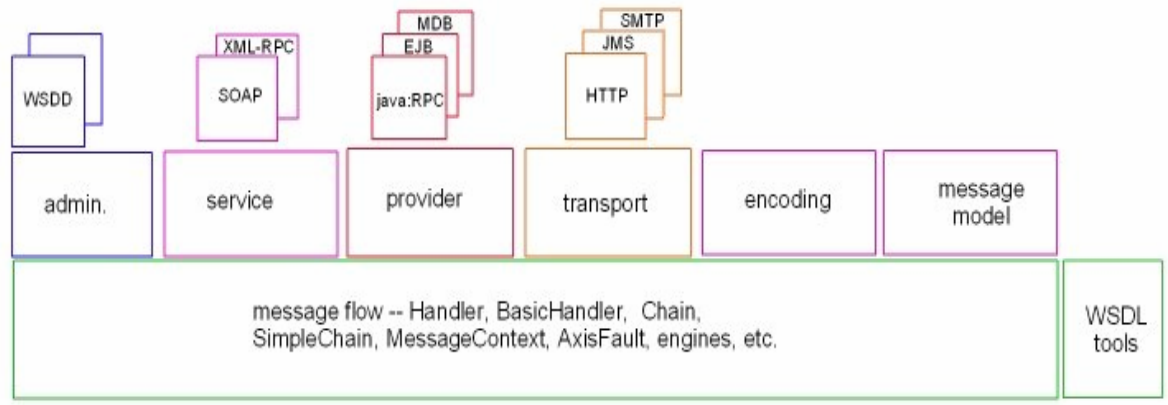


Figura 4 – Subsistemas

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

Na verdade a estrutura interna da plataforma Axis não é dividida em subsistemas como mostra a figura acima. Alguns subsistemas são separados em vários pacotes e estes divididos em subsistemas.

10.3 Subsistema de fluxo de mensagens.

10.3.1 Manipuladores e cadeias.

Manipuladores de mensagens são invocados em seqüência para processar mensagens. Em certo ponto da seqüência um manipulador pode mandar uma requisição (“request”) e receber uma resposta (“response”) ou mesmo processar uma requisição e produzir uma resposta. Uma vez que um manipulador é conhecido ele é considerado o ponto central (“pivot point”) da seqüência. Como descrito anteriormente os manipuladores podem estar na camada de transporte, de serviço ou na camada global. Os manipuladores de cada um dos tipos citados são combinados em cadeias. Assim a seqüência completa de manipuladores engloba três cadeias: transporte, serviço e global. A figura abaixo mostra duas seqüências de manipuladores: o lado cliente, à esquerda e o lado servidor à direita.

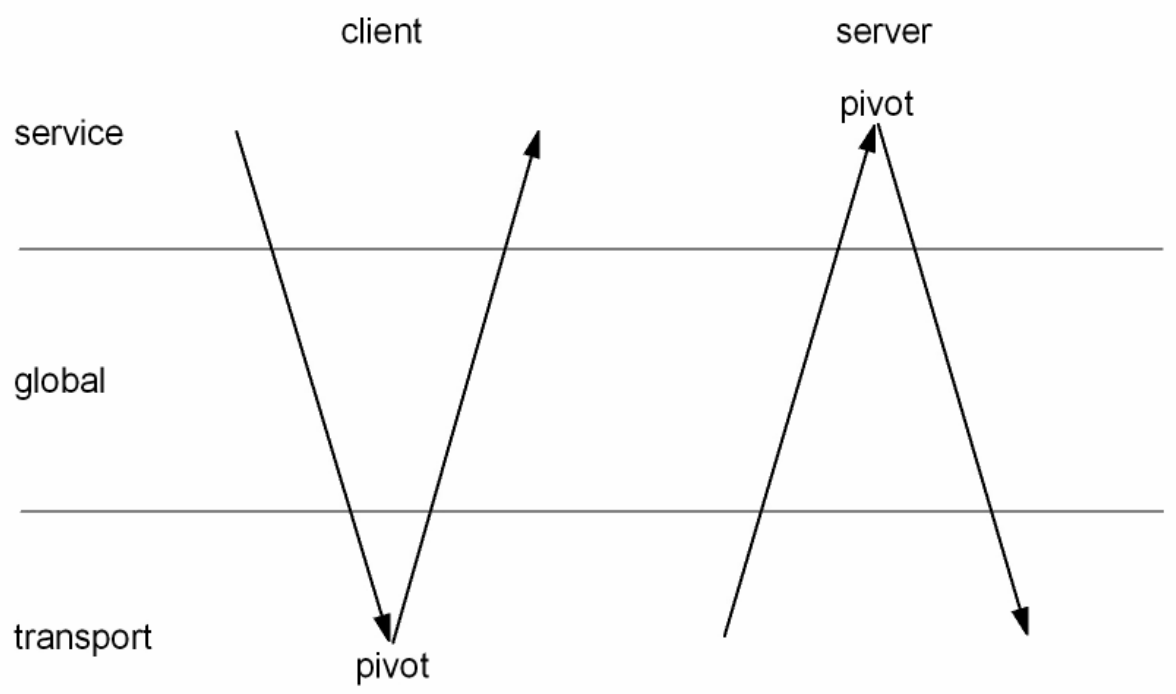


Figura 5 - Fluxo de mensagens.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

Um web service não necessita necessariamente enviar uma mensagem de resposta para cada requisição feita. Entretanto, manipuladores de respostas no caminho das mensagens (lado cliente ou servidor), mesmo que não haja uma mensagem de resposta, por exemplo, para controle de tempo sessão, liberação de recursos, etc.

Uma cadeia é uma conjunto de manipuladores, isto é, agrega uma coleção de manipuladores da mesma maneira que implementa a interface do manipulador, como é mostrado no diagrama UML a seguir:

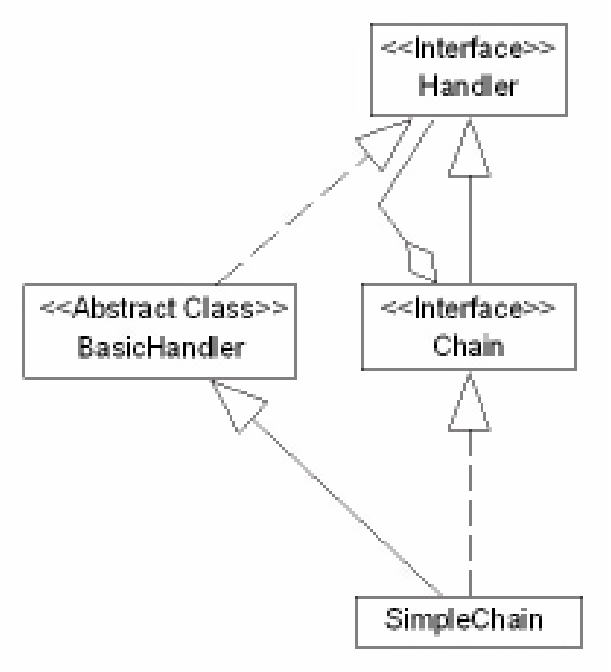


Figura 6 – Conjunto de Manipuladores.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

Uma cadeia também tem semelhança com o “design pattern” cadeia de responsabilidade, no qual uma requisição passa através de uma seqüência de manipuladores até ser processada.

Embora uma cadeia Axis possa processar uma requisição em estágios através de uma sucessão de manipuladores de mensagens, ela possui as mesmas vantagens que a cadeia de responsabilidade: flexibilidade e facilidade para inclusão de novas funções.

Uma mensagem é processada pela passagem através das cadeias apropriadas. Um contexto de mensagem é utilizado para passar a mensagem e ambientes associados, através da seqüência de manipuladores. As cadeias do Axis são projetadas para terem os manipuladores adicionados um de cada vez de acordo

com o fluxo de mensagens. Uma vez adicionados eles passam a manipular as mensagens dependendo do contexto de cada uma.

10.3.2 Cadeias alvo.

Uma cadeia alvo (“Targeted Chain”) é um tipo especial de cadeia que deve possuir um manipulador de requisição, um manipulador “pivot” e um manipulador de resposta. O diagrama de classe a seguir mostra como as cadeias alvo se relacionam com outras cadeias. Uma cadeia alvo é uma agregação de manipuladores que entendem a interface Chain que é um conjunto de manipuladores.

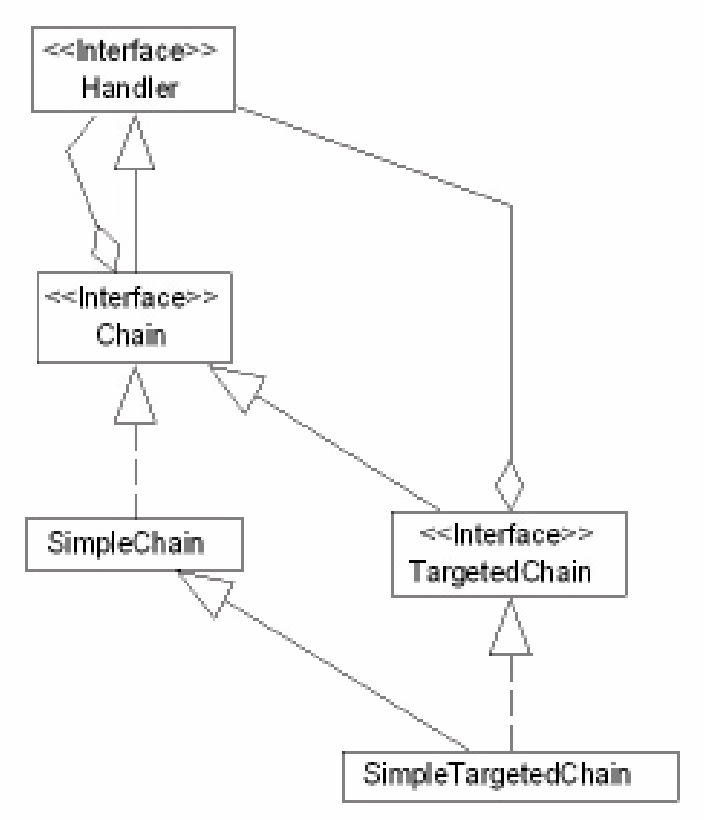


Figura 7 – Targeted Chain

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

10.3.3 Contexto de mensagens.

A estrutura atual de um objeto “MessageContext” é mostrada na figura a seguir. Cada contexto de mensagem deve ser associado com uma requisição de mensagem e/ou uma resposta de mensagem. Cada mensagem possui um SOAPPart e um objeto anexo os quais implementam a interface “Part”.

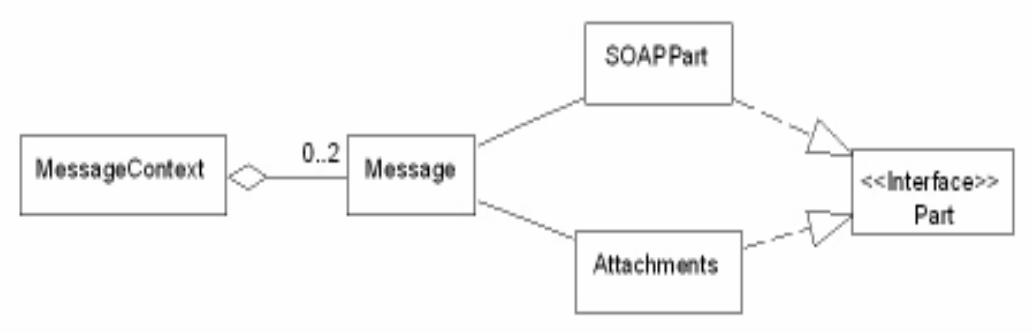


Figura 8 - Contexto de mensagens.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

A classificação do contexto de mensagens deve ser cuidadosamente considerada em relação à arquitetura do Axis. Uma vez que o contexto da mensagem aparece na interface de manipulação, esta não deve ser presa ou induzida a favor do SOAP. A implementação atual é induzida através do SOAP, no qual o método “setServiceHandler” direciona o manipulador especificado para o SOAPService.

10.3.4 Mecanismo.

A arquitetura Axis, possui a classe abstrata AxisEngine com duas subclasses concretas:

- AxisCliente, que trata as cadeias de manipuladores do lado cliente;

- AxisServer, que trata as cadeias de manipuladores do lado servidor.

O relacionamento entre estas classes é muito simples:

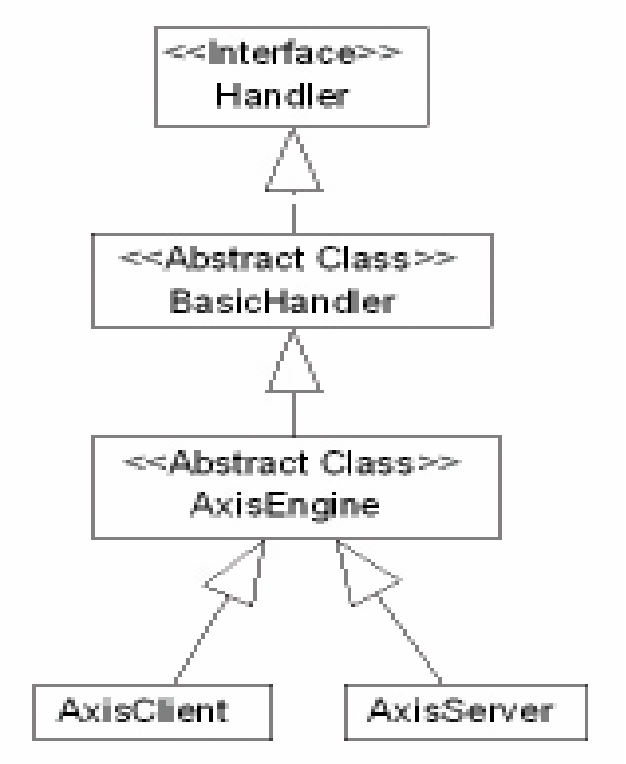


Figura 9 – Relacionamento AxisClient e AxiaServer.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

10.3.5 Configuração do mecanismo.

A interface EngineConfiguration é o meio de configuração dos “factories” de manipulação e de opções globais de uma instância do mecanismo do Axis. Uma instância de uma implementação concreta da classe EngineConfiguration deve ser passada para o mecanismo quando este é criado e o mecanismo deve ser notificado se o conteúdo do objeto “EngineConfiguration” for modificado. O mecanismo do Axis

mantém uma referência para o objeto “EngineConfiguration” e o utiliza para obter os “factories” dos manipuladores de mensagens e as opções globais.

A interface “EngineConfiguration” pertence ao subsistema de fluxo de mensagens o que significa que o subsistema de fluxo de mensagens não depende do subsistema de administração.

10.4 Subsistema de Administração.

O subsistema de administração fornece uma maneira para configurar os mecanismos do Axis. As informações para a configuração de um mecanismo, necessita de uma coleção de “factories” para artefatos em tempos de execução, tais como SOAPServices e cadeias de manipuladores e para o grupo de opções de configurações globais.

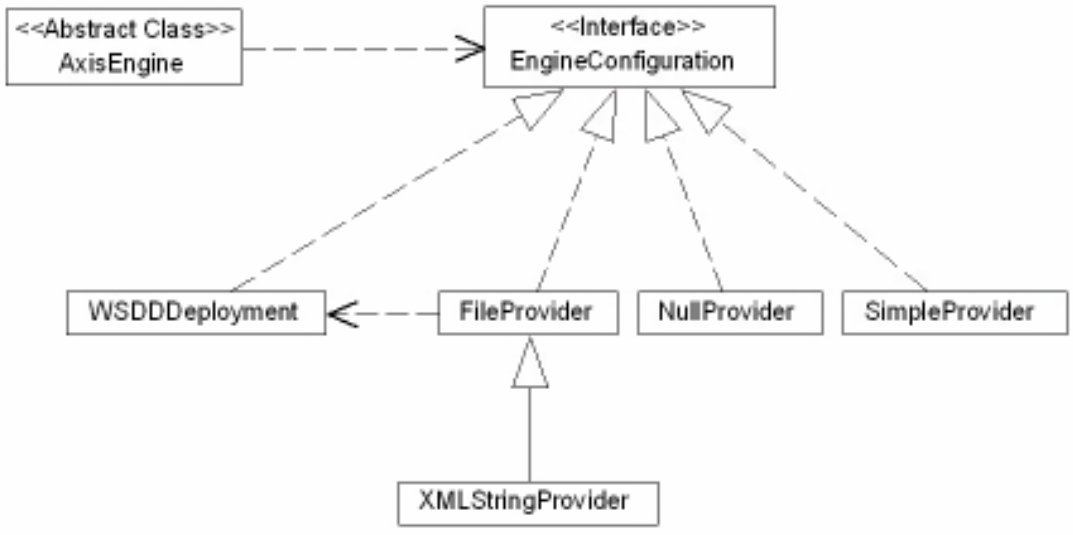


Figura 10 - Subsistema de Administração.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

10.4.1 Administração baseada em WSDD.

WSDD é uma extensão XML para a descrição de desenvolvimento e distribuição que é usada para configurar os mecanismos do Axis de forma estática. A estrutura da gramática WSDD é espelhada por uma hierarquia de classes de "factories" para artefatos criados em tempo de execução. A figura a seguir mostra as classes e os tipos de artefatos criados em tempo de execução.

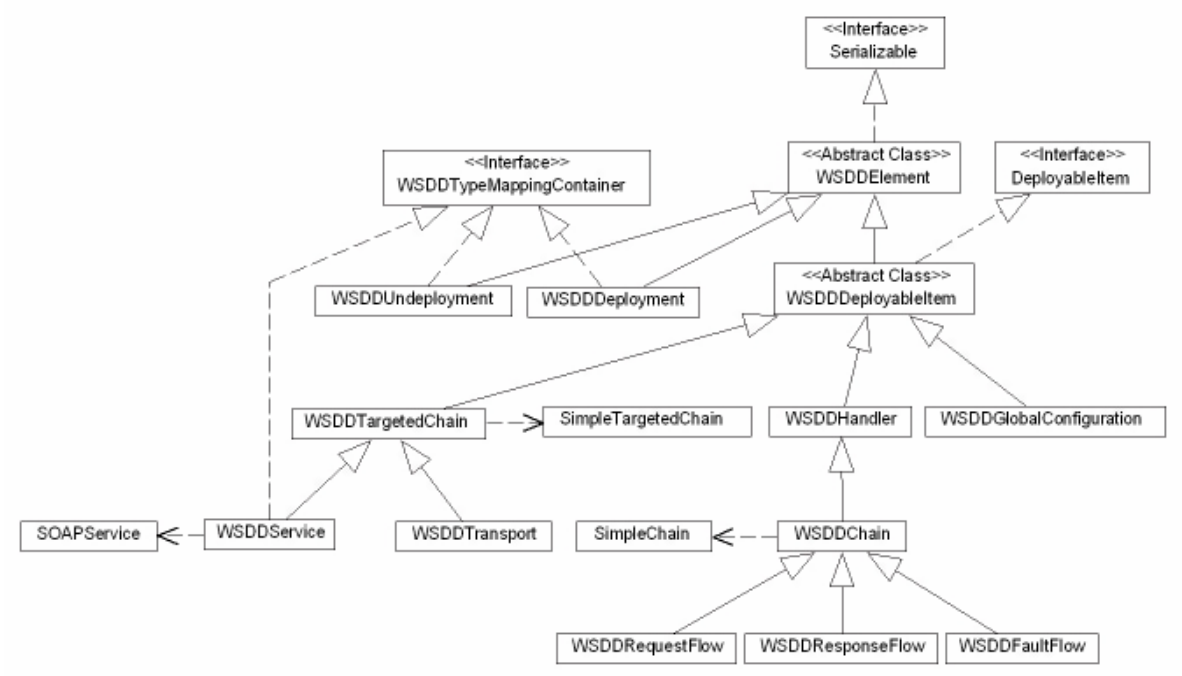


Figura 11 - Administração baseada em WSDD.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

10.5 Subsistema de Modelo de Mensagem.

10.5.1 Modelo de Mensagem SOAP.

A sintaxe XML para uma mensagem SOAP é muito simples. Uma mensagem SOAP consiste em um "envelope" contendo:

- Um cabeçalho opcional contendo nenhum ou vários parâmetros;
- Um corpo contendo nenhum ou vários parâmetros;

- Nenhum ou vários elementos não padronizados.

Modelo da mensagem SOAP:

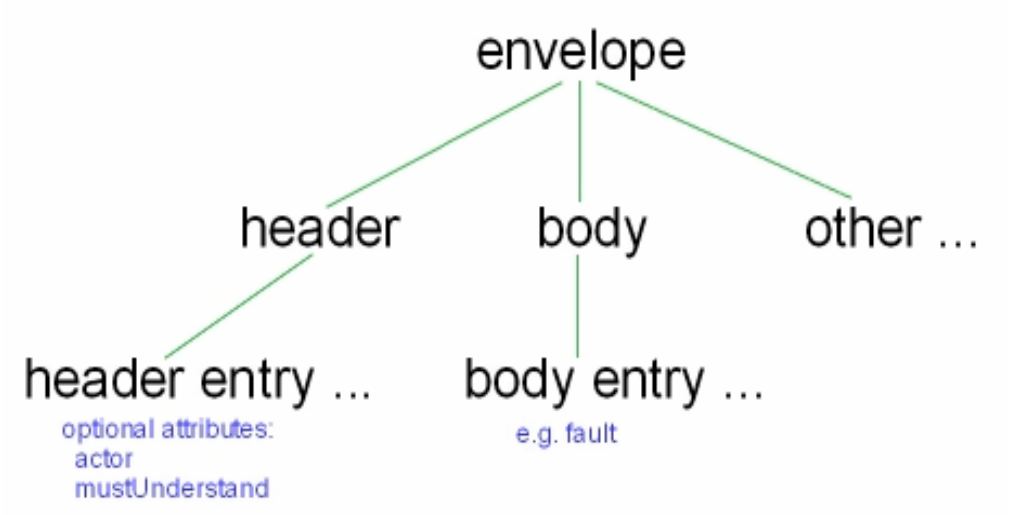


Figura 12 - Mensagem SOAP.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

10.5.2 Elementos da mensagem.

As classes que representam as mensagens SOAP formam uma hierarquia de classes baseadas na classe “MessageElement”. A classe SOAPHeaderElement aparece após o ator e deve compreender seus atributos.

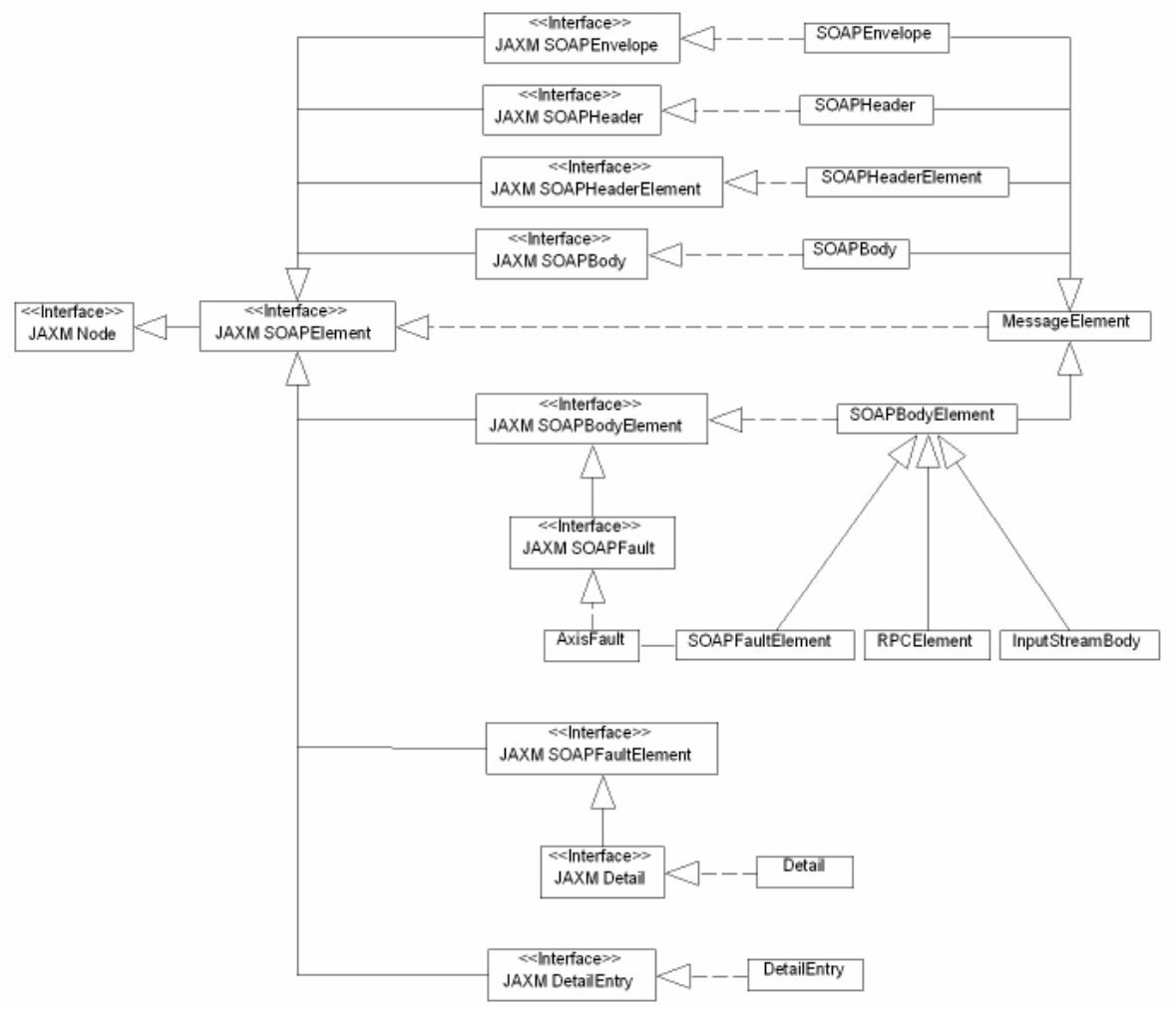


Figura 13 – Elementos das Mensagens.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

10.5.3 Deserialization.

A classe principal responsável pela análise XML é a DeserializationContext (‘DC’). DC gerencia a construção da árvore de análise e mantém uma pilha de manipuladores SAX, uma referência ao “MessageElement” que está sendo analisado.

Os manipuladores SAX formam uma hierarquia de classes:

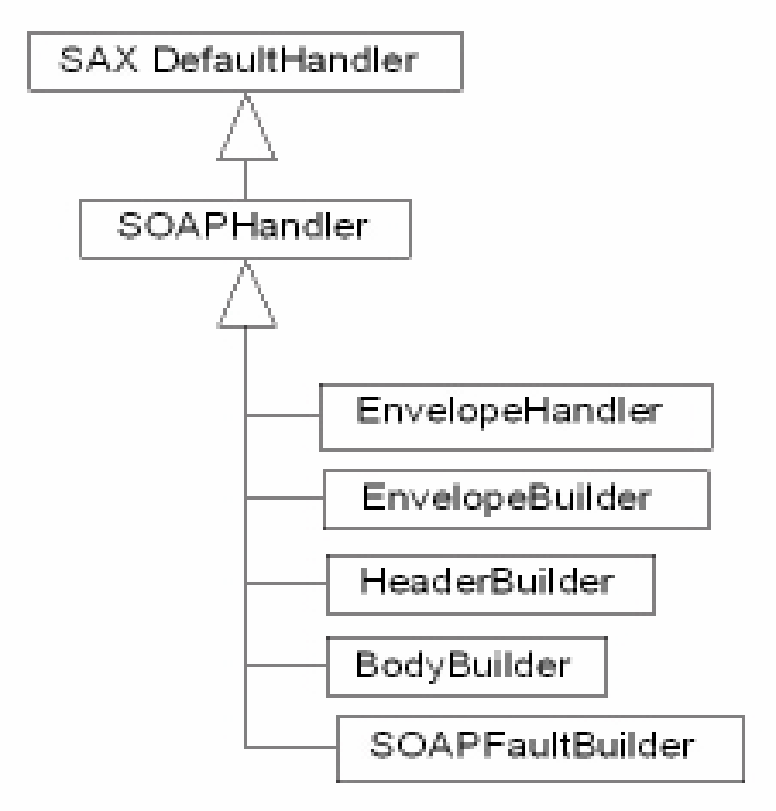


Figura 14 - Manipuladores SAX

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

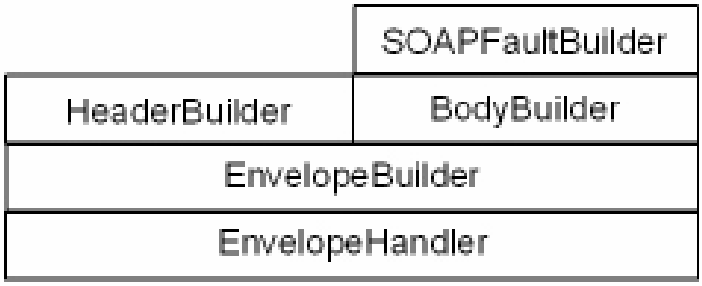


Figura 15 - Empilhamento de instâncias.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

10.6 Subsistema de Codificação.

A codificação é mais facilmente compreendida de baixo para cima. O requisito básico é transformar comandos de linguagem de programação e tipos de dados em representações XML. Em Axis, isto significa codificar objetos Java em XML e decodificar arquivos XML em objetos Java. As classes básicas que implementas estes passos são *serializes* e *deserializers.*

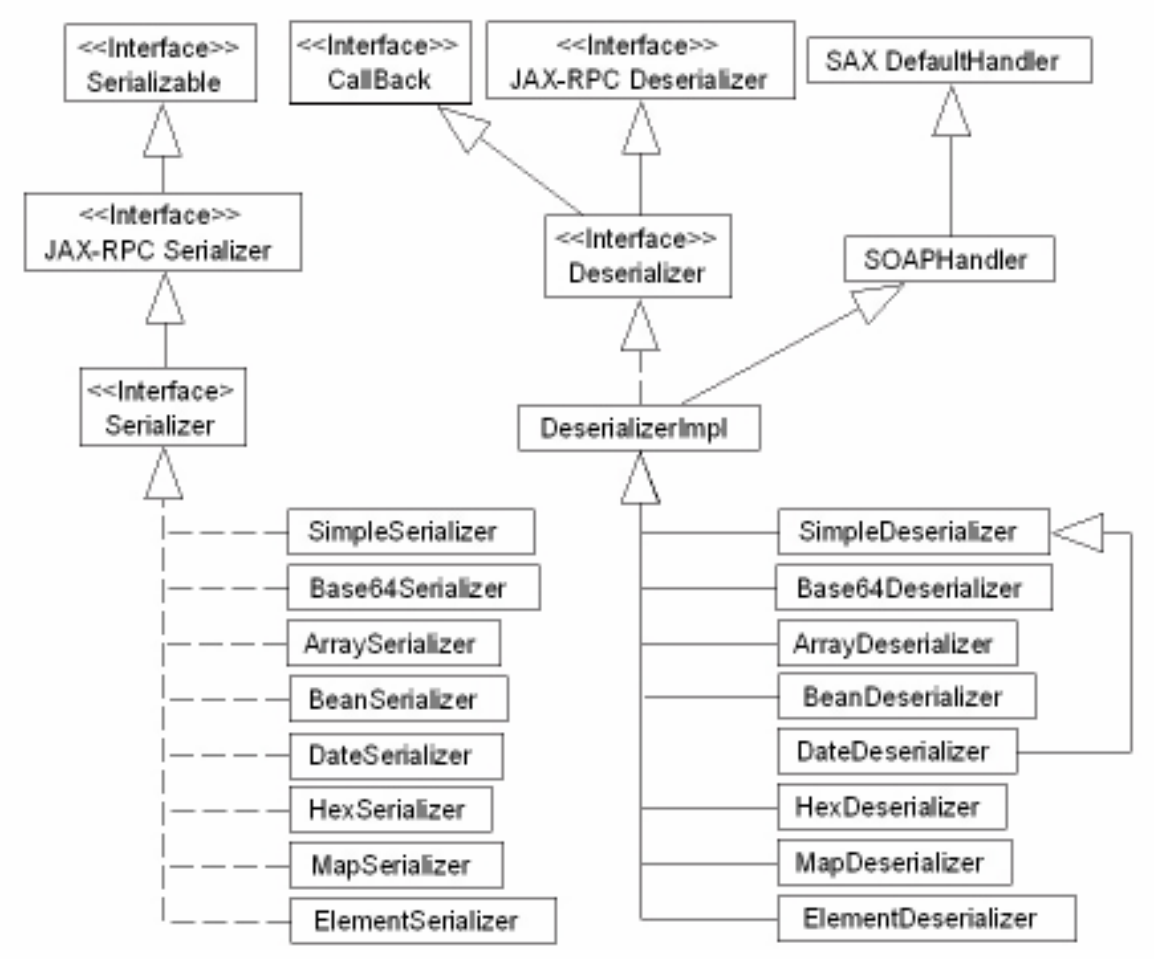


Figura 16 - Classes Serializers e Deserializers.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

Serializers e *Deserializers* específicos são escritos para suportar um mecanismo específico de processamento XML, como por exemplo, DOM ou SAX. Desta maneira "*serializer factories*" e "*deserializers factories*" são introduzidos para construir "*serializers*" e "*deserializers*" por um mecanismo de processamento XML o qual é especificado como um parâmetro.

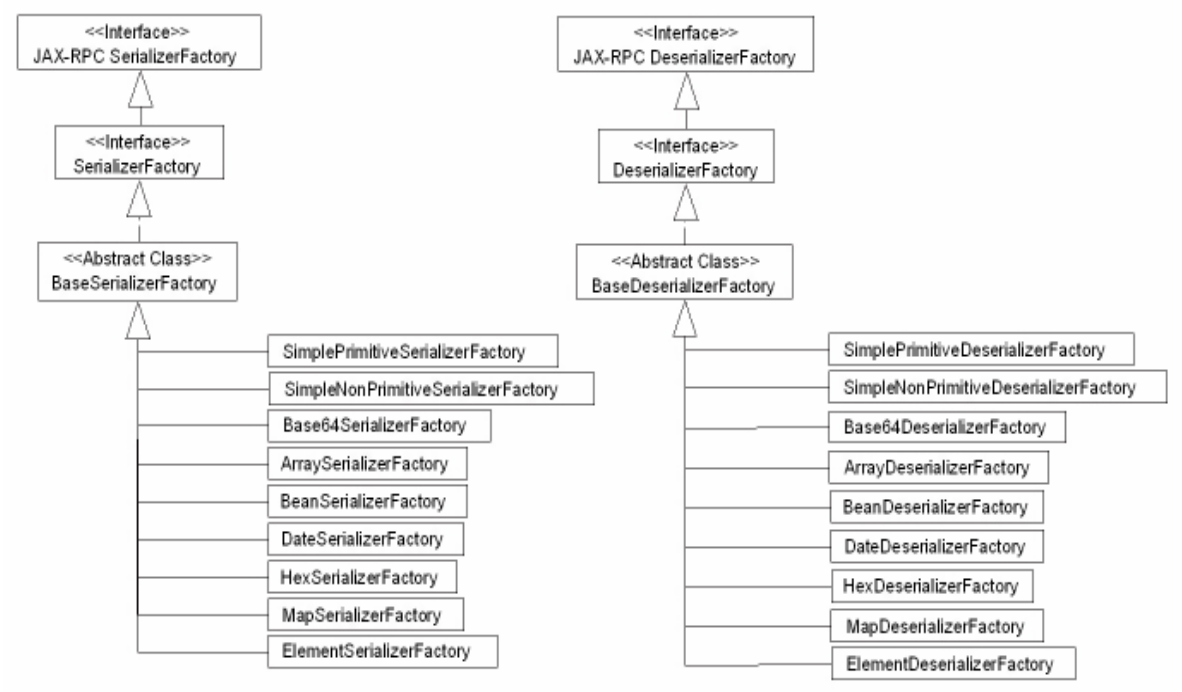


Figura 17 - Mecanismo de Processamento XML.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

Como é mostrado no diagrama acima, cada par de tipo Java e tipo de dado XML que necessitam de codificação e decodificação necessitam de "*serializers*" e "*deserializers*". Assim é necessário o mapeamento de cada par de dados de tipo Java e XML, identificados por um parâmetro "QName". Tal mapeamento é conhecido por "type mapping". A hierarquia da classe "type mapping" é mostrada na figura abaixo.

Figura 18 - Type Mapping.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

Existe um nível final de indireção. Como saber qual “type mapping” usar para uma mensagem em particular? Isto é determinado pela codificação que é especificada na mensagem. Um registro “type mapping” mantém um mapeamento do nome codificado (URI) para o “type mapping”. Vale ressaltar que o tipo de dado XML “QNames” é definido pela codificação.

Figura 19 – Codificação do Type Mapping.

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

Em resumo, para codificar um objeto Java ou valor de dado primitivo para um tipo de dado XML ou para decodificar o tipo de dado XML para Java é necessário conhecer:

- Conhecer to tipo Java que será relacionado;
- O parâmetro QName do tipo de dado XML que será codificado;
- O mecanismo de processamento XML que será utilizado;

10.7 Subsistema de Ferramentas WSDL.

O subsistema de ferramentas WSDL contém WSDL2java e Java2WSDL. O mecanismo de execução do Axis não depende destas ferramentas – elas estão presentes apenas para facilitar o trabalho do usuário.

10.7.1 WSDL2Java e Java2WSDL.

Esta ferramenta faz uma descrição de um web service escrito em WSDL e emite artefatos Java usados para acessar to web service.

Há três camadas na fermenta:

- Framework: SymbolTable, Emitter, WriterFactory;
- WSDL2Java plugin para o framework: WSDL2Java (principal), JavaWriterFactory, e todos os escritores WSDL: JavaPortTypeWriter, JavaBindingWriter, etc;
- Os atuais emissores WSDL2Java, um para cada arquivo gerado: JavaInterfaceWriter, JavaStubWriter, etc.

11 JWSDP

11.1 Introdução.

O pacote de desenvolvimento de Serviços web Java (JWSDP) é uma coleção de ferramentas e bibliotecas projetadas para o desenvolvimento de serviços web em Java de forma simples e eficiente. Apresentado pela primeira vez em janeiro de 2002 como um release “Early Access” (EA), o JWSDP trouxe junto à análise de XML, suporte ao SOAP, um recipiente de serviços (“Service Container”) e ferramentas integradas que podem ser usadas para criar e utilizar uma grande quantidade de serviços distribuídos baseados em protocolo XML (FOSTER, 2002).

Atualmente encontra-se à disposição o release EA2, o qual contém os seguintes componentes:

- Bibliotecas de processamento e de mensagens JAXP XML;
- Bibliotecas para chamada de procedimentos remotos JAX-RPC XML;
- Bibliotecas JAXR para acesso a registros XML;
- Biblioteca Java de sockets seguros (JSSE);
- Biblioteca de tags padrões JSP;
- Ferramenta para construção Apache/Tomcat Ant;
- O container Apache/Jakarta Tomcat Servlet;
- Um registro UDDI simples (WSDP Registry Tool);
- Uma ferramenta de distribuição de aplicações Web.

É de conhecimento geral que partes do JWSDP não são desenvolvidas pela SUN. Na verdade, o JWSDP é, em sua maioria, uma coleção de produtos já existentes; o JWDSP agrupa estas ferramentas de uma forma conveniente para sua instalação. Também, com o JWSDP, existem versões das ferramentas que

funcionam juntas, evitando um grande número de problemas quando tentamos executar algum código.

11.2 História do JWSDP.

Como citado, o JWSDP é uma coleção de ferramentas já existentes e, por certo ponto de vista, uma jogada de marketing da Sun em resposta à plataforma. Net da Microsoft, unindo e colocando em evidência tecnologias já existentes e dando a elas um nome comum.

Na maioria das vezes isto vem em benefício do usuário. Não é mais necessário se preocupar com o que as APIs estão fazendo para sobreviver ao mercado; as APIs existentes no JWSDP estão sob proteção da Sun e possuem boa chance de se tornarem padrões. Também não há necessidade de preocupar-se com as versões e compatibilidade entre diferentes bibliotecas, uma vez que estas vem junto com o JWSDP.

Existem alguns lugares onde a união de bibliotecas resulta em algumas coisas estranhas. Particularmente as bibliotecas JAXM, JAXP e JAX-RPC que são desenvolvidas por iniciativas separadas, pelo projeto Apache XML e pela comunidade Java.

Como são desenvolvidas separadamente, há lugares onde as APIs se sobrepõe. Por exemplo, algumas funcionalidades presentes no JAXM também existem no JAXP.

Outro aspecto que pode ser confuso sobre o JWSDP é que algumas de suas partes estão disponíveis em diferentes formas na Web. Por exemplo, a biblioteca JAXP é distribuída como parte do release 1.4 do JDK. O JWSDP inclui estas

bibliotecas para ser compatível com versões anteriores do JDK, especialmente aquelas que irão demorar algum tempo para migrar paras as versões mais novas.

Será dada na seqüência uma visão geral dos componentes do JWSDP.

11.3 JAXP.

A API Java para processamento XML (“Java Api for XML Processing” – JAXP) fornece interfaces padrões para análise e transformação de documentos XML. JAXP é projetado para preencher um vazio existente nas interfaces padrões do XML; existem interfaces padrões para análise e transformação de documentos XML, mas não existem interfaces para obtenção dos analisadores e transformadores.

Quando o XML começou a ganhar popularidade, não havia padrões para seu processamento. Com o passar do tempo, padrões foram desenvolvidos para o processamento do XML, mas estes padrões eram todos baseados em interfaces e eles não continham nenhuma classe associada. As bibliotecas se enquadravam nos padrões, implementando as interfaces padrões e adicionando a elas seus próprios mecanismos para a criação de instâncias das classes necessárias. Assim, enquanto o processamento de código XML, baseado em interfaces, devia ser portável, os bits de código usados para a criação destes objetos não eram portáveis. Isto significava que as bibliotecas XML não podiam ser compartilhadas sem que pequenas modificações fossem feitas no código da aplicação.

JAXP fornece um conjunto padrão de interfaces e classes que permitem a criação dos documentos XML. JAXP não redefine os padrões adotados pelo XML, ao invés disto dá um impulso para o uso destes padrões (DOM, SAX, XSLT), os quais já são amplamente aceitos. JAXP garante que os códigos utilizando estes

padrões não terão que ser alterados se a biblioteca XML, que suporta a interface padrão, for mudada.

Uma vez que o JAXP não define as interfaces XML por si só, é comum concluir-se que algumas interfaces apresentadas no JAXP não estão nos pacotes Java e Javax. Ao invés disto, a Sun escolheu adotar estes padrões já largamente em uso. Isto significa que a modificação de código para usar o JAXP será limitada àquelas seções de código que criam os analisadores e transformadores XML.

11.4 JAXM.

A API Java para mensagens XML (“Java Api for XML Messaging” – JAXM) possui um problema similar ao anteriormente citado; quando o padrão SOAP (“Simple Object Access Protocol”) foi proposto ele não possuía um conjunto padrão de bibliotecas para Java. Isto não foi uma surpresa uma vez que a Microsoft estava envolvida na criação deste padrão. JAXM aborda este assunto fornecendo uma interface padrão para o SOAP 1.1 e para o SOAP com anexo, de maneira que os programadores Java podem facilmente enviar e receber mensagens SOAP.

JAXM fornece vários recursos além da implementação padrão do SOAP. Isto fornece um mecanismo que suporta a entrega confiável de mensagens para quem usa o protocolo SOAP ou protocolos específicos baseados no SOAP (como por exemplo, ebXML).

Parte do JAXM fornece uma simples implementação para a manipulação de documentos XML. De maneira ideal, JAXM deveria simplesmente alavancar as interfaces fornecidas pelo JAXP, mas isto não acontece.

Isto ocorre porque JAXM e o JAXP foram desenvolvidos em paralelo, por times diferentes da comunidade de desenvolvedores Java. Isto tornou o processo de

integração dos documentos XML do JAXM e do JAXP mais difícil. Se for necessário receber uma mensagem via SOAP e processá-la posteriormente (por exemplo, com transformação XSLT) é necessário transformar o documento de uma representação para outra.

A Sun não fez muita publicidade sobre este assunto. Contudo, espera-se que no futuro estes problemas sejam sanados no JAXM.

11.5 JAX-RPC.

A API Java para a chamada de procedimentos remotos baseados em XML (JAX-RPC) fornece um mecanismo para que o que aparenta ser chamada de objetos através da rede via mensagens baseadas no SOAP. JAX-RPC permite a implementação de Serviços web descritos através de documentos WSDL (“Web Service Definition Language”) que é o padrão aparente para descrição de Serviços web.

Com o JAX-RPC, a implementação do que parece ser um objeto Java pode, na realidade, ser armazenado em uma máquina através da rede (incluindo ai, a internet), em qualquer linguagem que suporte SOAP. Isto cria um poderoso mecanismo para diminuição de problemas em sistemas de negócios. Ao contrário de outros sistemas distribuídos (COM, CORBA) o RPC baseado em XML pode ampliar arquiteturas sem necessidade de grandes investimentos em suporte de software.

JAX-RPC atua de forma similar ao RMI, no qual objetos stubs são criados na invocação de objetos remotos. Conceitualmente os dois sistemas são utilizados de forma idêntica. O que difere JAX-RPC e RMI é o formato dos dados transferidos entre duas máquinas. RMI utiliza um protocolo especifico do Java, em baixo nível (ou CORBA IIOP), enquanto JAX-RPC utiliza XML. Por estes motivos, o RMI deve

ser mais rápido, uma vez que seus protocolos são mais eficientes – mas é importante lembrar que o JAX-RPC não tem seu foco voltado para a performance, mas na interoperabilidade.

JAX-RPC, como JAXM, era um projeto da comunidade de desenvolvimento Java e foram desenvolvidos em paralelo. Entretanto JAX-RPC executa melhor a função de ocultar detalhes de implementação. Ele não necessita expor a estrutura do XML tanto quanto o JAXM.

11.6 JAXR.

Uma vez que os Web Services tenham sido definidos via WSDL, existe a necessidade de um mecanismo para encontrá-los. Documentos WSDL são muitas vezes publicados em mecanismos de registros, os quais fornecem um mecanismo para o armazenamento e recuperação da descrição dos serviços.

Os registros permitem aos usuários procurar por serviços que irão satisfazer suas necessidades e obter as especificações destes serviços.

Atualmente a interface mais comum para o registro de serviços é o UDDI (“Universa Description Discovery and Integration”) e o repositório e registro ebXML (“Electronic Business Using XML”). JAXR, API Java para registros XML, fornece uma interface abstrata para a busca de serviços; JAXR pode ser usado para evitar que o usuário tenha que conhecer as especificações dos registros e repositórios UDDI e ebSML.

Os registros fornecem um rico e complexo mecanismo para a classificação de serviços em categorias. Uma grande parte da API JAXR é direcionada para fornecer uma visão padronizada desta classificação. Esta é a principal razão pela qual a api JAXR deve ser mais significativa para a maioria dos usuários JNDI, porque não faz

mais sentido incluí-las no pacote JNDI. Por outro lado, não é possível usar os registros WSDL sem estas categorias.

JAXR, como os pacotes vistos anteriormente, foi desenvolvido em paralelo. Entretanto o comitê JAXR fez um bom trabalho observando os esforços de outras comunidades de desenvolvimento Java e surgiu com uma especificação que pode ser implementada utilizando JAX-RPC, JAXM, JAXP.

11.7 JSSE.

Pode ficar a impressão de que o JWSDP não é nada além de processamento de documentos XML; o que não é verdade. O JWSDP está fortemente relacionado ao pacote Java XML, o qual é limitado aos pacotes vistos anteriormente. Entretanto, o JWSDP inclui pacotes adicionais que são úteis para a criação e aplicações baseadas na web.

O JSSE ("Java Secure Socket Extension") é um bom exemplo disto. O JSSE fornece um mecanismo para comunicação criptografada através de redes e gerenciamento de chaves envolvidos na criptografia.

O JSSE fornece uma implementação livre do SSL ("Secure Secret Layer") V3 e suporte ao TLS ("Transport Layer Security") 1.0.

Mas o mais importante é que esta implementação facilita dramaticamente a adoção de comunicação segura em organização espalhadas pelo mundo.

Além de fornecer suporte para sockets de baixo nível para criptografia, o JSSE fornece manipuladores adicionais de URLs de modo que a classe "Java.net.URL" consiga compreender e processar URLs HTTP. Isto significa que os serviços web que necessitam de nível de transporte seguro, podem ser facilmente implementados em Java.

11.8 JSTL.

Como os serviços web não se referem apenas a XML, eles não se referem somente ao processamento de negócios. Deve haver um mecanismo que apresente funcionalidades aos usuários finais, tão bem como para máquinas remotas. É comum o uso de JSP ("Java Server Pages") para encapsular a interface com o usuário de uma aplicação; isto fornece uma interface mais simples do que codificar os servlets à mão.

Contudo os JSPs tradicionalmente precisam incluir código Java ou bibliotecas de tags customizadas, para poderem acessar as funcionalidades básicas do negócio. O JSTL ("JSP Standard Tag Library") é projetada para aliviar esta dificuldade. O JSTL fornece um grupo padrão de tags para tarefas comuns do JSP, tais como a interação entre coleções de objetos, consulta à base de dados e internacionalização de textos.

O JSTL também fornece *expressões de linguagem* o que permite o usuário acessar diretamente objetos de negócio com uma simples expressão. Isto permite ao usuário escolher a linguagem que melhor se adapta a suas necessidades, mas mais importante de tudo, protege o usuário de padrões JSP que possam ser desenvolvidos no futuro que possam incluir diretamente *expressões de linguagem.*

O JSTL deriva diretamente de um projeto Apache Jakarta de mesmo nome, o qual posteriormente foi adotado pela comunidade de desenvolvimento Java.

11.9 Ant e Tomcat.

Como o pacote JSTL, Ant vem do projeto Apache Jakarta. O Ant não é uma biblioteca Java – é uma ferramenta. Está incluso no JWSDP pelo simples fato de possibilitar a codificação e construção de aplicações Java de moto mais simples. O

Ant vem com serviços embutidos para a criação de arquivos WAR (“Web Application Archive”), o qual é o pacote default para os Serviços web Java. O Ant é um projeto de código aberto, cujos códigos podem ser encontrados em www.apache.org.

Da mesma maneira, o Tomcat do Apache Jakarta é uma referência para a implementação do JSP e padrões de servlets. Desta forma, ele permite ao desenvolvedor hospedar um web service baseado em servlets. Ao contrário do Ant, o Tomcat não é o mais indicado para a distribuição e instalação de ambientes em produção, por não fornecer um bom controle de redundância, entrega confiável de mensagens, EJBs, ou hospedar outros recursos importantes. Entretanto, ele é grátis, e leve e fácil para instalar; é muito útil para testar aplicações em máquinas de desenvolvimento. Com o Ant, o Tomcat é um produto de código aberto e está disponível em www.jakarta.apache.org.

Junto com o Ant e o Tomcat, o JWSDP inclui um servidor de registro WSDP (“Serviços web Developer Package”). Ele é um simples servidor UDDI, novamente para desenvolvimento e teste. Ao contrário do Ant e Tomcat, WSDP é um produto da Sun e não é código aberto.

11.10 Conclusão.

O JWSDP fornece uma coleção de bibliotecas e ferramentas projetadas para dar ao usuário tudo que ele necessita para iniciar o desenvolvimento e testes de serviços web. Além das bibliotecas de interface padrões , referências às implementações de cada biblioteca são fornecidas. Em alguns casos (JAXP), estas referências estão voltadas à qualidade de desenvolvimento; em outros casos (JAXM), são apenas suficientes ao desenvolvimento sem se preocupar com a qualidade deste. Em todos os casos, as interfaces são projetadas para permitir a mudança de implementações por versões alternativas.

O JWSDP também fornece algumas ferramentas que facilitam o desenvolvimento de serviços web. Elas não substituem servidores de produção (como o WebSphere da IBM ou BEA WebLogic) e outras ferramentas, mas permitem que se inicie no desenvolvimento de serviços web.

Uma vez obtido os arquivos de instalação do JWSDP, o usuário tem tudo que precisa para o desenvolvimento de serviços web.

12 ETTK.

12.1 Introdução.

A plataforma ETTK (IBM Emerging Technologies Toolkit) é um kit para o desenvolvimento de software voltado ao projeto, desenvolvimento e execução de serviços web. A plataforma ETTK fornece um ambiente no qual é possível executar exemplos de tecnologias emergentes que demonstram recentes especificações e protótipos das equipes de desenvolvimento e pesquisa em tecnologia da IBM.

Além disso, a plataforma torna disponível material introdutório que permite aos desenvolvedores se familiarizar com o desenvolvimento de tecnologias autônomas e serviços web.

A plataforma ETTK se originou do pacote da IBM conhecido como WSTK (“Web Services Tookit”). Com a mudança de denominação do pacote WSTK para ETTK, o escopo das tecnologias inclusas no pacote foi ampliado para incluir tecnologias independentes. O novo kit de ferramentas reúne tecnologias relacionadas de vários laboratórios de pesquisa e desenvolvimento da IBM. Os programas de demonstração e funções do ETTK podem ser executados tanto no sistema operacional Linux como no Windows®.

12.2 Funcionamento.

Os componentes de software básico necessários para a criação de serviços web e programas independentes são fornecidos com o ETTK. Junto há uma visão geral da arquitetura de tecnologias independentes, serviços web, programas de exemplo, utilitários e algumas ferramentas que são úteis no desenvolvimento e distribuição de programas independentes e serviços web. O kit de ferramentas também inclui o mecanismo SOAP e um servidor de aplicações.

O ETTK pode ser executado nos sistemas operacionais Windows® 2000, Windows Server 2003, Windows XP, RedHat Enterprise Linux para plataforma Intel 2.1 ou UnitedLinux 1.0.

12.3 Conteúdo do ETTK.

O pacote ETTK contém os seguintes recursos:

- Infra-estrutura para serviços webs e programas independentes, exemplos e documentação para que programadores possam criar seus próprios programas independentes;
- Ambiente pré-instalado (versão do WebSphere Application Server-Express e JDK);
- APIs para o lado cliente para comunicação com documentos WSDL ou um registro UDDI; e APIs para publicação de serviços web;
- Implementação de protótipos de especificações recentemente anunciadas de serviços web;
- Demonstrações que podem ser usadas para o teste do ETTK; Publicação de um serviço e a utilização de um cliente que se comunica com um registro UDDI para encontrar o serviço web e invocá-lo;
- Um conjunto de utilitários para serviços web: Perfil de usuário, medições, contabilidade, contrato e notificação. Junto é fornecido um demonstrativo, que utiliza estes utilitários e faz uso do nível de acordo de serviços web (WSLA);
- Uma visão das funções de segurança da tecnologia, tais como WS-Security, gerenciamento de sistemas e tecnologias SOAP;

- Documentação sobre a arquitetura de serviços web, WS-Inspection e especificações WSDL;
- Um tutorial para o desenvolvimento de serviços web.

12.4 Tecnologias Existentes no ETTK.

O pacote ETTK possui as seguintes tecnologias:

- IBM Cloudscape 10.0: É a plataforma toda desenvolvida em Java que forma o sistema de gerenciamento de base de dados objeto-relacional conhecido como (ORDBMS). Com uma pequena base, porém com grande funcionalidade, o Cloudscape diminui o custo de desenvolvimento, teste e suporte. O Cloudscape permite a criação de versões de aplicações pessoais que podem ser executadas em qualquer máquina virtual Java;
- Active Dependency Integration (ADI): É um modelo de execução e informação para execução de sistemas, serviços e regras de negócios. Suporta a modelagem de tipo de entidades (por exemplo: eventos, aplicações, regras de negócios), a informação associada aos tipo de entidades e a semântica de seus relacionamentos. O modelo de execução ADI monitora e gerencia entidades e seus relacionamentos. Automaticamente atualiza as informações das entidades e reage em resposta a determinadas ocorrências da entidades e violações de integridade.
- Autonomic Manager Toolset (AMTS): O AMTS propõe uma API para a codificação de recursos de serviços web (WS-Resource), sobre o WS-Resource Framework, combinando manipuladores para cada

habilidade ou tipo de porta sob um simples recurso. A infra-estrutura do AMTS propõe uma API para a utilização de entradas do WS-ServiceGroup. O AMTS fornece uma “factory” para a criação de gateways permitindo a objetos Java RMI receberem notificações de serviços web.

- Gerenciamento de Serviços Web Distribuídos (WSDM): O comitê técnico OASIS publicou a versão 0.5 do Administrador Utilizando Serviços Web e Administração de Especificações de Serviços Web. A implementação do ETTK destas especificações permite o gerenciamento de serviços web em execução e o suporte para múltiplos recursos em um único ponto de serviço.
- Policy-Based Pluggable Discovery Framework: O framework para descoberta suporta múltiplas conexões de um usuário e controle de acesso através de múltiplos protocolos de descoberta de serviços através dos padrões do WS-Policy. Programadores e administradores podem especificar políticas para a verificação e autenticação de usuários, através de regras de autenticação e restrições de acesso. Estas políticas são codificadas em XML, de acordo com as especificações do WS-Policy e são automaticamente aplicadas pelo framework de descoberta.
- Objetos de Serviços de Dados (SDO): As APIs do SDO utilizam a implementação do Framework de Modelagem do Eclipse (EMF) 2.0. Os objetos de serialização e desserialização do SDO permitem a integração com o núcleo de execução de serviços web do WebSphere da IBM.

- Common Base Event (CBE): O CBE foi montado sobre a estrutura do EMF. Ele define a estrutura de um evento em um formato consistente. A proposta do CBE é facilitar a comunicação entre diferentes aplicações que suportam logging, administração, gerenciamento de erros, computação independente e funções de e-business. O formato do evento é codificado em um documento XML.
- WS-MetaDataExchange: Define um conjunto de troca de mensagens que permitem o retorno dos meta dados de um serviço. Um conjunto de classes permite o retorno dos meta dados codificados em WSDL.
- Serviço de Objetos de Dados Adaptados Semanticamente (Semantic SDO) é uma extensão da implementação do serviço de objetos de dados existente no ETTK e foi codificado tendo como base o EMF. O “Semantic SDO” adiciona a funcionalidade de uso do RDF e da Ontologia de linguagem web do W3C (OWL), anexando um meta modelo semântico em um data grama do SDO e usando-o para navegar e pesquisar por diferentes data gramas independente de seu modelo estrutural, sem a necessidade de qualquer mudança na API SDO ou da execução de alguma mudança nos dados.

13 GLUE.

13.1 Introdução.

A plataforma webMethods Glue™ é uma plataforma projetada para o desenvolvimento e distribuição de aplicações com serviços web, JSPs e servlets.

A plataforma Glue pode ser utilizada de duas formas: standalone e modo Hosted. No modo standalone, Glue é um servidor de aplicações que, segundo seus desenvolvedores, procura ser mais simples que a utilização de EJB (Entreprise Java Beans). No modo Hosted pode se conectar a outros servidores de aplicação ou servlet e fornecer aos serviços web.

A figura abaixo fornece uma visão geral da arquitetura da plataforma.

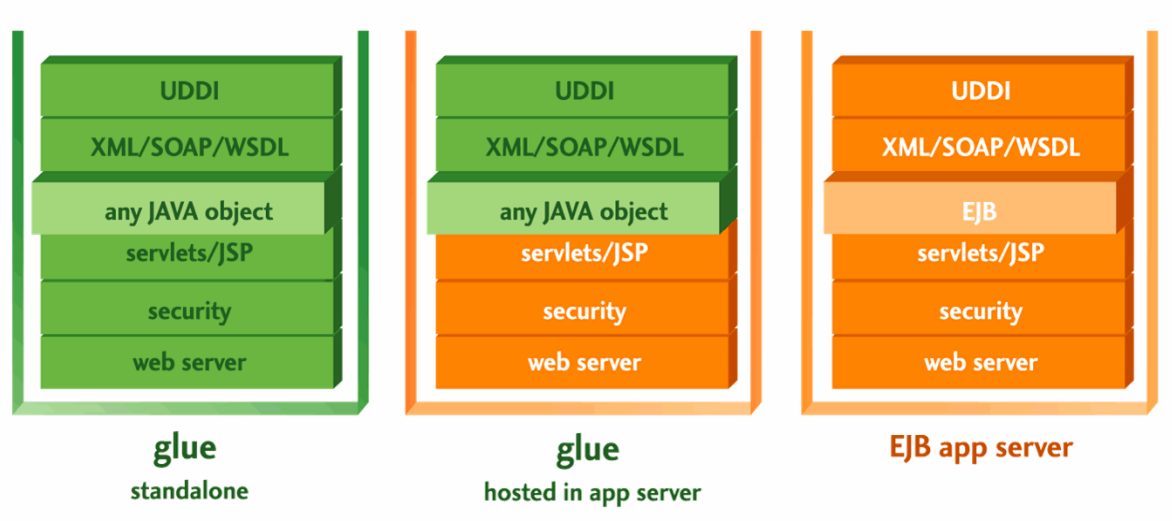


Figura 20 - Arquitetura da Plataforma webMethods Glue

Fonte: WebMethods Glue User Guide http://www1.webmethods.com/docs/glue/guide/index.html

Existem 3 edições da plataforma Glue, e todas, com exceção da edição Standard estão disponíveis para instalação em mainframes.

- Standard : Sem suporte e livre para uso comercial, utilizado para prototipação e projetos de baixo custo

- Professional : Plataforma de classes de negócio, utilizada para projetos robustos e de alta performance de desenvolvimento.
- Enterprise : Possui clusterização de Servlets, JSPs e objetos Java plenos, evitando a complexidade de EJB.

A plataforma webMethods Glue™ também fornece a ferramenta webMethods Fabric™ que permite às organizações migrarem sua aplicações legadas para a arquitetura orientada a serviços da Glue.

13.2 Edições

A seguir são descritos os recursos básicos de cada uma das edições da plataforma Glue.

13.2.1 Standard.

A plataforma Glue Standard pode ser utilizada para a maioria da aplicações comerciais. É uma plataforma que permite a prototipação e desenvolvimento de projetos de baixo custo (WEBMETHODS).

A edição standard possui as seguintes características:

- Publicação de objetos Java com serviços web;
- Invocação de serviços web como objetos Java locais;
- Servidor web, mecanismo de servlets e JSP integrados, permitindo o desenvolvimento local;
- Pode ser hospedado em mecanismos servlets de terceiros;
- Utilitários para conversão wsdl2java e java2wsdl;
- Desenvolvimento de aplicações utilizando J2EE;
- Serviços web seguros usando http e controle de autenticação;

- Configurações armazenadas em arquivos XML;
- Suporte a cabeçalhos SOAP, anexos e estilos de documentos;
- Suporte a JAX-RPC e JAXM;
- Um proxy para suporte para controle de mensagens através de firewalls;

13.2.2 Professional.

A edição Glue Professional é uma plataforma volta ao desenvolvimento de aplicações de grande porte e possui, além dos recursos da edição Standard, as seguintes características:

- Console pare gerenciamento em tempo real que fornece uma apresentação gráfica dos eventos do servidor Glue;
- Distribuição remota; permite a distribuição de serviços web assim que são detectadas alterações no código;
- Integração com EJB;
- Transferência de mensagens sobre JMS;
- Autenticação LDAP via JASS; permite a autenticação dos serviços web utilizando o sistema de autenticação local;
- Assinaturas digitais e criptografia para serviços web;
- Roteamento de serviços web;
- Serviços virtuais; provê recursos para a interceptação e processamento tanto de chamadas Java ou SOAP;

13.2.3 Enterprise.

A edição Glue enterprise é projetada para permitir o agrupamento de nodos Glue unida com a habilidade de agrupar servlets, JSPs e objetos Java. O Recurso

de gerenciamento através de um aplicativo gráfico que permite a visualização de cada um dos nodos do agrupamento.

A figura a seguir mostra a arquitetura da edição Enterprise:

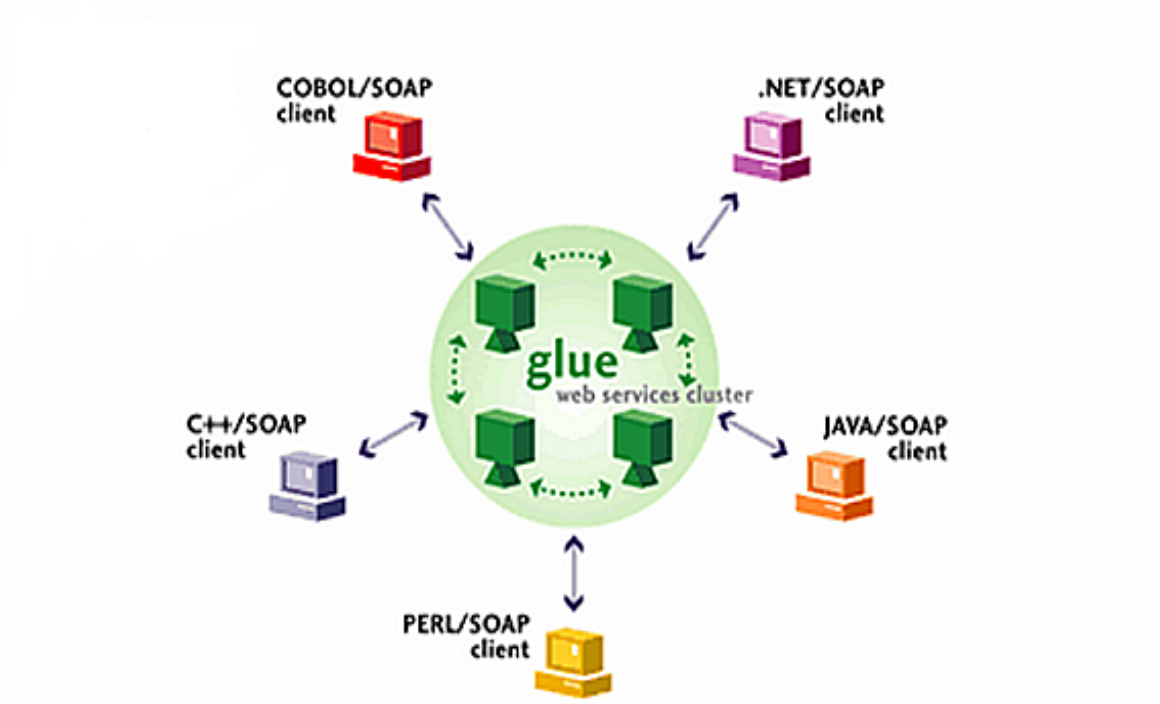


Figura 21 - Estrutura da Edição Enterprise.

Fonte: WebMethods Glue User Guide http://www1.webmethods.com/docs/glue/guide/index.html

13.2.4 webMethod Fabric™

A plataforma webMethods Fabric™ é uma arquitetura orientada a serviço (ESOA), que fornece a infra-estrutura para o desenvolvimento, distribuição e gerenciamento de aplicações baseadas nos padrões dos serviços web. O objetivo da plataforma é fornecer um conjunto de recursos ESOA que atuam entre os produtores e servidores de serviços.

A plataforma também inclui recursos SOA como um servidor UDDI tolerante a falhas, descoberta de serviços dinâmicos e estáticos, recursos de garantia de

qualidade de serviços (QoS). webMethods Fabric™ pode ser executada em modo standalone ou em um servidor J2EE ou .NET.

13.3 Connection Pooling.

13.3.1 Conexões de Entrada (Inbound Connections).

Quando um servidor Glue recebe um pedido de conexão de um cliente remoto, ele concede a conexão, cria a conexão HTTP de entrada e aloca uma "thread" do conjunto de "threads" para atender a conexão. Se não existem "threads" disponíveis, a requisição é colocada em uma fila até que a "thread" se torne disponível. Uma vez que a requisição tenha sido atendida, o servidor HTTP decide entre fechar a conexão ou a manter ativa, assim, outras requisições do mesmo cliente podem ser rapidamente processadas.

13.3.2 Conexões de Saída (Outbound Conections).

Quando um cliente Glue deseja se comunicar com um servidor remoto, ele solicita para um conjunto de conexões de saída por uma conexão HTTP para um servidor. Quando a requisição tiver sido enviada e a resposta tenha sido recebida, a conexão é mantida aberta e devolvida ao conjunto de conexões para ser reutilizada. O conjunto de conexões inicialmente está vazia e cresce de acordo com a demanda de requisições.

13.3.3 Keep Alive.

Por padrão, a plataforma Glue tenta manter as conexões ativas. Contudo esta configuração pode ser alterada dentro da arquitetura da plataforma.

13.3.4 Proxies.

A plataforma Glue gera dinamicamente os proxies, usando a API Standard JDK, o que permite que a criação dos stubs e sua atualização sejam automáticas. A função dos Proxies criados é a de fazer o roteamento das chamadas Java para outro tipo de objeto que implementa a interface “electric.reference.IReference”, que executa a transformação das chamadas Java para o protocolo apropriado.

A plataforma Glue possui seis implementações da interface IReference:

1. JavaToSoap, que converte chamadas Java para o protocolo SOAP;
2. InstrumentingReference, que é um intermediário para concentração de medidas;
3. ObjectService; entrega chamadas Java para um objeto local;
4. VirtualService; entrega chamadas Java para um objeto virtual;
5. StatelessSessionBeanService; entrega chamadas Java para um EJB sem estado;
6. ReferenceChain; que gerencia uma cadeia de IReferences.

13.4 Tipos de Serviços.

Todos os serviços que podem processar mensagens de entrada, devem implementar a interface “electric.service.IService”. Quando um objeto que não implementa esta interface é publicado, a plataforma Glue automaticamente o “encapsula” em uma instância de “electric.service.object.ObjectService”, a qual implementa “IService” e entrega este objeto para um serviço. A alteração de configurações e propriedades, tais como política de ativação de serviços, namespace e ações SOAP, podem ser feitas através de um objeto Context no momento da publicação do serviço.

Quando uma mensagem de entrada é recebida através de uma camada de protocolo, esta camada converte a mensagem no seu equivalente em Java e a entrega ao serviço via interface IService. O código Java resultante do IService é mandado de volta ao cliente no formato apropriado da camada de protocolo.

A integração do Glue com o EJB funciona através da definição de uma implementação do IService chamada "electric.service.ejb.StatelessSessionBeanService" que sabe como entregar mensagens para um bean.

13.5 Camadas de Transporte.

A plataforma Glue pode suportar múltiplas camadas de transporte. A interface IServer define uma abstração que pode aceitar o recebimento de requisições e enviá-las para um diretório em particular ou um registro local. A classe "Servers" fornece métodos estáticos para a manipulação de uma coleção de "IServers".

Existem 2 implementações do IServer:

1. ServletServer, que processa as mensagens que chegam através do protocolo HTTP;
2. JMSContext, que processa mensagens que chegam através do JMS.

A figura abaixo mostra o relacionamento entre um registro local e instâncias do IServer.

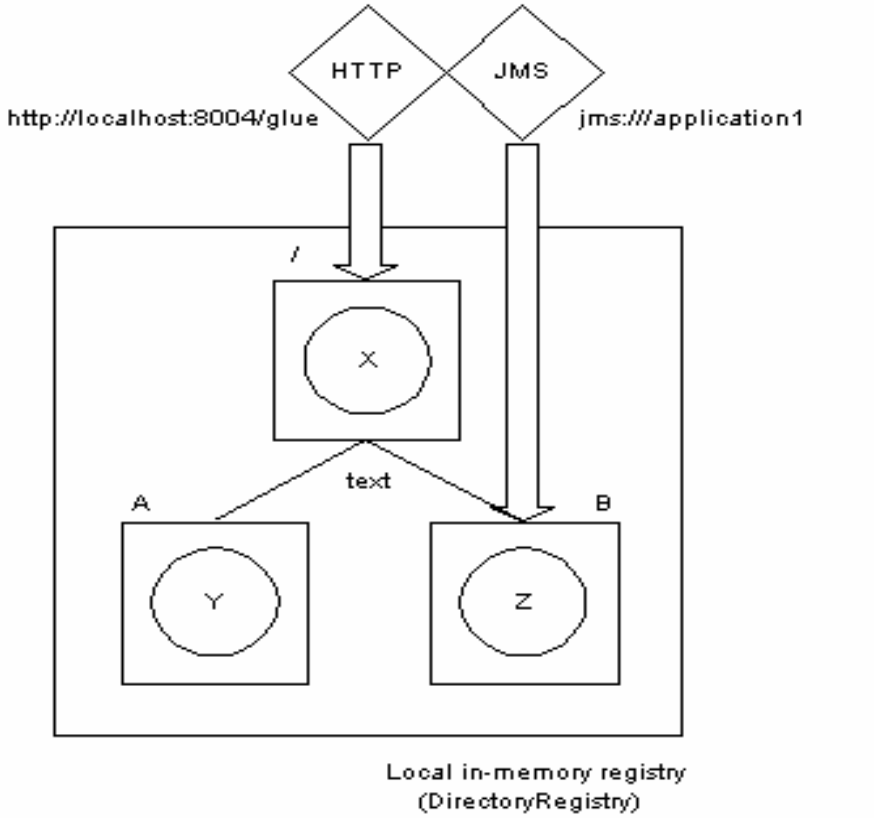


Figura 22 - Relacionamento Entre Registro e IServer.

Fonte: WebMethods Glue User Guide http://www1.webmethods.com/docs/glue/guide/index.html

13.6 Registros.

A API "Registry" fornece os mecanismos para a publicação e ligação de serviços via um ou mais mecanismos de procura de diretórios conhecidos. A interface "IRegistry" define uma abstração de diretório, e a classe "Registry" fornece métodos estáticos para a manipulação de uma coleção de implementações de IRegistry.

A plataforma Glue possui cinco implementações do IRegistry:

1. DirectoryRegistry, que pode publicar um serviço em uma estrutura de diretório em memória;

2. LocalRegistry, que pode ligar diretamente a um serviço local para incrementar a o processamento de mensagens locais;
3. SOAPRegistry, que pode ligar um serviço SOAP descrito através do WSDL e acessado via HTTP, JMS ou outros protocolos de transporte;
4. CrossApplicationRegistry, que pode ligar a serviços locais através dos limites do carregamento das classes.
5. CompoundRegistry, que agrega outros registros.

Registros adicionais podem ser adicionados para suportar serviços de procura (lookup services).

13.7 Conclusão.

Uma vez realizado o estudo teórico das plataformas de desenvolvimento de serviços web, o objetivo para a próxima etapa é a especificação de uma aplicação que será desenvolvida utilizando as plataformas apresentadas.

Serão analisados aspectos tais como:

- Segurança;
- Ambiente de Desenvolvimento;
- Recursos de Software e Hardware necessários;
- Desempenho;
- Integração com outras tecnologias de computação distribuída;
- Limitações;
- Avaliação das plataformas para o desenvolvimento de serviços web em ambientes corporativos;
- Avaliação das plataformas para o desenvolvimento de B2B.

Vale ressaltar que outras medidas de desempenho poderão surgir ao longo do desenvolvimento do protótipo, em virtude de necessidades ainda não detectadas até o momento.

14 PLATAFORMA AXIS.

14.1 Instalação.

A instalação da plataforma Axis, pressupõe a existência de um servidor web; na instalação em questão usaremos o servidor Jakarta Tomcat versão 5.0.28. Estaremos avaliando a versão 1_2rc3 da plataforma Axis.

A plataforma Axis implementa a API JAX-RPC, a qual é uma das maneiras de se programar serviços em JAVA. A especificação da API pode ser encontrada no site da SUN (http://java.sun.com/j2ee/1.4/download.html) .

A plataforma Axis é compilada em um arquivo JAR (Axis.jar) e declarada nas bibliotecas JAXRPC.JAR e SAAJ.JAR. Além destas, são utilizadas várias outras bibliotecas que podem ser agrupadas em um arquivo Axis.war, o qual pode ser publicado em um Servlet Container.

14.2 Configurando do Servidor Web.

Na instalação do servidor web (Tomcat 5.0.28), o diretório weabb/axis da plataforma Axis deve ser copiado para o diretório <Tomcat>/webapps/ ficando a estrutura de diretórios como mostrado na figura abaixo:

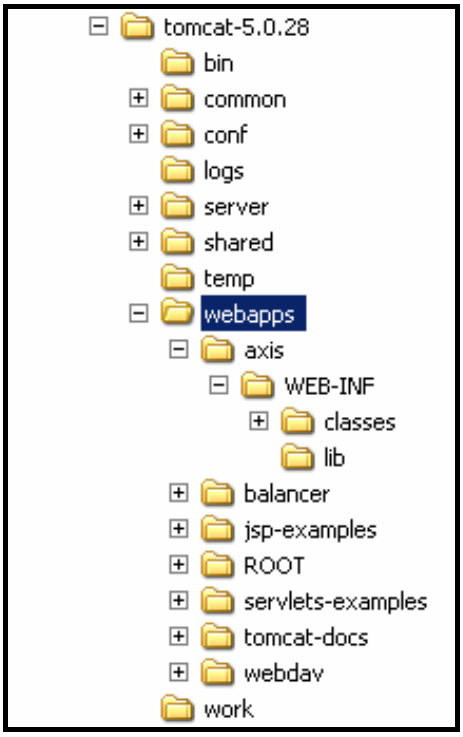


Figura 23 – Estrutura de Diretórios do Servidor Tomcat-Axis.

No diretório webapps do servidor web, encontra-se do diretório WEB-INF e este diretório possui algumas informações de configuração e também é usado para a publicação de web services.

14.3 Bibliotecas.

A plataforma Axis necessita utilizar um analisador XML. O JDK 1.4 da Sun possui o analisador Crimson para análise de código XML. Em nossa avaliação, contudo, usaremos o analisador Xerces XML, embora outros possam ser utilizados, deste que compatíveis com a extensão JAXP 1.1 XML.

As bibliotecas do analisador XML devem ser adicionadas ao diretório <Tomcat>/weapp/WEB-INF/lib.

A figura abaixo mostra a estrutura após a adição das bibliotecas.

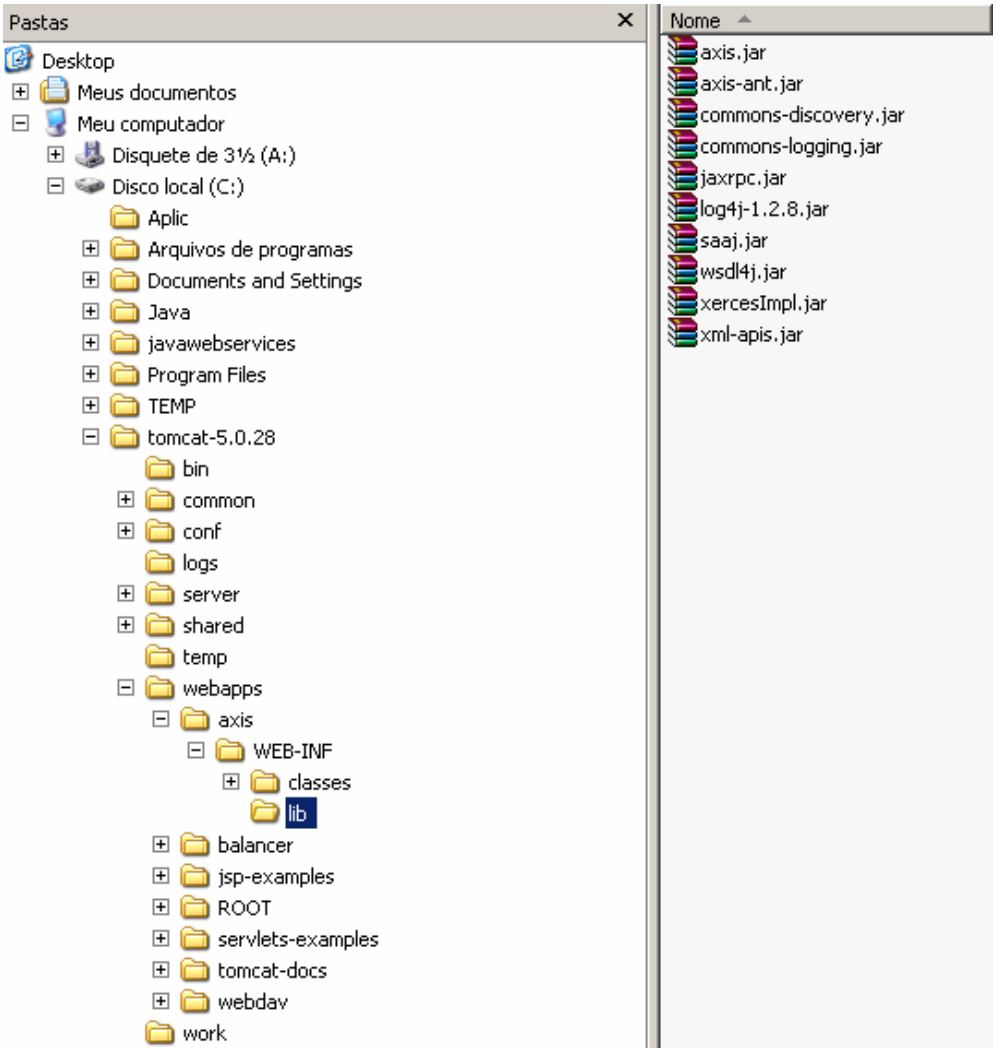


Figura 24 – Diretório de Bibliotecas do Servidor Web.

14.4 Verificação da instalação.

14.4.1 Página Inicial.

Com o servidor web iniciado, a página inicial do Axis pode ser acessada através do seguinte endereço: http://localhost:8080/axis. Esta página mostrará as seguintes opções: Validation, List, Call, Visit, Administer Axis, SOAPMonitor.

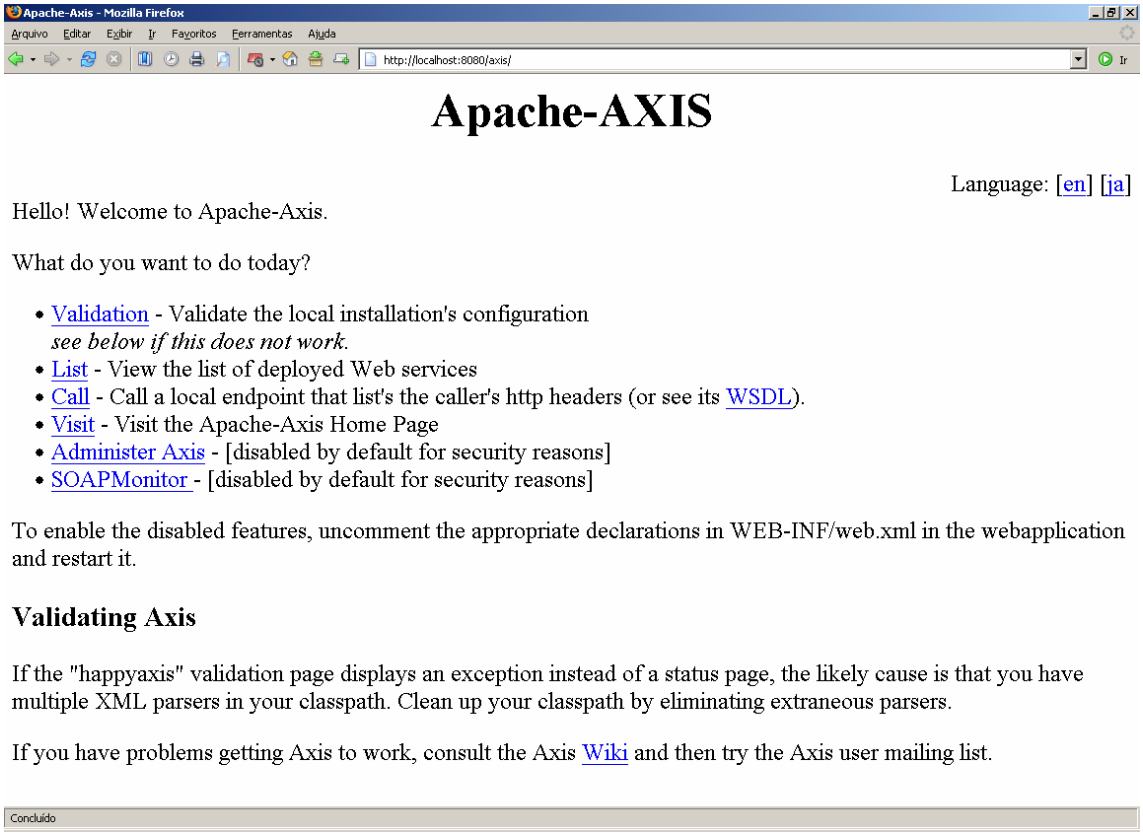


Figura 25 – Página Inicial da Plataforma Axis.

14.4.2 Validação da instalação.

O link Validation acessa a página happyaxis.jsp e fornece um diagnóstico da instalação.

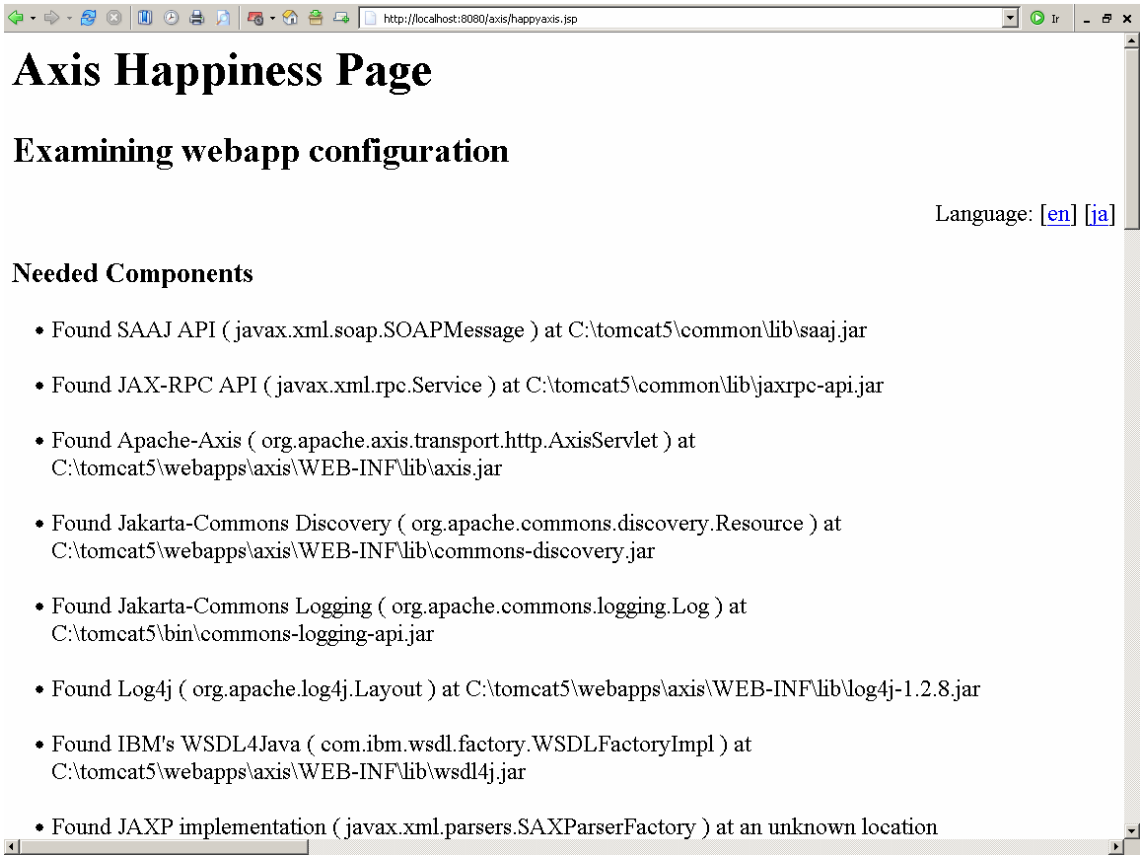


Figura 26 – Axis Happines Page.

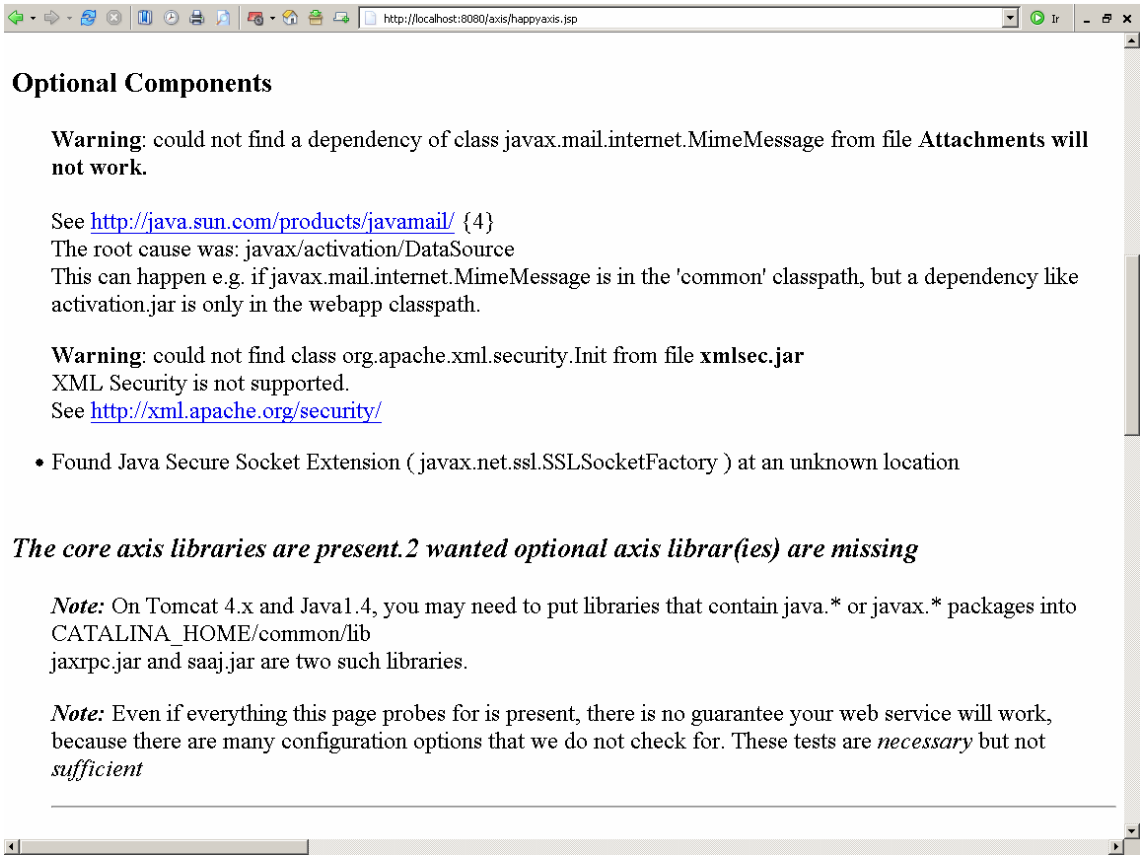


Figura 27 – Optional Components.

O link “List” mostra uma lista dos serviços webs implementados.

Figura 28 – Outros Serviços.

O link “Call” mostra o cabeçalho http do Envelope Soap, ou o arquivo WSDL gerado através dele.

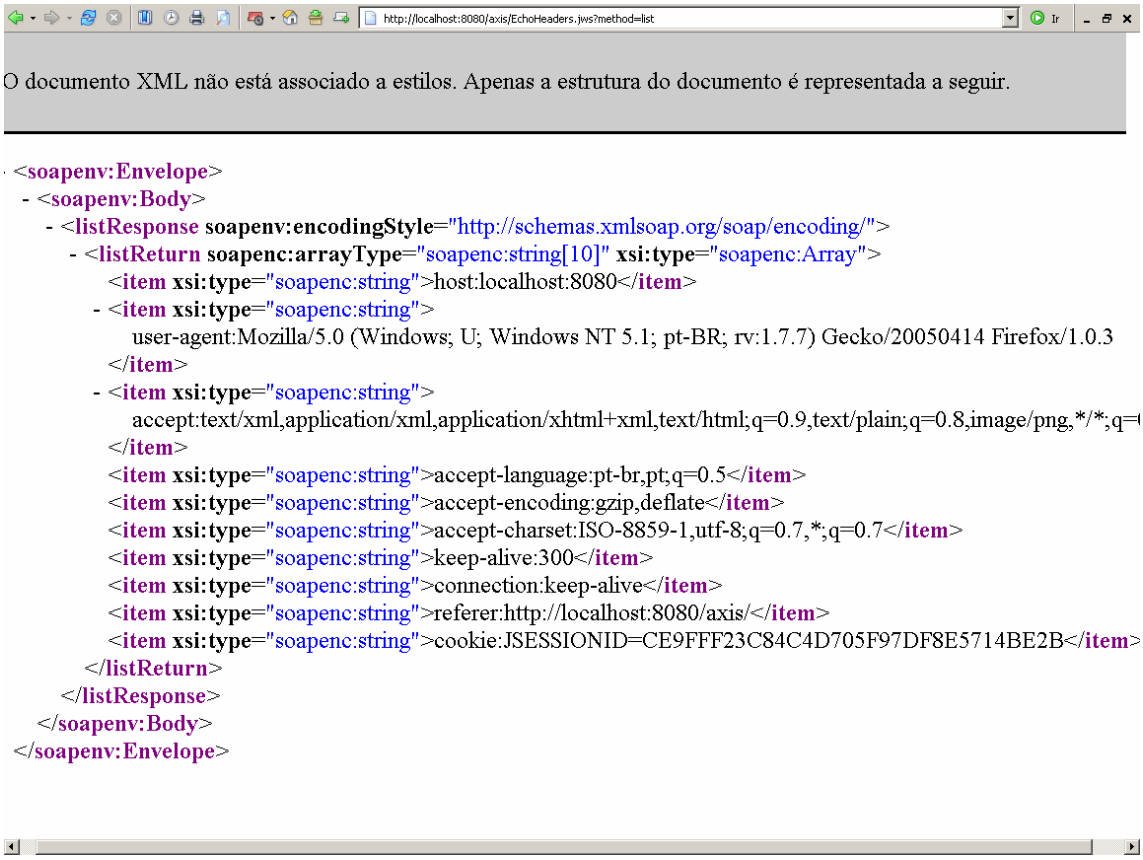


Figura 29 – Especificação do Envelope SOAP.

O link “Visit” leva à página principal do projeto web service apache Axis (http://ws.apache.org/axis/).

O link “Administer Axis” leva à página de administração da plataforma Axis.

O link “SOAPMonitor” leva à página de monitoramento do protocolo SOAP.

14.5 Desenvolvimento de Um Novo Web Service.

Para a instalação de uma nova aplicação, devem ser colocadas no diretório “classes”, sob o diretório WEB-INF do Axis, no servidor Tomcat, as classes compiladas que fazem parte da aplicação.

14.5.1 Variáveis de Ambiente no Cliente.

As classes colocadas no diretório <Tomcat>/axis/WEB-INF/classes, formam o serviço web. É necessário configurar o Axis para que este serviço web seja publicado. A plataforma Axis utiliza um arquivo WSDD (Web Service Deployment Descriptor), que mostra em um arquivo XML as características do serviço web e quais os métodos que devem ser implementados.

Para que o serviço web possa ser publicado as seguintes bibliotecas Java devem ser reconhecidas: axis.jar, commons-discovery.jar, commons-login.jar, jarxrpc.jar, saaj.jar, log4j.jar, xerces.jar.

As bibliotecas descritas devem ficar localizadas no diretório c:\axis\lib, ficando a estrutura de diretórias da seguinte maneira:

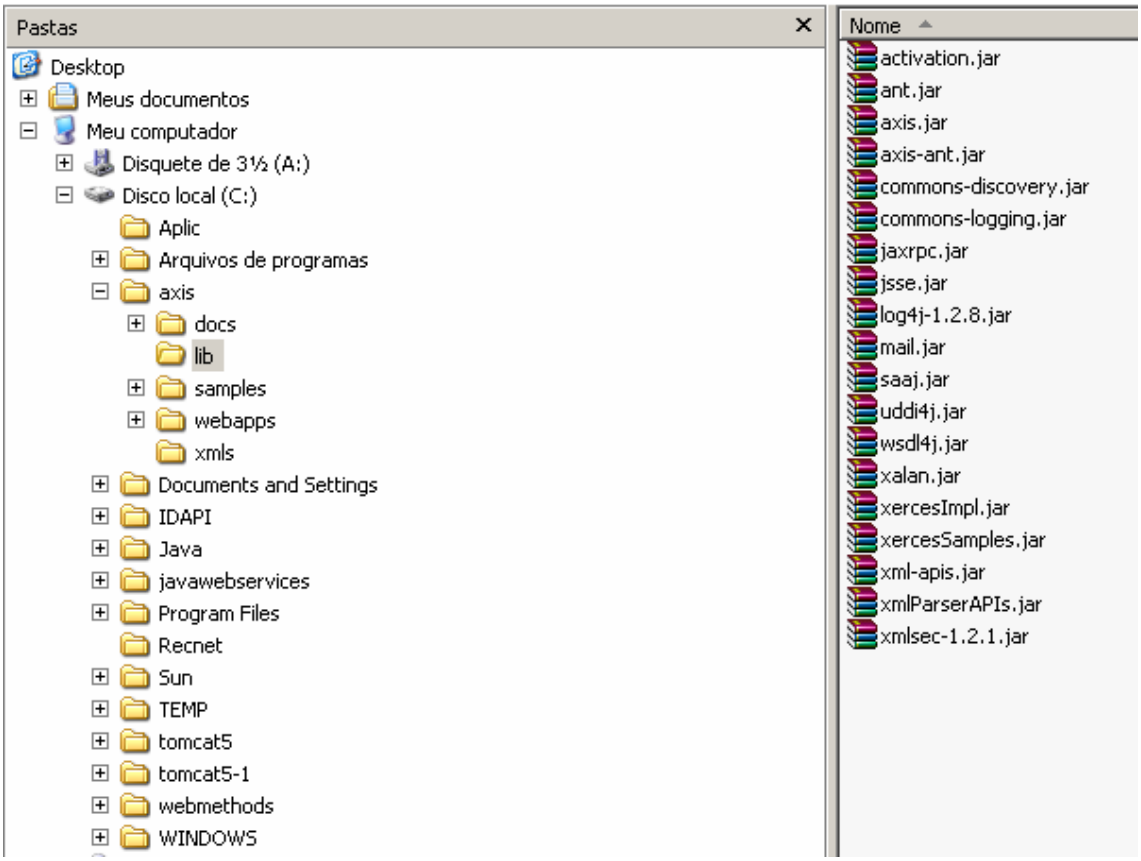


Figura 30 – Estrutura de Diretórios da Plataforma Axis.

É necessária a criação e configuração das seguintes variáveis de ambiente: AXISCLASSPATH, AXIS_LIB, AXIS_HOME. Os passos para a configuração das variáveis de ambiente nos S.Os Windows NT/2000/XP são descritos no Anexo I.

14.6 Estudo de Caso Calculadora.

14.6.1 Criação e Publicação do Serviço Web.

Serão descritos a seguir os passos necessários para a implementação do estudo de caso da calculadora na plataforma Axis.

Inicialmente iremos criar o arquivo “Calculadora.jws” dentro do diretório %CATALINA_HOME%/weapps/axis. É importante chamar a atenção para a extensão do arquivo “jws” (Java Web Service), com o seguinte código:

Quadro 1 - Código da Classe Calculadora.Java – AXIS.

```
package app.ws.calc;
public class Calculadora implements ICalculadora {
    /** Creates a new instance of Calculadora */
  public int getSoma(int i1, int i2) {
    return i1 + i2;
  } //getSoma
  //-----------------------------------------
  public int getSub(int i1, int i2) {
    return i1 - i2;
  } // getSub
  //-----------------------------------------
  public int getDiv(int i1, int i2) {
```

```
    return i1/i2;
  } // getDiv
  //-----------------------------------------
  public int getMult(int i1, int i2) {
   return i1*i2;
  } //getMult
} //Calculadora
```

Pode-se acessar o serviço através do endereço: http://serverhost:8080/axis/Calculadora.jws, onde "serverhost" é o endereço do servidor onde o serviço é disponibilizado.

14.7 Teste da Aplicação.

Através do axis é possível executar uma aplicação usando o protocolo HTTP e o método get. Assim é possível testar uma aplicação através de uma URL. No caso em estudo o endereço para teste será o seguinte: http://localhost:8080/axis/Calculadora.jws?method=getSoma&i1=2&i2=4.

O endereço é formado pela união do endereço do serviço web (http://localhost:8080/axis/Calculadora.jws"), a variável "method" que indica o método a ser chamado e os parâmetros necessários a execução deste método (i1 e i2).

O resultado é mostrado através de um documento XML com o valor 6.0. É mostrado a seguir o documento XML resultante:

Quadro 2 - Resultado da Execução da Classe Calculadora – AXIS.

```
1 <soapenv:Envelope>
2 <soapenv:Body>
3<getSomaResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
4 <getSomaReturn href="#id0"/>
5 </getSomaResponse>
6<multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="xsd:float">6.0</multiRef>
7 </soapenv:Body>
8 </soapenv:Envelope>
```

14.8 Criação da Aplicação Cliente.

Com o serviço publicado através de um arquivo WSDL no servidor, é necessário criar uma aplicação do lado cliente que implemente a interface fornecida.

Para tanto criaremos o servidor Tomcat5-1, que será o servidor web do cliente . O arquivo server.xml será alterado para que as requisições HTTP sejam atendias pela porta 7080. Esta instalação do tomcat será idêntica ao servidor da calculadora, mas possuirá a aplicação específica do cliente.

Teremos então a seguinte estrutura de diretórios.

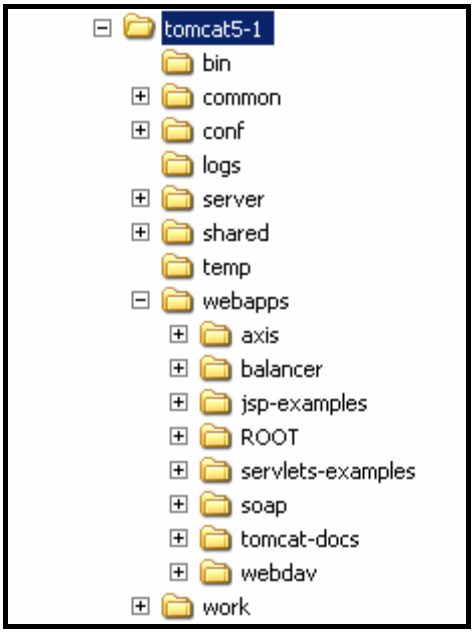


Figura 31 – Estrutura de Diretórios do Servidor Tomcat do Lado Cliente.

No diretório weapps/axis do servidor tomcat do cliente, será criada a classe Cliente.Java com o seguinte código:

Quadro 3 - Código da Classe Cliente.Java – AXIS.

```
import org.apache.axis.client.Service;
 import org.apache.axis.client.Call;


 public class Cliente {
    public static void main(String[] args) throws Exception {
        // Endereço, local onde se encontra o Web Service
        String local = "http://localhost:8080/axis/Servico.jws";


        // Criando e configurando o serviço
        Call call = (Call) new Service().createCall();
        // Configurando o endereço.
        call.setTargetEndpointAddress(local);
        // Marcando o método a ser chamado.
        call.setOperationName("soma");


        // Parâmetros da função soma.
        Object[] param = new Object[]{new Integer(2),new Integer(4)};
        // Retorno da Função
        Integer ret = (Integer)call.invoke(param);
           // Imprime o resultado: ret = 2 + 4.
        System.out.println("Resultado da soma : " + ret);
    }
 }
```

O Código acima será armazenado no arquivo Cliente.Java que quando executado apresentará “Resultado da soma : 6”

A plataforma Axis suporta WSDL de 2 maneiras:

1. “Quando se instala um serviço em AXIS os clientes podem acessar a URL desse serviço (com um navegador padrão) com “?WSDL" no fim para obter um documento WSDL, gerado automaticamente e que descreve o serviço
2. É fornecida a ferramenta "WSDL2Java" que constrói *proxies* e *skeletons* para serviços com descrições WSDL.

Quadro 4 - Comando Wsdl2Java.

```
$ Java org.apache.axis.wsdl.WSDL2Java <WSDL-file-URL>
```

O acesso o arquivo WSDL do serviço é feito através do seguinte endereço: http://serverhost:8080/axis/Calculadora.jws?wsdl. No caso em estudo e de acordo com as configurações feitas, a descrição em XML da interface a ser implementada será mostrada da seguinte forma:

Quadro 5 - Especificação XML da Implementação da Interface-AXIS.

```
<wsdl:definitions
targetNamespace="http://localhost:8080/axis/Calculadora.jws">
<!--
WSDL created by Apache Axis version: 1.2RC3
Built on Feb 28, 2005 (10:15:14 EST)
-->
<wsdl:message name="getSubRequest">
```

```
<wsdl:part name="i1" type="xsd:float"/>
<wsdl:part name="i2" type="xsd:float"/>
</wsdl:message>
        <wsdl:message name="getSomaResponse">
<wsdl:part name="getSomaReturn" type="xsd:float"/>
</wsdl:message>
        <wsdl:message name="getSomaRequest">
<wsdl:part name="i1" type="xsd:float"/>
<wsdl:part name="i2" type="xsd:float"/>
</wsdl:message>
        <wsdl:message name="getMultResponse">
<wsdl:part name="getMultReturn" type="xsd:float"/>
</wsdl:message>
        <wsdl:message name="getDivRequest">
<wsdl:part name="i1" type="xsd:float"/>
<wsdl:part name="i2" type="xsd:float"/>
</wsdl:message>
        <wsdl:message name="getDivResponse">
<wsdl:part name="getDivReturn" type="xsd:float"/>
</wsdl:message>
        <wsdl:message name="getSubResponse">
<wsdl:part name="getSubReturn" type="xsd:float"/>
</wsdl:message>
        <wsdl:message name="getMultRequest">
<wsdl:part name="i1" type="xsd:float"/>
```

```
<wsdl:part name="i2" type="xsd:float"/>
</wsdl:message>
        <wsdl:portType name="Calculadora">
        <wsdl:operation name="getDiv" parameterOrder="i1 i2">
<wsdl:input message="intf:getDivRequest" name="getDivRequest"/>
<wsdl:output message="intf:getDivResponse" name="getDivResponse"/>
</wsdl:operation>
        <wsdl:operation name="getMult" parameterOrder="i1 i2">
<wsdl:input message="intf:getMultRequest" name="getMultRequest"/>
<wsdl:output message="intf:getMultResponse" name="getMultResponse"/>
</wsdl:operation>
        <wsdl:operation name="getSoma" parameterOrder="i1 i2">
<wsdl:input message="intf:getSomaRequest" name="getSomaRequest"/>
<wsdl:output message="intf:getSomaResponse" name="getSomaResponse"/>
</wsdl:operation>
        <wsdl:operation name="getSub" parameterOrder="i1 i2">
<wsdl:input message="intf:getSubRequest" name="getSubRequest"/>
<wsdl:output message="intf:getSubResponse" name="getSubResponse"/>
</wsdl:operation>
</wsdl:portType>
        <wsdl:binding name="CalculadoraSoapBinding" type="intf:Calculadora">
<wsdlsoap:binding                                                style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="getDiv">
<wsdlsoap:operation soapAction=""/>
```

```
        <wsdl:input name="getDivRequest">
<wsdlsoap:body    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>
</wsdl:input>
        <wsdl:output name="getDivResponse">
<wsdlsoap:body    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/axis/Calculadora.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
        <wsdl:operation name="getMult">
<wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getMultRequest">
<wsdlsoap:body    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>
</wsdl:input>
        <wsdl:output name="getMultResponse">
<wsdlsoap:body    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/axis/Calculadora.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
        <wsdl:operation name="getSoma">
<wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getSomaRequest">
<wsdlsoap:body    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>
```

```
</wsdl:input>
        <wsdl:output name="getSomaResponse">
<wsdlsoap:body    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/axis/Calculadora.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
        <wsdl:operation name="getSub">
<wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getSubRequest">
<wsdlsoap:body    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>
</wsdl:input>
        <wsdl:output name="getSubResponse">
<wsdlsoap:body    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/axis/Calculadora.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
        <wsdl:service name="CalculadoraService">
        <wsdl:port binding="intf:CalculadoraSoapBinding" name="Calculadora">
<wsdlsoap:address location="http://localhost:8080/axis/Calculadora.jws"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

15 PLATAFORMA DE DESENVOLVIMENTO DE WEB SERVICE – GLUE.

15.1 Instalação.

Passos para Instalação e configuração do ambiente de desenvolvimento de Web Service GLUE:

1. Para a instalação acontecer com êxito, a máquina virtual JAVA (JVM) deve estar instalado no equipamento;
2. Através de um arquivo de instalação o usuário inicia facilmente os procedimentos de instalação, onde uma tela com as informações institucionais abrirá, sendo que o usuário deverá clicar em NEXT para continuar;
3. O usuário deverá aceitar os termos tradicionais de licença de uso de um Software na Versão DEMO standard;
4. O usuário deverá escolher o diretório de instalação;
5. Abrirá uma tela para escolher onde ficarão os ícones;
6. Será dada a opção para escolher entre a opção de instalação típica ou customizada, onde escolhemos a primeira;
7. É apresentado um sumário descritivo, com as opções antes selecionadas, dando a oportunidade de o usuário retornar e modificar alguma que não foi de seu agrado:
8. Inicia o processo de instalação propriamente dito:
9. É apresentada uma tela final com a finalização da instalação, solicitando que o computador seja iniciado novamente para que as configurações realizadas surtam efeito:

Para o funcionamento da plataforma GLUE, utilizaremos o servidor jakarta tomcat versão 5.0.28, sendo que a versão utilizada do Glue é a 5.0.2.

15.2 Bibliotecas.

Após a instalação da plataforma GLUE, no diretório /lib várias bibliotecas são instaladas. Estas bibliotecas serão utilizadas para a confecção de um novo web service através desta plataforma. Entre estas, temos a biblioteca JAXRPC-API.JAR.

A figura abaixo mostra a estrutura do diretório que contém as bibliotecas, após a instalação do GLUE.

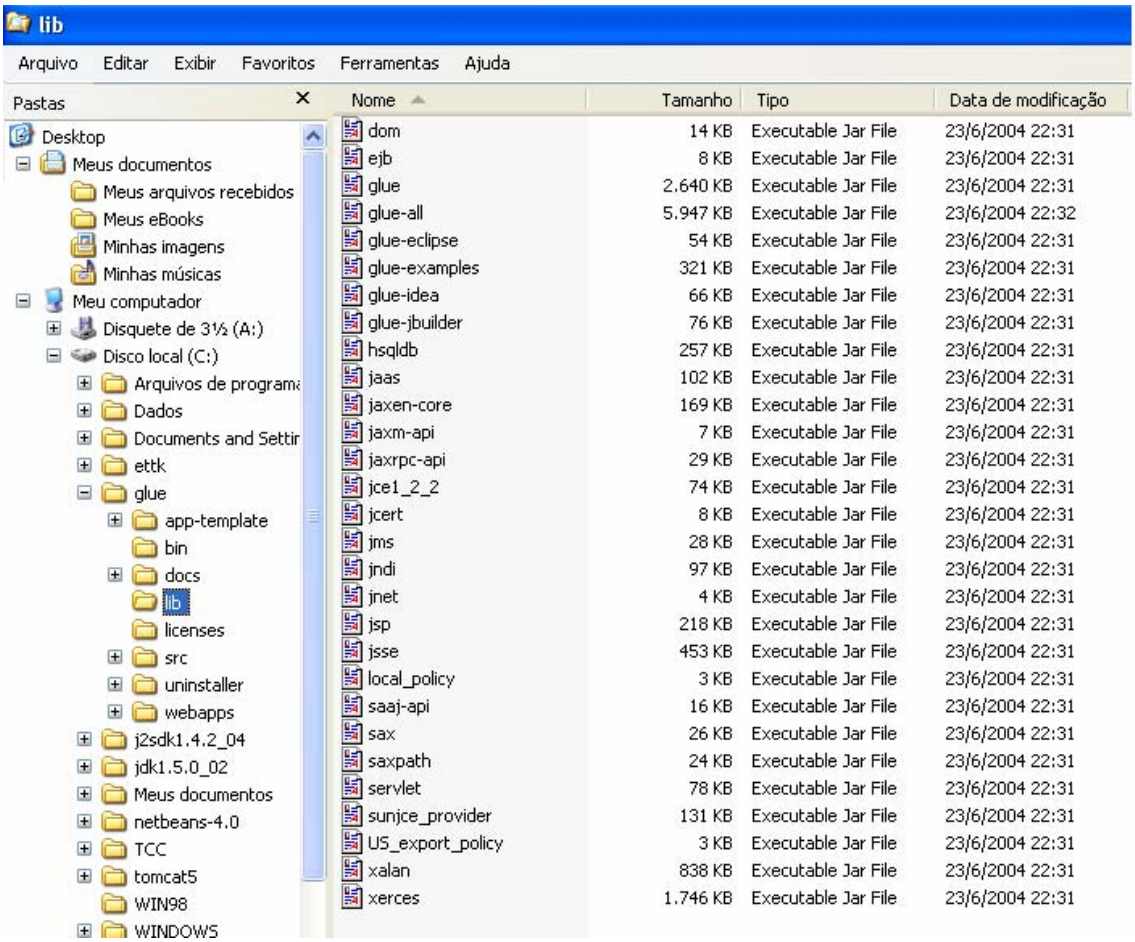


Figura 32 – Diretório de Bibliotecas do GLUE.

Ao utilizarmos a plataforma GLUE, estaremos utilizando a API JAX-RPC produzida pela Java Community Process, que executa chamadas RPC SOAP.

Mesmo que a API padrão GLUE forneça um conjunto de funcionalidades de fácil utilização compatível com JAX-RPC, é também fornecida a implementação da API JAX-RPC para que os desenvolvedores GLUE tenham liberdade de escolha. Para acessar um tutorial da API JAX-RPC, este pode ser encontrado no endereço http://java.sun.com/webservices/docs/1.0/tutorial/index.html.

Cabe lembrar que a API GLUE JAX-RPC não está completa ainda, a WebMethods (empresa desenvolvedora do GLUE) está em processo de licenciamento da JAX-RPC TCK da SUN, a fim de passar nos testes de compatibilidade.

15.3 Configurando do Servidor Web.

No decorrer da explanação a respeito da plataforma de desenvolvimento GLUE, iremos nos apoiar no exemplo de uma aplicação, que será a de uma calculadora. Na instalação do servidor web (Tomcat 5.0.28), no diretório webapp, deve ser copiado o diretório weabb/calcapp, que é onde se encontra a determinada aplicação, ficando a estrutura de diretórios como mostrado na figura abaixo:

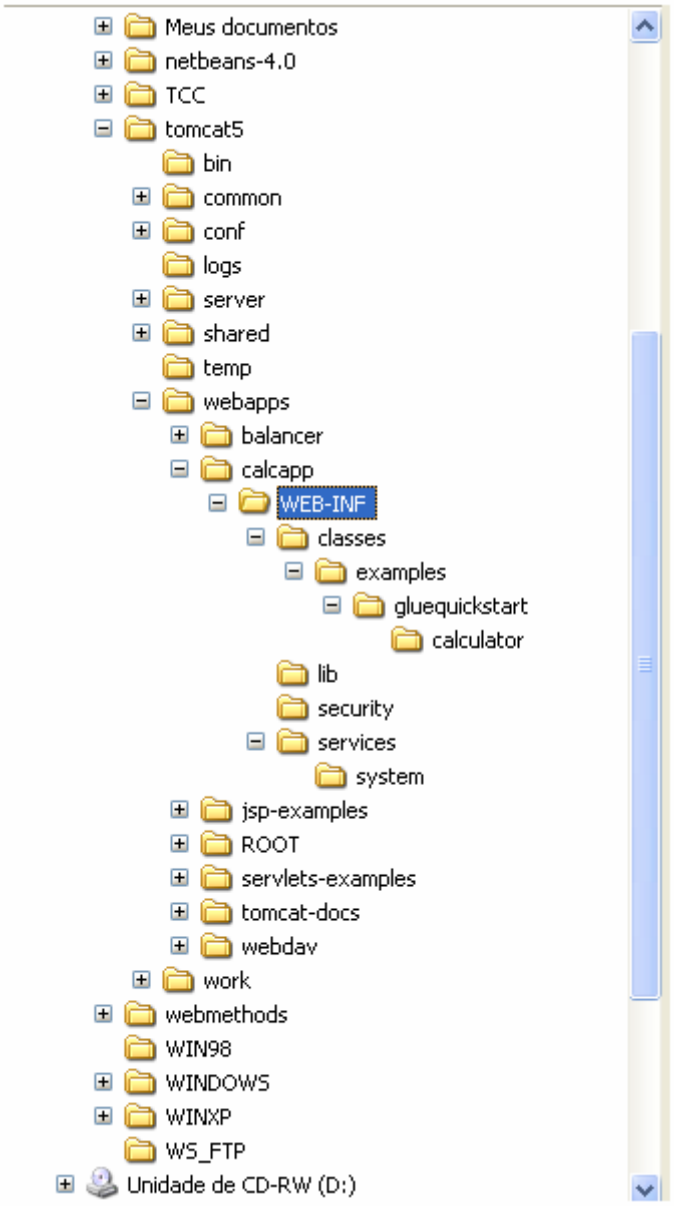


Figura 33 – Estrutura de Diretórios do Servidor Tomcat-GLUE.

No diretório webapp do servidor Tomcat, encontra-se do diretório WEB-INF. Este diretório possui algumas informações de configuração, mas também poderá ser usado para a publicação de web services. Abaixo deste está um diretório /lib que onde deverão ser copiadas as bibliotecas da plataforma GLUE, já mencionadas.

15.4 Verificação da instalação.

15.4.1 Página Inicial.

Com o servidor web iniciado, a página inicial da aplicação exemplo pode ser acessada através do seguinte endereço: http://localhost:8080/calcapp/. Esta página mostrará as boas vindas da plataforma GLUE, dando 3 opções: Uma local e dois links de auxílio ao desenvolvedor : webMethods home page e webMethods interest group .

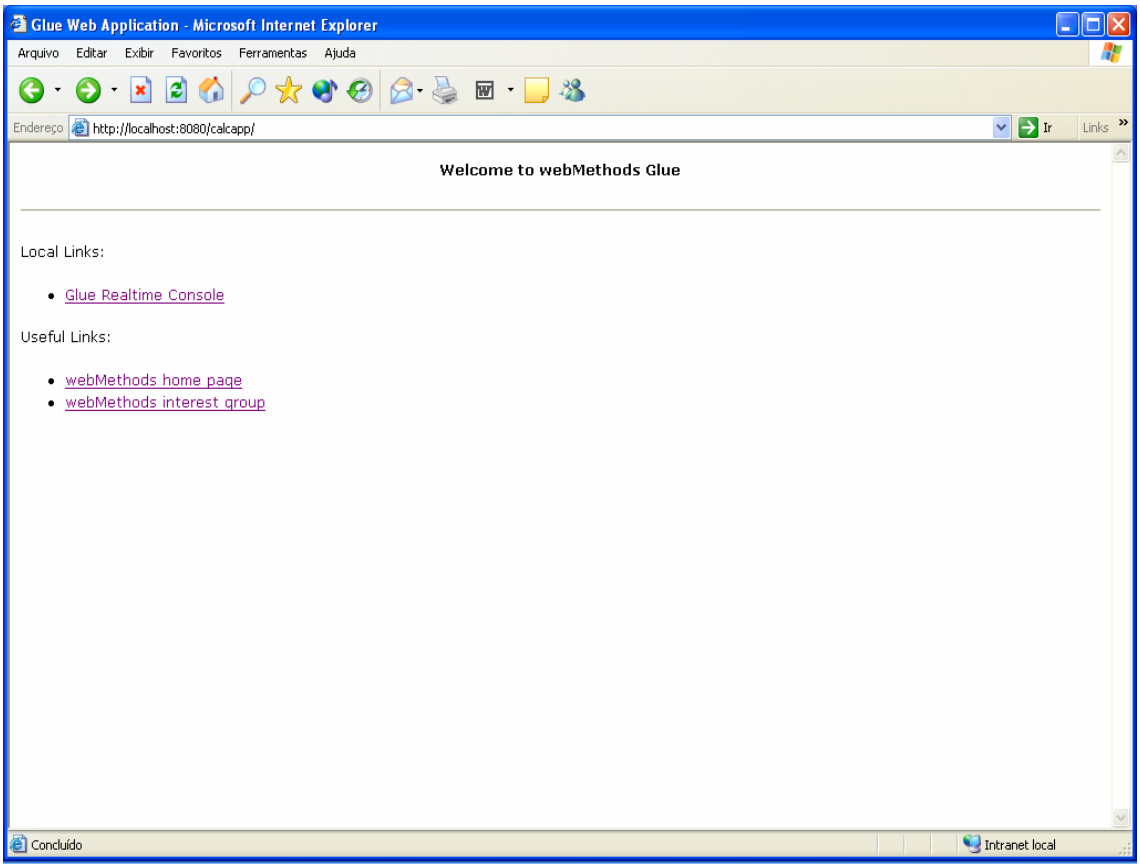


Figura 34 – Página Inicial da Plataforma GLUE.

O link local: Glue Realtime Console, abre uma janela informando que esta modalidade está disponível somente na versão Professional (Figura 18):

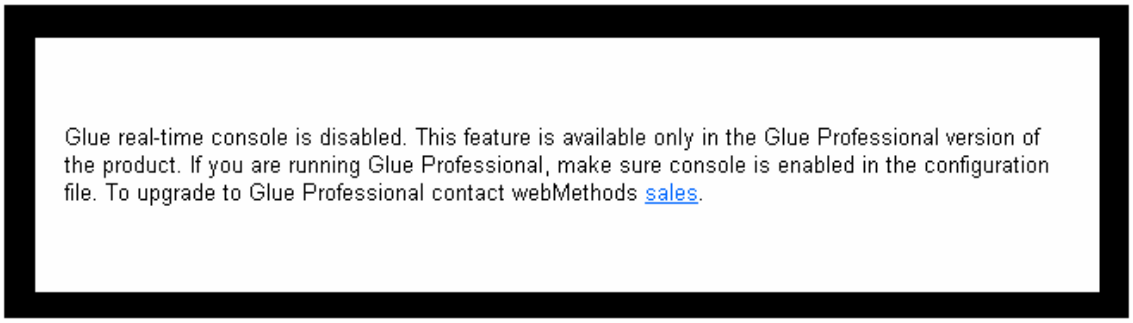


Figura 35

15.4.2 Validação da instalação.

Para fazermos uma validação da instalação, usaremos o seguinte endereço: http://localhost:8080/calcapp/services/calculadora.wsdl.

Aqui acontece um problema de acesso à página da Webmethods através do protocolo de comunicação SOAP. Este acesso é feito para validar a senha que é fornecida na instalação, desta forma ficamos impossibilitados de detalharmos o funcionamento real da plataforma o que diz respeito as mensagens recebidas, nos atendo , portanto, somente o que nos informa a documentação da ferramenta. A figura abaixo (Figura 19), mostra a mensagem que o Tomcat nos dá ao tentar fazer a autenticação do registro junto a empresa fornecedora:

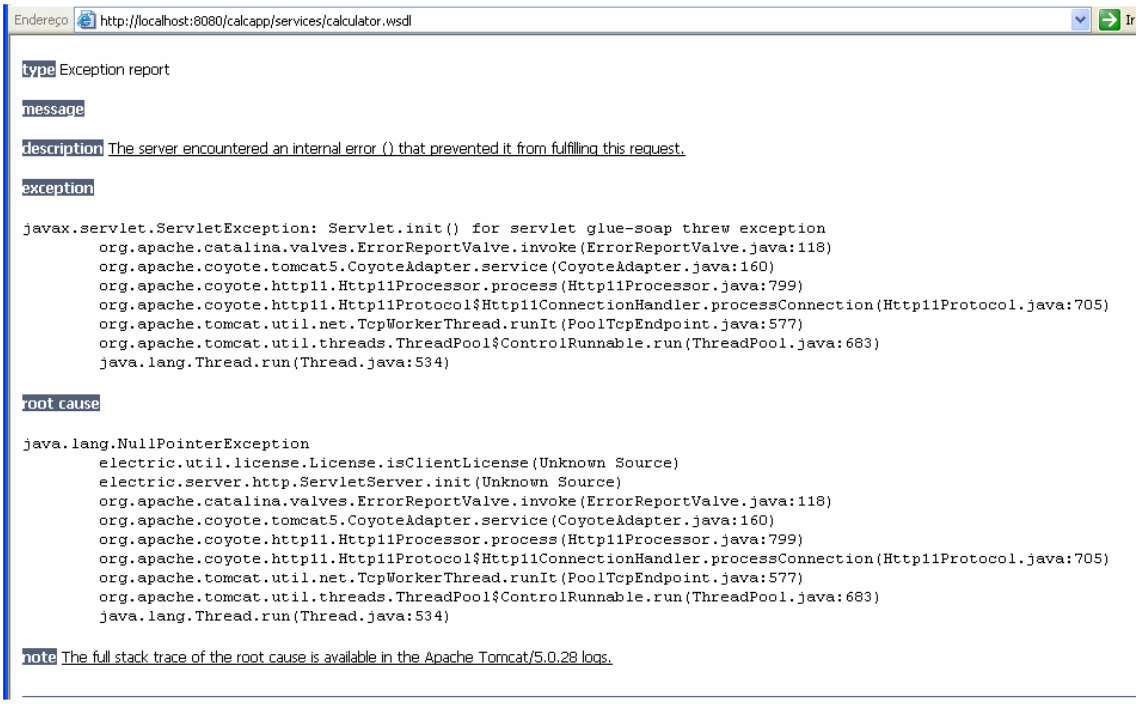


Figura 36

Vale ressaltar que este erro foi reportado ao produtor da plataforma, informado que se trata de um trabalho acadêmico e que tínhamos a necessidade de solucioná-lo, não havendo retorno por parte do mesmo.

15.5 Desenvolvimento de Um Novo Web Service.

Para a instalação de uma nova aplicação, devem ser colocadas no diretório “classes”, sob o diretório WEB-INF do aplicativo /calcapp, no tomcat, as classes compiladas que fazem parte da aplicação.

15.5.1 Variáveis de Ambiente no Cliente.

As classes colocadas no diretório /calcapp/WEB-INF/ formam o serviço web. Neste caminho serão colocadas as classes que deverão sofrer o processo de transformação em um arquivo WSDL para que o mesmo possa ser publicado na WEB. A ferramenta em questão faz este processo através de bibliotecas que o

usuário, ao instalar a plataforma já tem configurado no diretório lib. Existe, entretanto situações que o usuário não terá acesso a uma interface para acessar um serviço. Desta forma, a plataforma oferece via linha de comando uma forma de acessar o serviço remoto. Esta ação é possível através do comando Wsdl2Java. Um exemplo disso pode ser obtido com o seguinte comando:

Quadro 6 - Geração do Código Java a Partir do Arquivo WSDL - GLUE.

```
Wsdl2Java http://localhost:8004/glue/calculadora.wsdl Wsdl2Java
http://localhost:8004/glue/calculadora.wsdl -p
examples.gluequickstart.calcclient
```

Após este comando, infelizmente temos a mesma mensagem de erro visto na forma gráfica de acessar o serviço.

Para que haja o perfeito funcionamento da plataforma as variáveis de ambiente da plataforma são configuradas automaticamente na instalação. A seguir veremos como fazer para confirmar esta situação (Figura 20-24).

Os passos para a configuração das variáveis de ambiente nos S.O. Windows NT/2000/XP são descritos no Anexo II.

15.6 Estudo de Caso de uma calculadora.

15.6.1 Criação e Publicação do Serviço Web.

Serão descritos a seguir os passos necessários para a implementação do estudo de caso da calculadora na plataforma GLUE.

Inicialmente, iremos apresentar como será a classe Calculadora, que será a classe responsável pelo serviço registrado no servidor WEB. Esta classe possui os

métodos para execução de uma adição, subtração, multiplicação e divisão dos parâmetros passados através da implementação de uma classe de interface que veremos a seguir. Segue o código da classe Calculadora:

Quadro 7 - Código da Classe Calculadora.Java - GLUE.

```
public class Calculadora implements ICalculadora {
    /** Creates a new instance of Calculadora */
  public int getSoma(int i1, int i2) {
    return i1 + i2;
  } //getSoma
  //-----------------------------------------
  public int getSub(int i1, int i2) {
    return i1 - i2;
  } // getSub
  //-----------------------------------------
  public int getDiv(int i1, int i2) {
      return i1/i2;
  } // getDiv
  //-----------------------------------------
  public int getMult(int i1, int i2) {
   return i1*i2;
  } //getMult
} //Calculadora
```

A seguir temos a classe ICalculadora. Esta classe é a interface implementada pela classe Calculadora:

Quadro 8 - Código da Interface ICalculadora.Java - GLUE.

```
public interface ICalculadora extends Remote {


  public int getSoma(int i1, int i2);


  public int getSub(int i1, int i2);


  public int getDiv(int i1, int i2);


  public int getMult(int i1, int i2);
}
```

O próximo passo será o “Start” do servidor, seguido pela publicação do objeto. No nosso exemplo uma classe CalcServer será criada. Esta classe publicará a instância de Calculadora e um serviço WEB, que poderá, em nosso exemplo, ser acessado no endereço : http://localhost:8004/glue/calculadora a fim de ser usado por uma aplicação cliente

Quadro 9 - Código da Classe CalcServer - GLUE.

```
import electric.registry.Registry;
import electric.server.http.HTTP;
public class CalcServer
  {
```

```
  public static void main( String[] args ) throws Exception
    {
    HTTP.startup( "http://localhost:8004/glue" );
    Registry.publish( "calculadora", new Calculadora() );
    }
  }
```

Para que o exemplo funcione, o servidor deve ser iniciado. É aí que está o problema que comentamos em itens acima. Quando o servidor está sendo iniciado a fim publicar o serviço, um erro de registro inválido acontece quando o mesmo se comunica com o servidor do fabricante para fazer o registro. Mas seguindo o raciocínio do exemplo, o comando para iniciar ao servidor seria:

Quadro 10 - Inicialização do Servidor - GLUE.

```
> java examples.gluequickstart.calculadora.CalcServer
[STARTUP] Glue (c) 2001-2004 webMethods, Inc.
[STARTUP] soap/http server started on http://127.0.0.1:8008/glue
```

Com este comando o servidor CalcServer seria iniciado e o serviço publicado.

Existe a possibilidade de verificar como a classe Calculadora ficaria em WSDL (Web Service Description Language), onde teríamos um arquivo XML com tal serviço, bastaria iniciar o browser e entrar como o endereço

http://localhost:8004/glue/calculadora.wsdl . GLUE gera dinamicamente o WDSL e o armazena.

Para fazer a ligação de um Web Service publicado dentro de um cliente GLUE é muito simples. O exemplo CalcClient1 abaixo demonstra como fazer esta ligação e invocação, mas de novo atentando ao detalhe de que na prática não conseguimos fazer o registro:

Quadro 11 - Invocação Do Web Service Pelo Cliente - GLUE.

```
import electric.registry.Registry;
public class CalcClient1
 {
 public static void main( String[] args ) throws Exception
  {
  String url = "http://localhost:8004/glue/calculadora.wsdl";
  ICalculadora calculadora = (ICalculadora) Registry.bind( url,
ICalculadora.class );
  System.out.println( "2 + 2 = " + calculadora.soma( 2f, 2f ) );
  System.out.println( "11 - 7 = " + calculadora.subtrai( 11f, 7f ) );
  System.out.println( "12 x 12 = " + calculadora.multiplica 12f, 12f ) );
  System.out.println( "75 / 25 = " + calculadora.divide( 75f, 25f ) );
  }
 }
```

Registry.bind(), retorna um PROXY gerado dinamicamente que implementa a interface especificada, que no nosso caso é a ICalculadora. Esta invocação é entregue ao serviço alvo via protocolo SOAP.

Para testarmos o serviço implementado, executaremos a classe o CalcClient1 enquanto o servidor CalcServer estiver ativo com o comando abaixo:

Quadro 12 - Teste da Classe CalcClient1 - GLUE.

```
> java examples.gluequickstart.calculadora.CalcClient1
[STARTUP] Glue (c) 2001-2004 webMethods, Inc.
2 + 2 = 4.0
11 - 7 = 4.0
12 x 12 = 144.0
75 / 25 = 3.0
> _
```

Se o cliente e servidor já possuírem interfaces JAVA e tipos de dados correspondentes para operarem em Web Service, pode-se utilizar o GLUE apenas com RMI, sem a necessidade de grandes mudanças, apenas utilizando o comando Wsdl2Java para geração do WSDL correspondente.

16 PLATAFORMA JWSDP

16.1 Instalação

A plataforma JWSDP é baseada em um conjunto de tecnologias que tem o objetivo de facilitar o desenvolvimento e distribuição de services web, utilizando a plataforma JAVA 2 (FOSTER, 2002).

As tecnologias que compõe e plataforma são as seguintes:

- Sun Java Streaming XML Parser Version 1.0;
- XML Digital Signature Version 1.0 EA2;
- XML and Web Services Security Version 1.0;
- WS-I Attachments Sample Application Version 1.0 EA3;
- JAXB Version 1.0.4 (Java Architeture for XML Building);
- JAXP Version 1.2.6 01 (Java API for XML Processing);
- JAXR Version 1.0.7 (Java API for XML Registries);
- JAX-RPC Version 1.1.2 01 (Java API for XML-based RPC);
- SAAJ Version 1.2.1 (SOAP with Attachments API for Java);
- JSTL Version 1.1.1 01 (JavaServer Pages Standard Tag Library);
- Java WSDP Registry Server Version 1.0 08;
- Tomcat (Java Servlet and JavaServer Pages container)
- Ant build tool.

Assim com outras plataformas, o pacote para desenvolvimento de web services JAVA (Java WSDP), necessita de um container web. Os container disponíveis são o servidor “Sun Java System Application Server Edition 8”, o servidor Jakarta tomcat 5.0,

O instalador da plataforma JWSDP é disponibilizado em um arquivo executável, **jwsdp-1_5-windows-i586.exe.**

A seguir é descrito o processo de instalação dos servidores e da plataforma JWSDP.

16.2 Containers JWSDP

16.2.1 Sun Java System Application Server Edition 8.

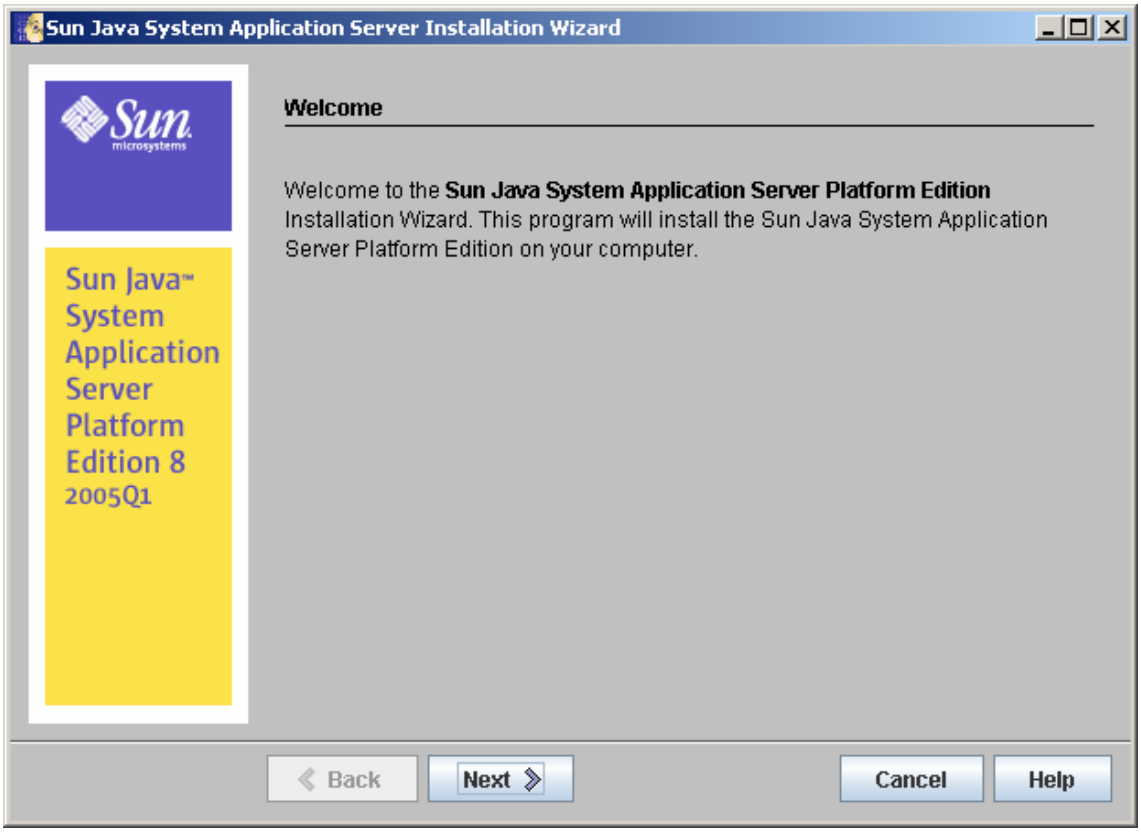


Figura 37 – Instalação do Servidor SJS.

Fonte: Sun Microsystems. Disponível em http://www.sun.com

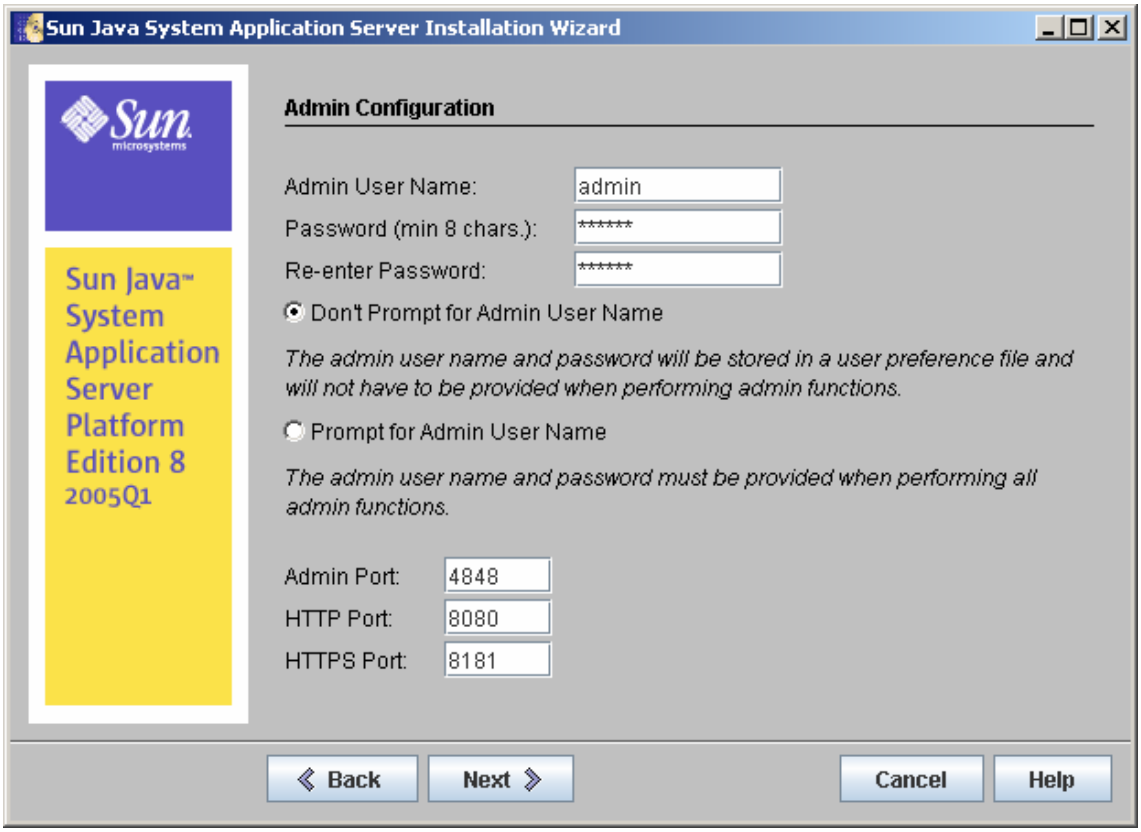


Figura 38 – Parâmetros de Instalação do SJS.

16.2.2 Tomcat 5.0 – JWSDP

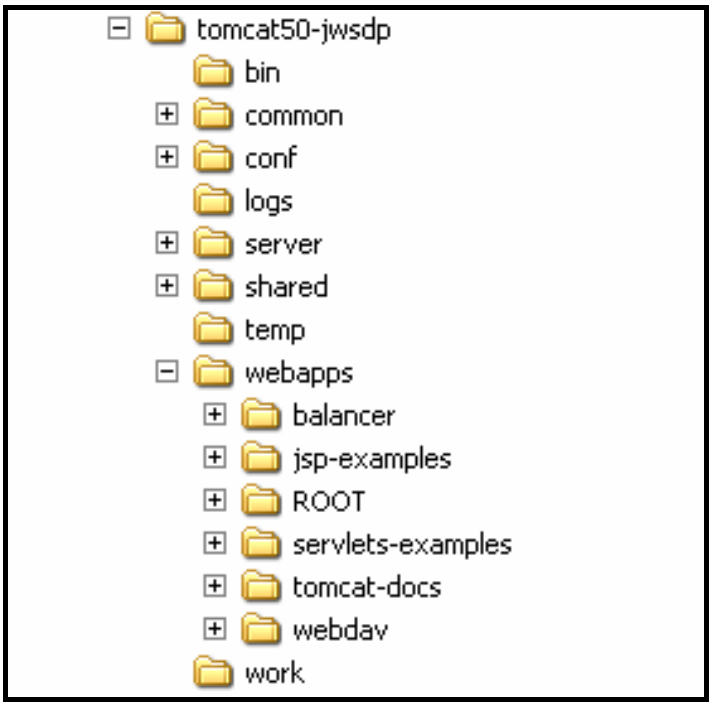


Figura 39 – Estrutura de Diretórios do Servidor Tomcat-JWSDP.

16.3 Estudo de Caso Calculadora.

No caso em estudo, usaremos o container Sun Java System Application Server Edition 8.1 e a IDE NetBeans 4.1RC2.

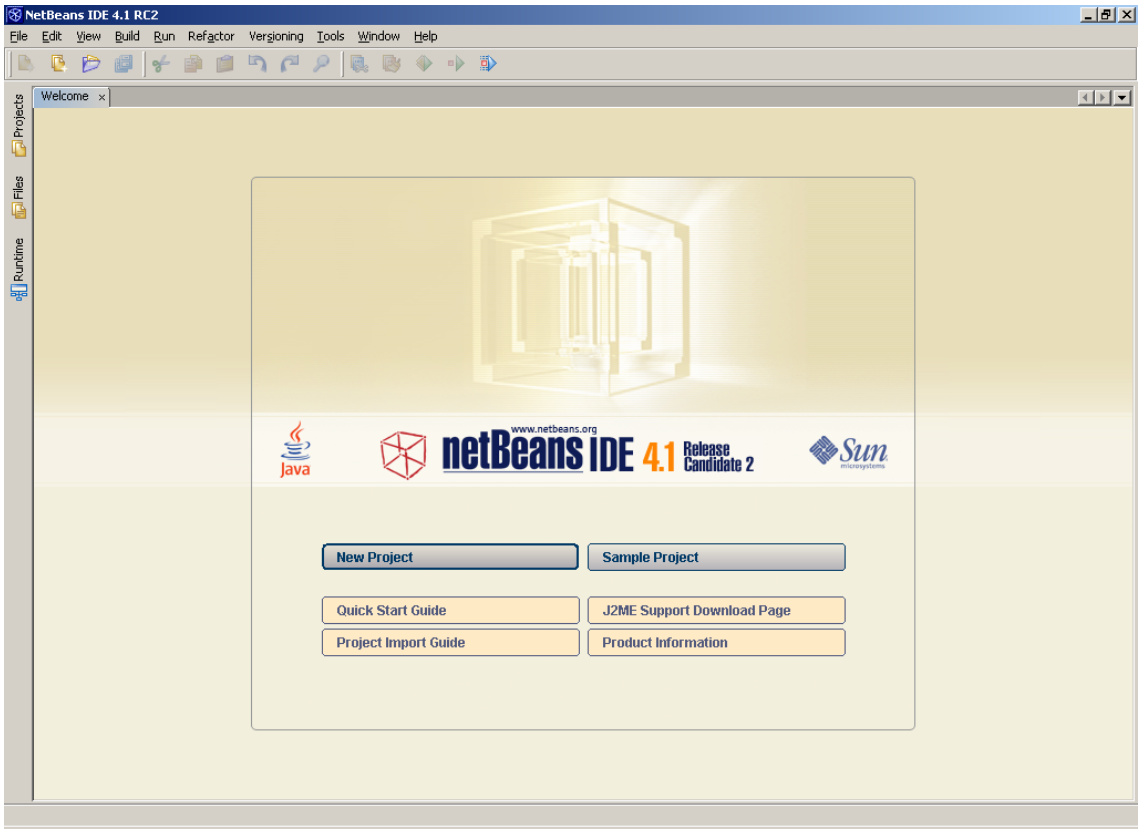


Figura 40 – NetBeans 4.1RC2.

Fonte: NetBeans Comunity. Disponível em http://www.netbeans.org

16.3.1 Registro do Servidor Web.

Para que seja possível a compilação dos serviços web, é necessário o registro de uma instância local do servidor web. Para tanto, deve-se seguir os seguintes procedimentos:

1. Selecionar Menu Tools → Server Manager;
2. Escolher a opção add Server. Selecionar Sun Java Systems Application Server 8.1 e escolher um nome para a instância selecionada;

3. Fornecer as informações do servidor, a localização da instância local e o nome de domínio escolhido.

16.3.2 Criando um Novo Projeto Para a Aplicação Web.

Na barra de opções do NetBeans 4.1 devemos seguir os seguintes passos:

1. Selecionar o menu File → New Project ;
2. Nas opções de tipos de projetos selecionar Web Application;
3. Chamaremos o projeto de calculadoraWS;

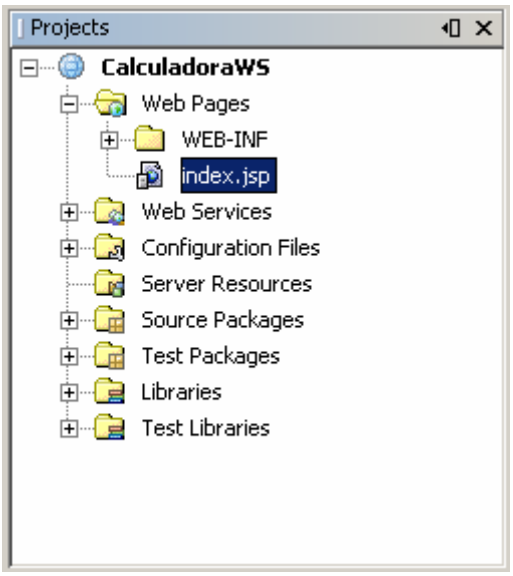


Figura 41 – Projeto CalculadoraWS.

16.3.3 Definição das Bibliotecas.

Com o uso da IDE NetBeans 4.1 não é necessário a configuração do classpath do projeto, para identificação das bibliotecas, uma vez que estas já estão embutidas na IDE.

16.3.4 Criação do Web Service.

Para a criação do serviço web, clicar no nome do projeto e selecionar New → Web Service. Nomear o serviço web como CalculadoraWS. Digitar app.calc.ws no nome do package do projeto.

Observamos que serão criadas automaticamente as classes CalculadoraWSImpl.Java. e CalculadoraWSSEI.Java.

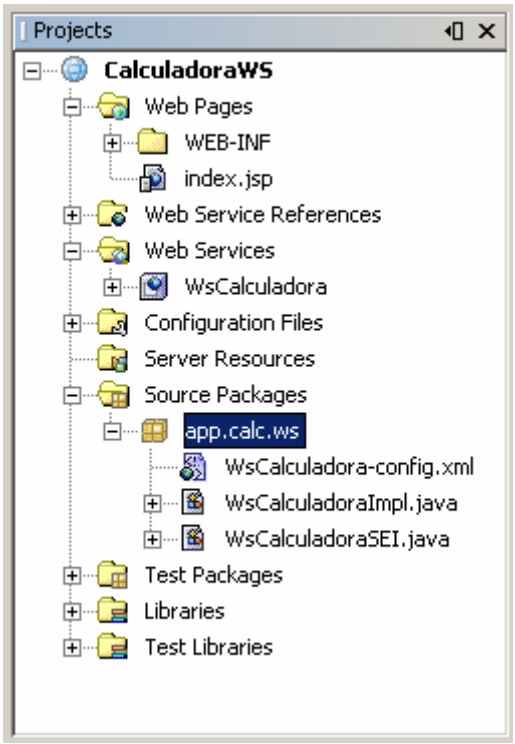


Figura 30 – CalculadoraWSImpl.Java e CalculadoraWSSEI.Java

16.3.5 Definindo os Métodos do Serviço Web.

Na opção Web Services, na janela de projetos, clicas no nome do serviço Web (CalculadoraWS) e selecionar Add Operation. Digitar o nome da operação e o tipo da variável de retorno. Usar a opção Add para incluir os parâmetros do método.

Teremos então o seguinte código gerado na classe interface (CalculadoraWSSEI.Java):

Quadro 13 - Código da Classe Interface - JWSDP.

```
package app.calc.ws;

public interface CalculadoraWSSEI extends java.rmi.Remote {

    /**

     * Web service operation

     */

    public int getSoma(int i1, int i2) throws java.rmi.RemoteException;

    /**

     * Web service operation

     */

    public int getSub(int i1, int i2) throws java.rmi.RemoteException;

    /**

     * Web service operation

     */

    public int getDiv(int i1, int i2) throws java.rmi.RemoteException;


    /**

     * Web service operation

     */

    public int getMult(int i1, int i2) throws java.rmi.RemoteException;

}
```

A IDE criará também o código para a classe CalculadoraWSImpl.Java, como mostrado a seguir:

Quadro 14 - Código da Classe CalculadoraImpl.Java Gerado pela IDE - JWSDP.

```
package app.calc.ws;


public class CalculadoraWSImpl implements CalculadoraWSSEI {


    // Enter web service operations here. (Popup menu: Web Service->Add
Operation)
    /**
     * Web service operation
     */
    public int getSoma(int i1, int i2) {
        // TODO implement operation
        return 0;
    }


    /**
     * Web service operation
     */
    public int getSub(int i1, int i2) {
        // TODO implement operation
        return 0;
    }


    /**
     * Web service operation
```

```
     */
    public int getDiv(int i1, int i2) {
        // TODO implement operation
        return 0;
    }

    /**
     * Web service operation
     */
    public int getMult(int i1, int i2) {
        // TODO implement operation
        return 0;
    }
}
```

Devemos, então, completar o código da classe CalculadoraWSImpl.Java, que ficará da seguinte forma:

Quadro 15 - Código da Classe CalculadoraWSImpl.Java Alterado - JWSDP.

```
package app.calc.ws;


public class CalculadoraWSImpl implements CalculadoraWSSEI {


    // Enter web service operations here. (Popup menu: Web Service->Add
Operation)
    /**
package app.calc.ws;


public class CalculadoraWSImpl implements CalculadoraWSSEI {


    public int getSoma(int i1, int i2) {


        return i1 + i2;
    }
    /**
     * Web service operation
     */
    public int getSub(int i1, int i2) {


        return i1 - i2;
    }
    /**
     * Web service operation
```

```
     */
    public int getDiv(int i1, int i2) {

        return i1/i2;
    }
    /**
     * Web service operation
     */
    public int getMult(int i1, int i2) {

        return i1*i2;
    }
}
```

16.3.6 Geração e Configuração de Um Manipulador de Mensagens SOAP.

Na janela de projetos, clicar sobre o nome do projeto e selecionar a opção New → File/Folder. Em categorias, selecionar Web Services. Em File Types, selecionar Message Handler. Nomear o manipulador de mensagens como CalculadoraWSLogger e digitar no campo package o valor app.calc.ws. Uma classe CalculadoraWSLogger.Java é criada.

Na janela de projetos, clicar na opção Web Services e selecionar a opção Configure Handlers. Será exibida uma tela para a configuração do manipulador de mensagens SOAP, clicar no botão add e selecionar a classe CalculadoraWS.

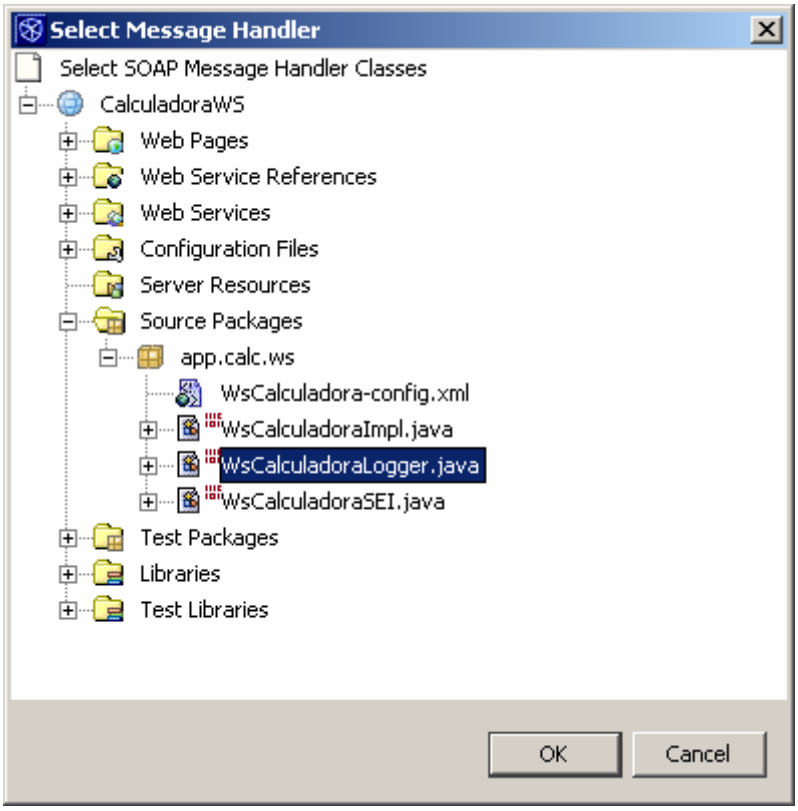


Figura 42 – Manipulador de Mensagens SOAP.

16.3.7 Tornando o Serviço Web Disponível.

Tornar um serviço web disponível, significa colocá-lo a disposição para que seja utilizado pelos clientes.

16.3.8 Deploy do Serviço Web.

Na tela de projetos, clicar no nome do projeto e selecionar propriedades. Na caixa de diálogo de propriedades do projeto, selecionar a opção Run. Digitar /CalculadoraWS nas opções Context Path e Relative URL. Clicar no nome do projeto e selecionar a opção Run Project.

O Servidor Web é iniciado e a seguinte mensagem deve aparecer no browser:

Quadro 16 - Mensagem de Erro no Deploy do Serviço Web - JWSDP.

```
Invalid wsdl request http://localhost:8080/CalculadoraWS/CalculadoraWS for
web service CalculadoraWS
```

16.3.9 Registrando e Testando o Servidor Web.

Na janela de projetos, clicar em Web Services, selecionar o serviço web CalculadoraWS e selecionar a opção Add to Registry.

É importante ressaltar que o servidor web deve estar ativo para execução deste procedimento, caso contrário a seguinte mensagem de erro será mostrada (Figura 31):

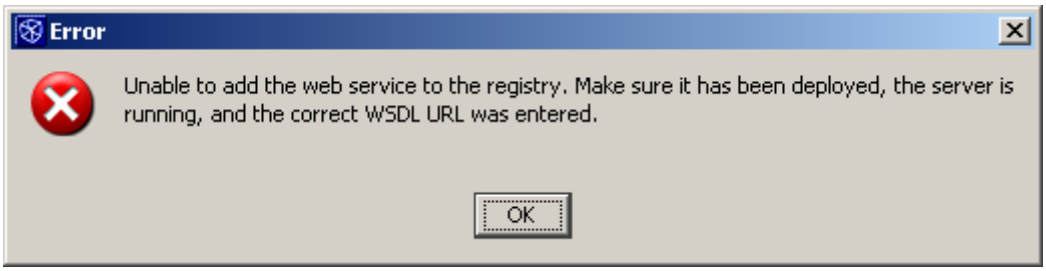


Figura 43

Com o servidor web ativo, será apresentada uma janela com a url utilizada para o deploy do serviço. Este endereço será utilizado posteriormente quando da criação do serviço web do cliente.

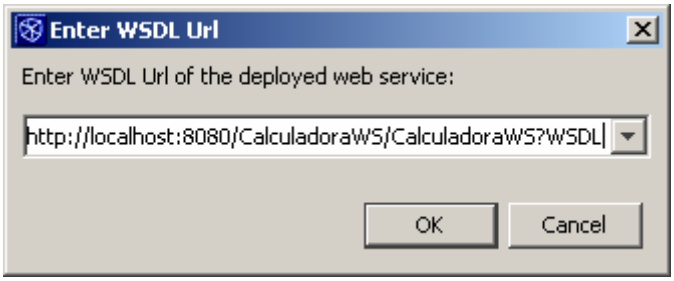


Figura 44 - URL WSDL.

Na janela Runtime, expandir a opção Web Services até encontrar a opção com o nome do método que se quer testar. Clicar no nome do método e escolher a

opção Test Operation. Uma janela de diálogo surgirá (Figura 55) com os parâmetros do método. Preencher os valores e clicar no botão Submit. No painel Resulta deverá aparecer o valor resultando do método.

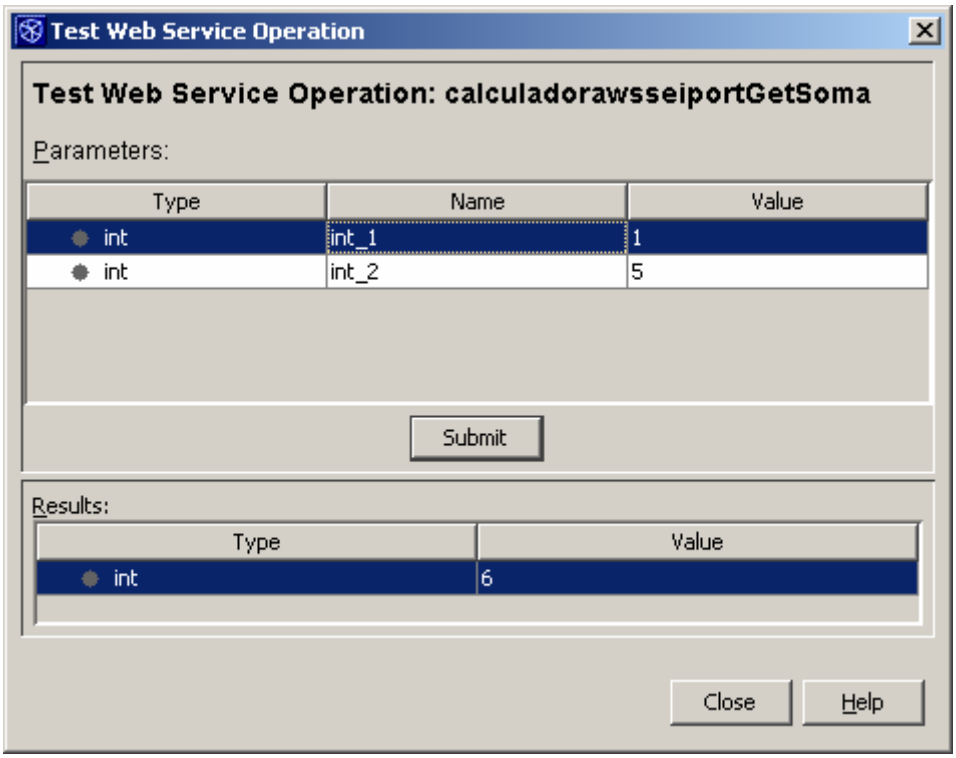


Figura 45 - Teste de Operação do Web Service.

Na janela Runtime, expandir a opção Sun Java System Application Server 8.1 e selecionar View Server Log. Será mostrada a seguinte mensagem:

Quadro 17 - Log do Servidor SJS - JWSDP.

```
Log message: Sun May 08 09:55:25 BRT 2005--getSoma int_1:1 int_2:5 |#]
```

Esta mensagem é gerada pelo manipulador de mensagens SOAP, que foi criada anteriormente.

16.4 Utilizando o Web Service.

Na janela de projetos, seleciona no menu a opção File → New Project. Em categorias selecionar Web. Em projetos selecionar Web Application. Digitar CalculadoraCliente no campo Project Name. No campo Server deve ser selecionado o servidor Sun Java System Application Server 8.1.

16.4.1 Descobrindo Informações do Web Service.

Clicar no nome do projeto, na janela Projects e escolher New → Web Service Client. No campo WSDL URL digitar o endereço do serviço web ativo, no caso em questão (http://localhost:8080/CalculadoraWS/CalculadoraWS?WSDL) , e clicar no botão Retrieve WSDL. Preencher o campo Package com o nome do pacote usado (app.ws.calc).

Na janela de projetos, expandir a opção Web Services References até encontrar os métodos do web service. Selecionar um dos métodos e executar a opção Text Operation. Uma janela de diálogo surgirá (Figura 56) com os parâmetros do método. Preencher os valores e clicar no botão Submit. No painel Result deverá aparecer o valor resultando do método.

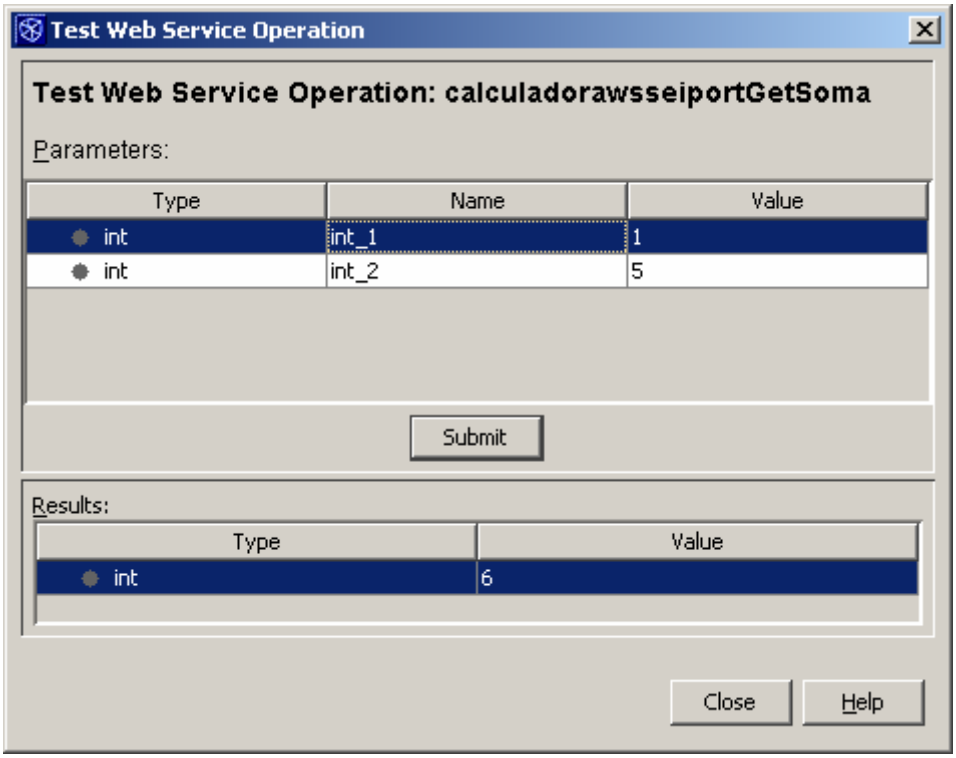


Figura 46 - Teste de Operação do Web Service Cliente.

Pode-se observar que o procedimento acima é idêntico ao executado na classe CalculadoraWS, porém a execução dos métodos da classe é executada através do cliente, acessando a classe CalculadoraWS no servidor.

No nome do projeto, selecionar a opção New → Servlet. No campo Class Name, digitar CalculadoraServlet e no campo Package, digitar app.ws.calc.

Na opção Configure Servlet Deployment, preencher o campo Servlet Name com CalculadoraServlet e URL Mapping(s) com /CalculadoraServlet.

Na janela de edição da IDE, será mostrado o código do Servlet.

16.4.2 Criando o Cliente do Web Service.

Na janela do editor clicar dentro do código do método processRequest. Selecionar a opção Web Service Client Resources → Call Web Service Operation, selecionar uma das operações, no caso em estudo, getSoma.

O seguinte código será incluído no método processRequest:

Quadro 18 - Código do Método "processRequest" do Servlet Gerado Pela IDE - JWSDP.

```
try {
        getCalculadoraWSSEIPort().getSoma(/* TODO enter operation
arguments */);
    } catch(java.rmi.RemoteException ex) {
        // TODO handle remote exception
    }
```

Pode-se observar que o método getSoma não possui os parâmetros necessários para sua execução, assim como não existe o código para tratamento da exceção na execução do método. Portanto o código deve ser alterado e ficará da seguinte forma:

Quadro 19 - Código Alterado do Método "processRequest" do Servlet - JWSDP.

```
try {          out.println(getCalculadoraWSSEIPort().getSoma(5, 2));
    } catch(java.rmi.RemoteException ex) {
        out.println("Erro --> :" + ex);
    }
```

16.4.3 “Empacotando” a Aplicação.

A aplicação pode ser empacotada em um arquivo EAR e então feita seu deploy. Opcionalmente o web service e o web service do cliente podem ser distribuídos separadamente.

16.4.4 Criando uma Aplicação J2EE.

No menu da IDE, selecionar a opção File → New Project. Em categorias, selecionar Enterprise. No campo Project Name digitar CalculadoraApp.

Na janela de projetos, clicar no nome do projeto e selecionar a opção J2EE Modules. Selecionar Add J2EE Module

A aplicação J2EE deverá apresentar a seguinte estrutura:

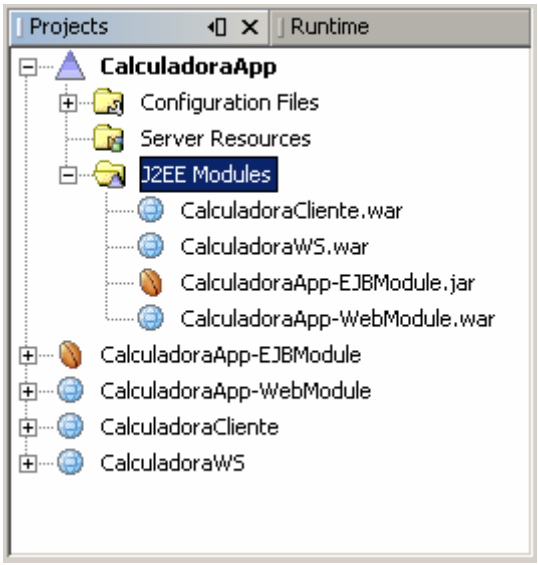


Figura 47 - Aplicação J2EE.

16.4.5 Deploy da Aplicação J2EE.

Na janela de projetos, clicar no nome do projeto e selecionar a opção Properties. Na janela de diálogo das propriedades do projeto, selecionar a opção Run. No campo Cliente Modulo URI, selecionar a opção CalculadoraCliente.war e digitar CalculadoraServlet no campo Relative URL.

Clicar no nome do projeto e selecionar a opção Run Project.

É possível que a seguinte mensagem de erro apareça:

Quadro 20 - Mensagem de Erro no Deploy da Aplicação J2EE - JWSDP.

```
C:\Aplic\CalculadoraApp\nbproject\build-impl.xml:227:
Deployment application in domain failed; Cannot deploy. Application already exists.
```

Caso isto ocorra, deve-se seguir o seguinte procedimento:

1. Na janela Runtime, selecionar a opção Server Registry → Sun Java System Application Server 8.1.;
2. Expandir as opções Enterprise Applications e Web Applications;
3. Se as opções CalculadoraApp, CalculadoraWS ou CalculadoraCliente aparecerem na listagem, executar o undeploy de cada uma delas;
4. Executar novamente a opção Run Project.

Neste ponto, com o servidor web em funcionamento o resultado obtido através da execução do método deve aparecer na janela Output da IDE.

Também pode ser utilizada uma interface para a entrada dos parâmetros dos dados, como uma página JSP

17 PLATAFORMA EMERGING TECHNOLOGIES TOOLKIT (ETTK) .

A IBM tem sido de grade importância para evolução do Web Service. Participou de vários projetos envolvendo SOAP, criação de tecnologias com padrão gráfico, etc. Desta forma podemos dizer que para a elaboração de um Web Service com o conjunto de ferramentas IBM. Iremos utilizar além do ETTK, também o IBM Rational Software Development Plataform, que chamaremos de RAD. O RAD é um ambiente de desenvolvimento JAVA de serviço completo baseado na tecnologia utilizada para o eclipse. O que percebemos com facilidade do uso do Eclipse como código base, é em relação aos assistentes encontrados para criação dos projetos desejados.

A verão do ETTK que estaremos avaliando é a 2.2 e a da ferramenta RAD a 6.0.0

17.1 Instalação.

Para o correto funcionamento de nosso exemplo, baseado no funcionamento de uma calculadora simples, utilizará o ETTK em conjunto com o RAD. A seguir descreveremos os passos para a instalação do ETTK e logo após descreveremos da ferramenta RAD.

17.1.1 ETTK

1 – Uma tela inicial é aberta, com a apresentação do ETTK e suas funcionalidades. Sendo que para seguir o usuário deverá clicar no NEXT.

2 – Nesta janela, Configure Application Server, será configurado a Port number e o HostName. Para estes dois requisitos foram configurados 4400 e localhost

respectivamente.

3 – A próxima janela a ser aberta, Configure Proxies. Nesta janela abre um aviso informando que somente necessita informar os dados se o equipamento estiver conectado a um proxy, como não é o caso, avançamos com NEXT.

4 – A seguir é só finalizar a instalação e reiniciar o equipamento que o ETTK estará completamente instalado no diretório C:\ETTK.

17.1.2 RAD

Antes de iniciarmos os passos de instalação do RAD, lembramos que para que se possa instalar com excelência esta ferramenta, vários arquivos devem ser baixados do site da IBM e estes arquivos são muito grandes. Arquivos estes que chegam a mais de 1 Gbyte de tamanho e portanto o usuário deverá necessariamente estar conectado a uma Internet de banda larga.

1 – Antes de instalar propriamente a ferramenta, na documentação do ETTK, sugere que esta seja instalada em um diretório separado para que os projetos criados sejam armazenados no mesmo e não haja confusão de caminho. Para isso criaremos, conforme orientação, o diretório C:\Workspaces\ettk.

2 – Após esta etapa de direcionamento para este diretório ignorando o padrão sugerido na instalação, o usuário deverá selecionar o botão NEXT para prosseguir.

3 – Após muitos minutos de descompactação dos arquivos de instalação, abre uma janela onde o usuário deverá optar por instalar o Rational, pois outros aplicativos também são oferecidos.

4 – O aplicativo é instado e com a grande vantagem de já estar adequado ao idioma local.

Como a ferramenta RAD nos que diz respeito ao servidor é construída sobre a base dos produtos Apache, em nosso exemplo utilizaremos o servidor e um mecanismo SOAP próprio, que estão integrados à interface do usuário, facilitando muito o seu uso.

17.2 Criando um novo Web Service

17.2.1 Um novo projeto

Como a ferramenta em questão é baseada em JAVA, nosso exemplo será sustentado na criação de uma classe do padrão Java Bean básico e suas variáveis privadas seguidas de seus métodos Get e Set, sendo que os demais métodos públicos utilizados serão criados pela ferramenta.

O primeiro passo é iniciar a ferramenta e acessar o menu Novo → Projeto, como mostra a figura abaixo:

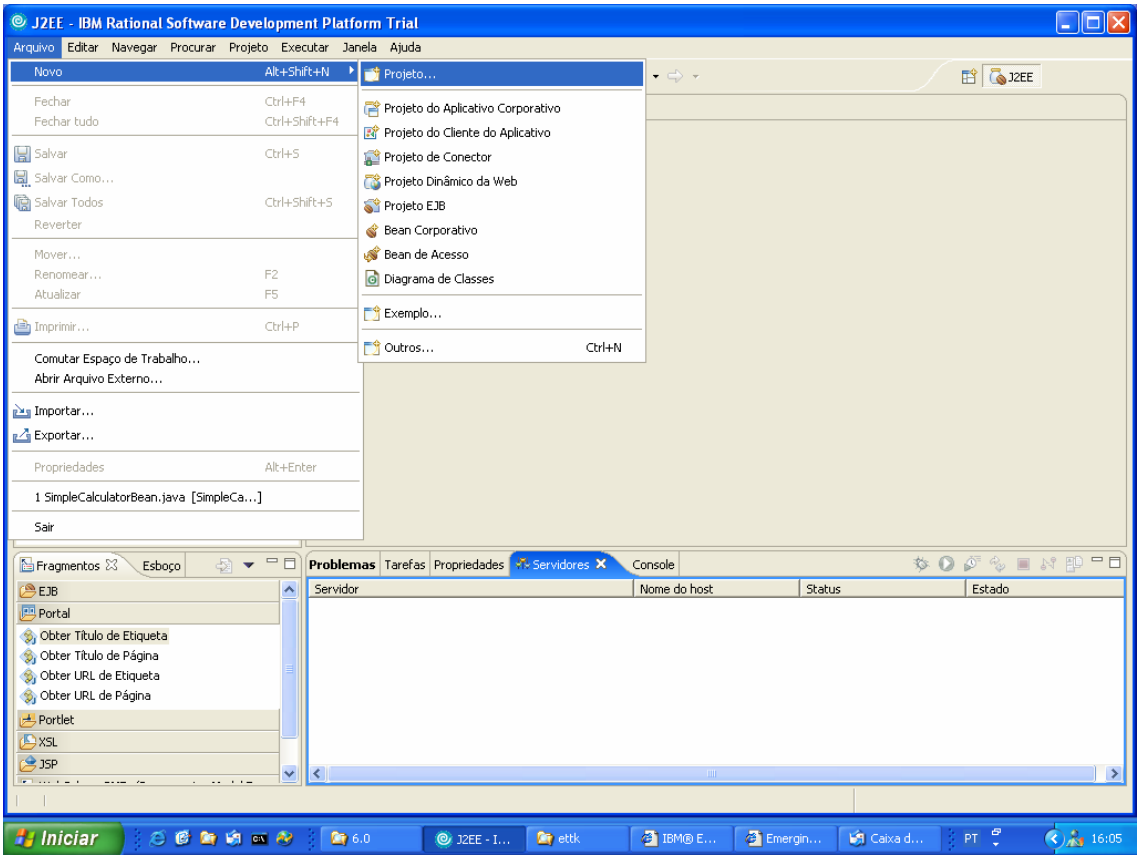


Figura 48 - Criação de Novo Projeto - ETTK.

Será aberta uma nova janela para o usuário escolher qual o tipo de projeto deseja iniciar. Como iremos utilizar para teste uma página JSP, sendo que somente depois iremos criar o projeto Web Service propriamente dito, escolheremos a opção projeto Dinâmica Web, e clicamos em avançar como mostra a figura a seguir:

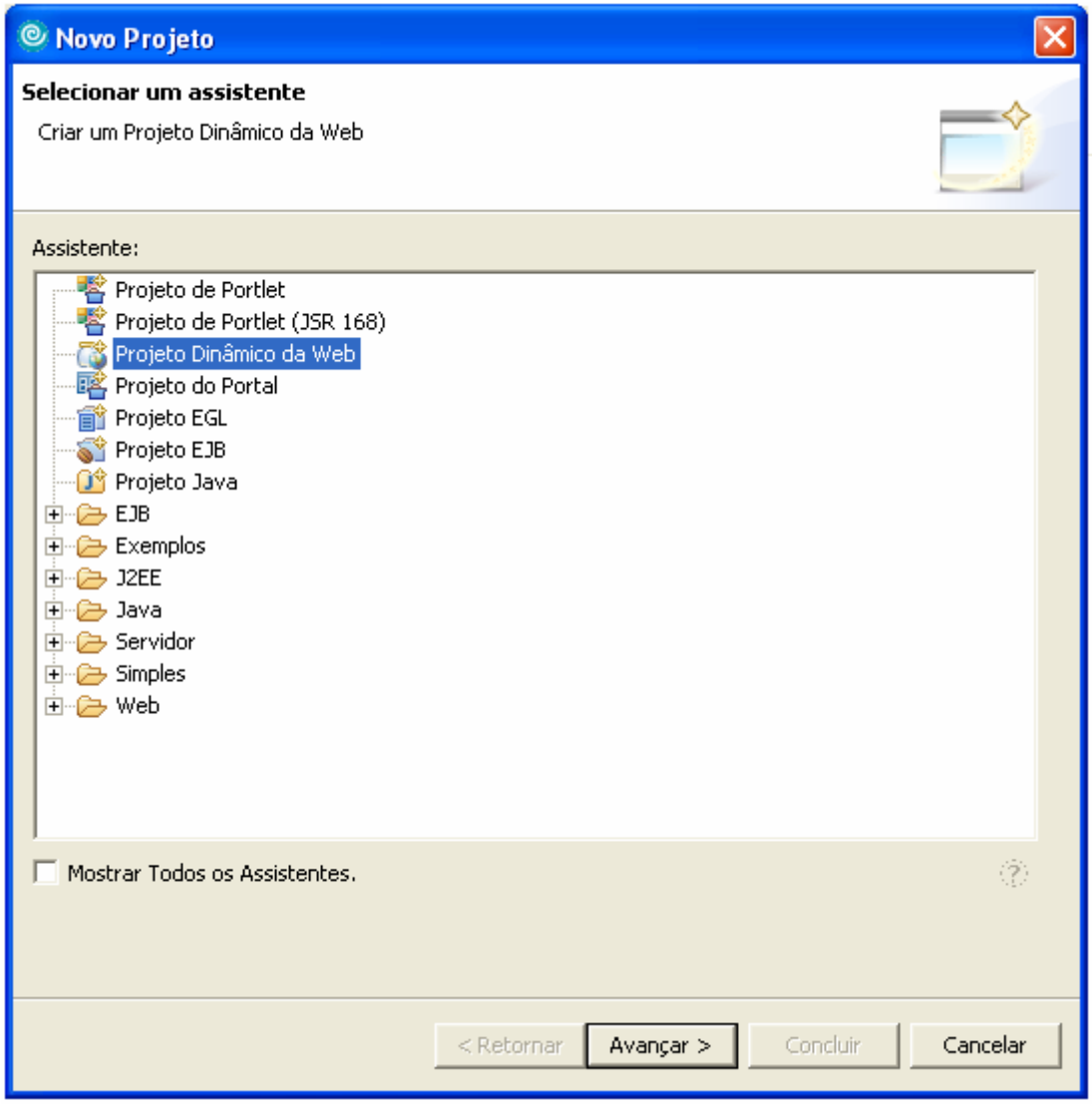


Figura 49 - Definição do Tipo de Aplicação - ETTK.

Com isso, podemos criar nosso projeto, pois a ferramenta exige a criação de um novo projeto, onde serão armazenados todos os arquivos pertinentes ao mesmo, como classes, arquivos de configurações, etc.

Em nosso exemplo será criado o projeto SimplesCalculadora, sendo que os demais campos a ferramenta já preenche de acordo com o nome dado. O projeto EAR será utilizado para gerenciar as instâncias do servidor, este será criado automaticamente, sendo assim serão criados 2 projetos, Web e EAR. Após isso o

usuário deverá simplesmente dar avançar para prosseguir. Este procedimento é mostrado a seguir:

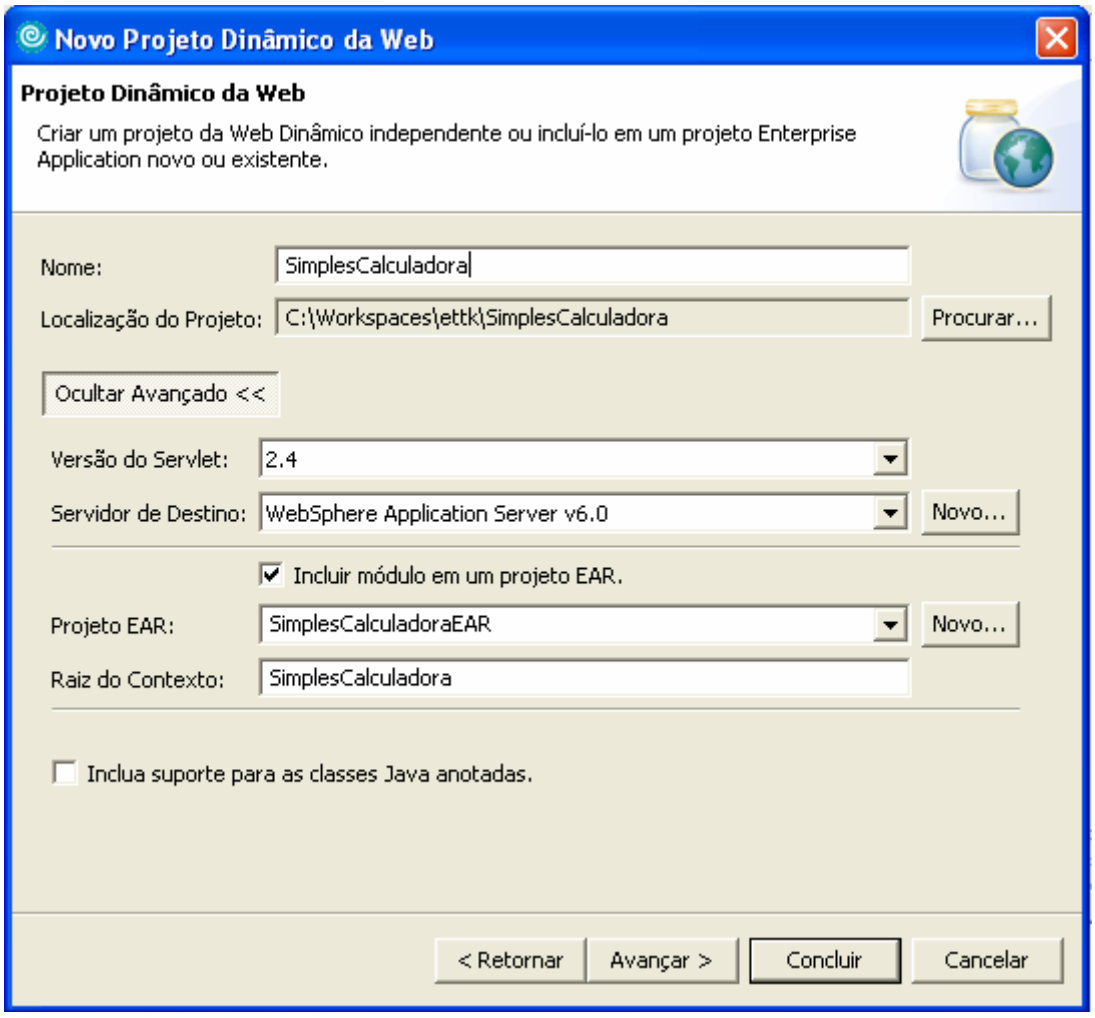


Figura 50 - Definição do Projeto EAR - ETTK.

A janela a seguir mostra somente os recursos que serão utilizados pela aplicação, sendo que devemos aceitar o padrão sugerido e avançar:

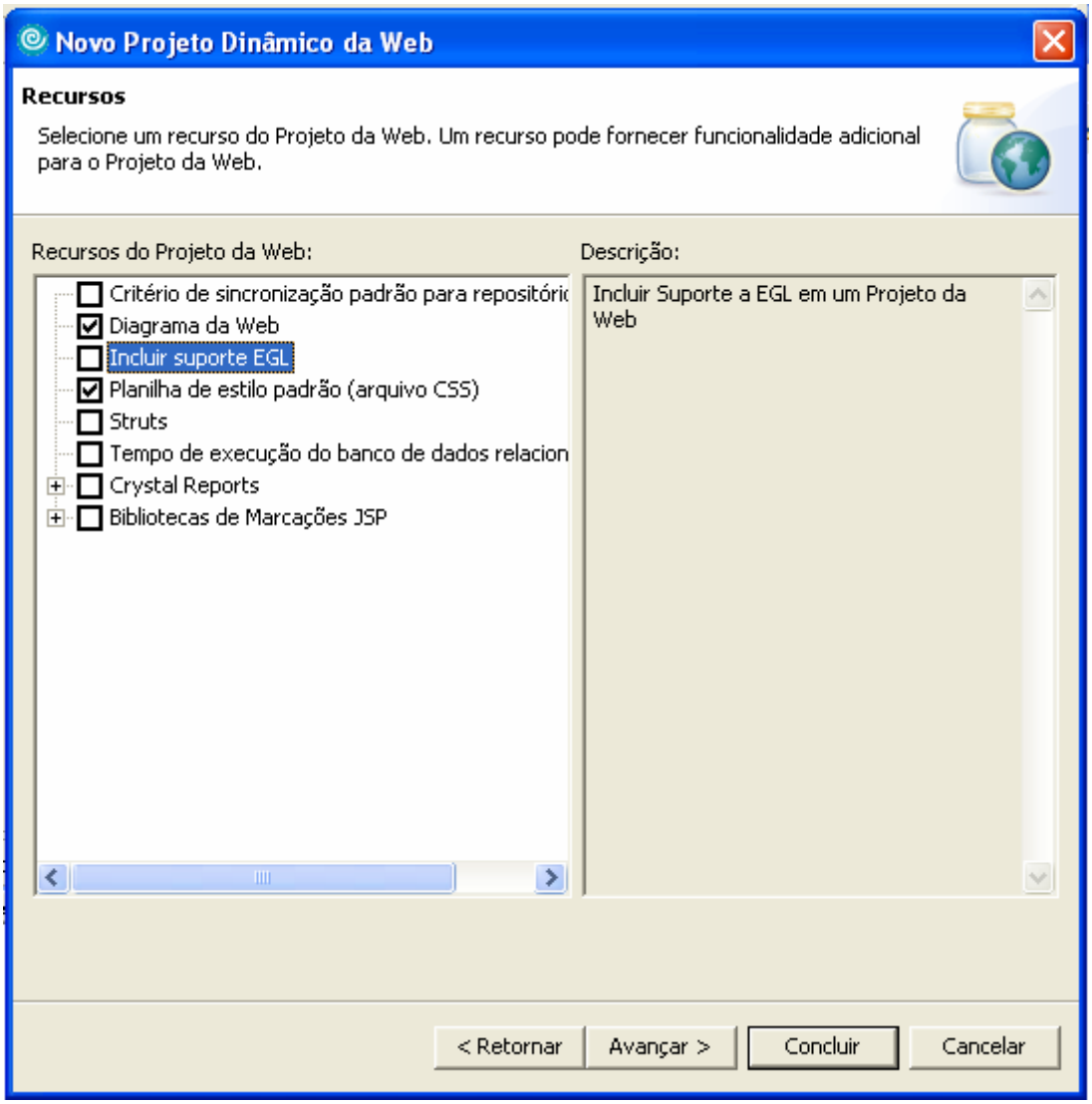


Figura 51 - Display dos Recursos do Projeto - ETTK.

Para encerrar a confecção da aplicação Web, iremos selecionar o gabarito padrão que será utilizado. De acordo com a seleção assumida a página de resultados terá sua aparência:

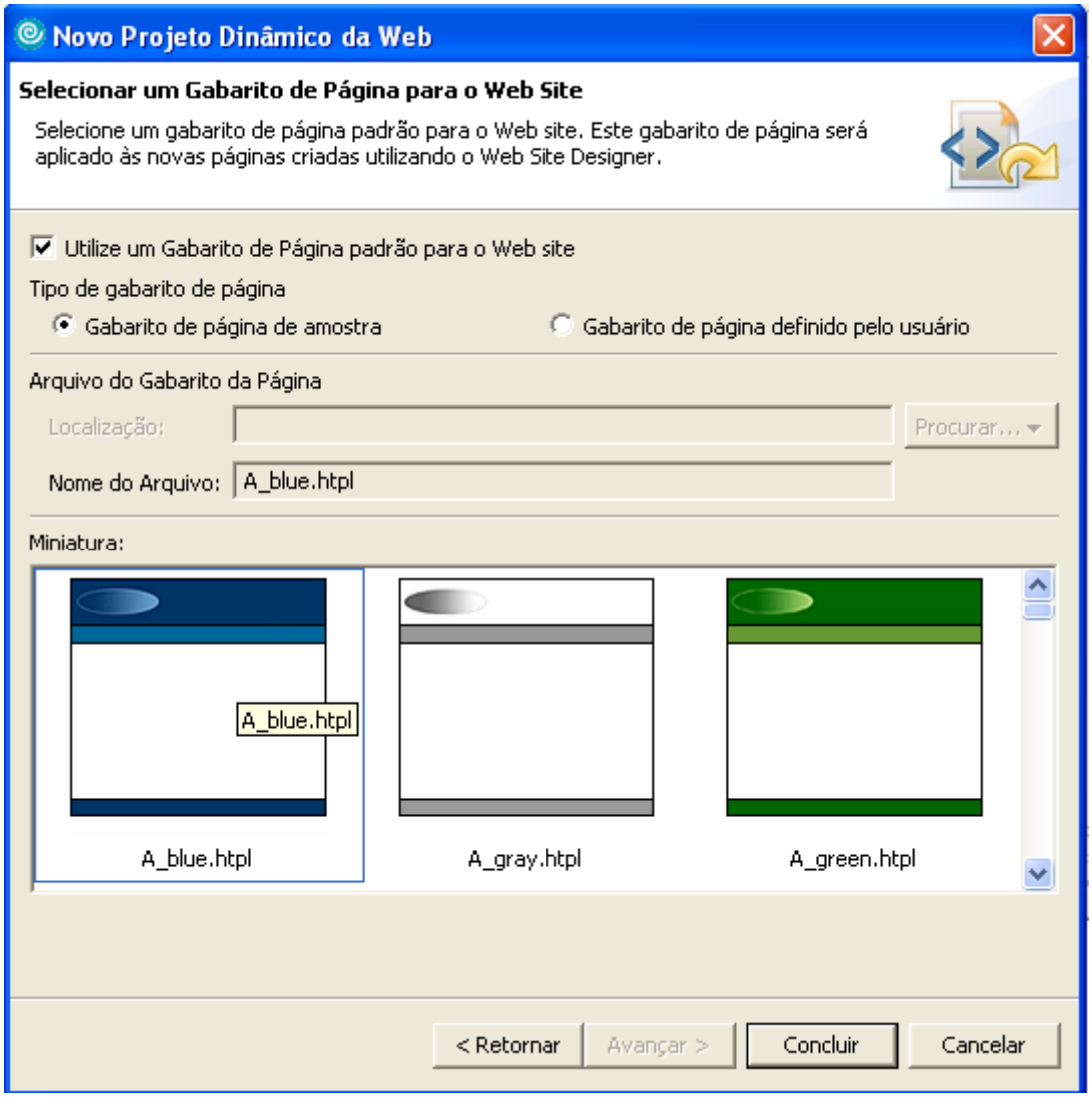


Figura 52 - Definição do Gabarito da Interface da Aplicação Web - ETTK.

Depois de criado o projeto Web, este aparece no lado direito com o nome de SimplesCalculadora. Com isso, o usuário deverá acrescentar duas bibliotecas que são necessárias para o momento em que este projeto será utilizado na construção do Web Service. Clicando com o botão direito do mouse em cima do projeto, selecione propriedades no menu de contexto. Após isto, deverá ser selecionado caminho de construção JAVA, que é um conjunto de arquivos JAR que serão utilizados para o projeto em questão. Neste local deverão ser adicionadas duas

bibliotecas externas com o botão incluirJarsExternos: Xerces e SOAP.JAR. A seguir a ilustração deste procedimento:

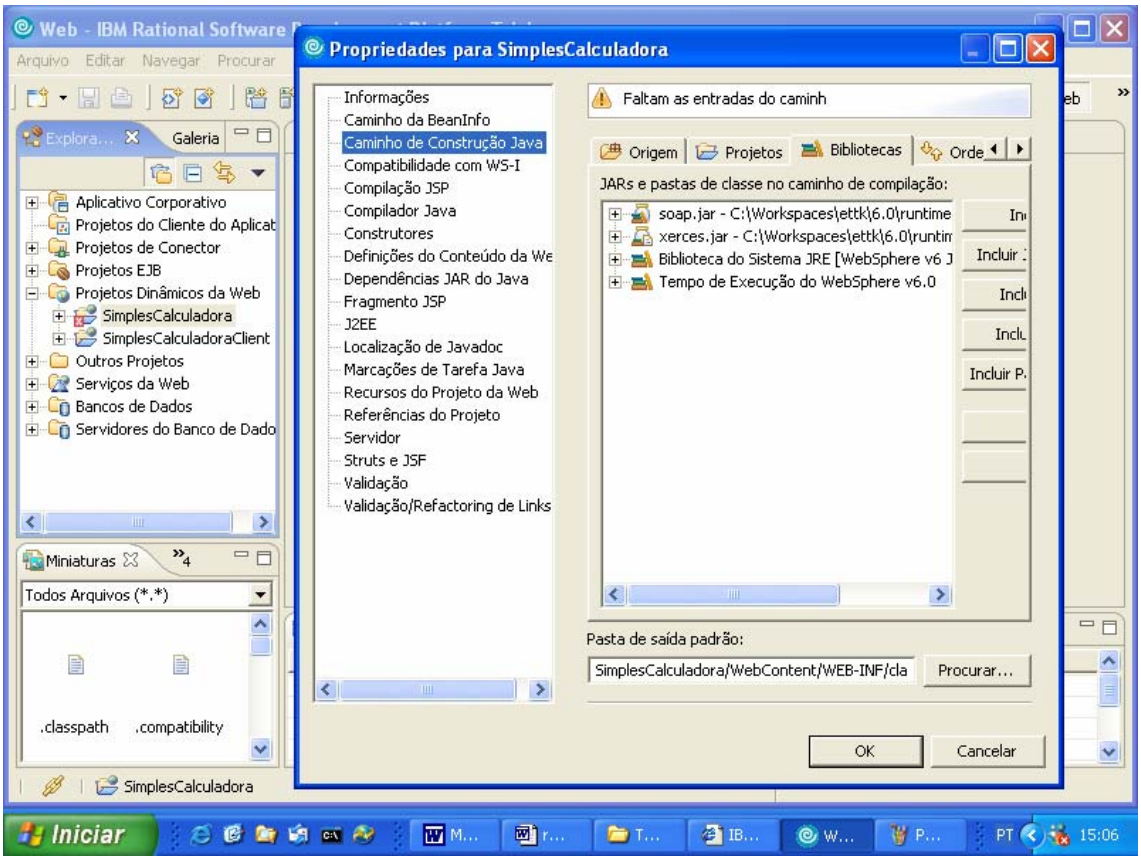


Figura 53 - Definição das Bibliotecas - ETTK.

17.3 Criando o servidor ETTK

Para que possamos utilizar a ferramenta RAD com o servidor ETTK, devemos fazer a integração de ambos. Apara isso temos que ir ao diretório C:\ETTK\BIN e executar o script dos (Arquivo bat) startserver, como mostra a figura abaixo:

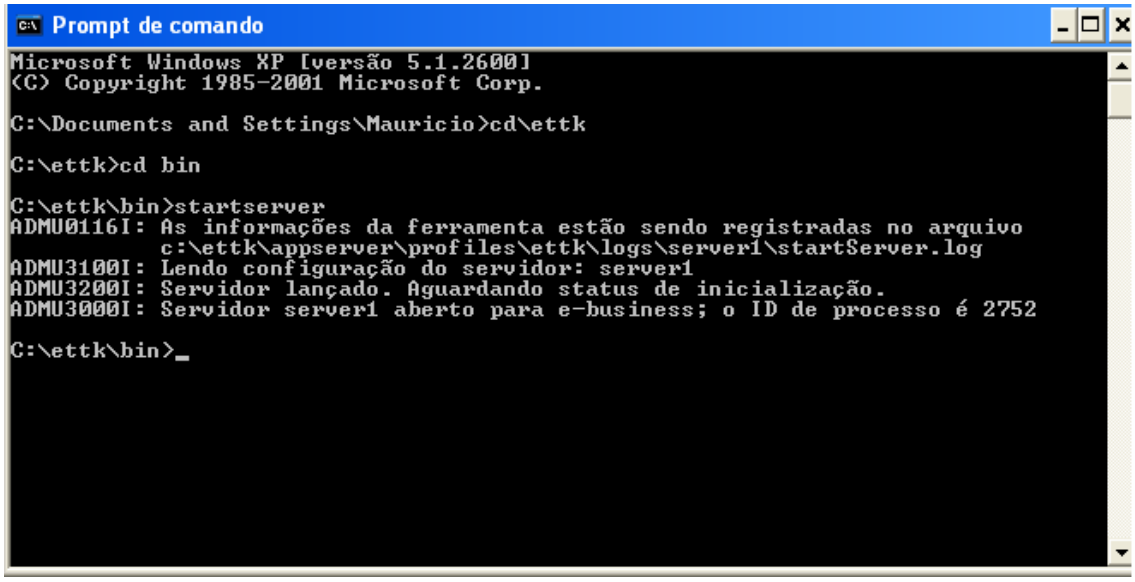


Figura 54 - Iniciando o Servidor Web - ETTK.

Feito isso, devemos ir janela → preferências → Tempo de execução instalados e incluir o novo servidor, como mostra figura abaixo :

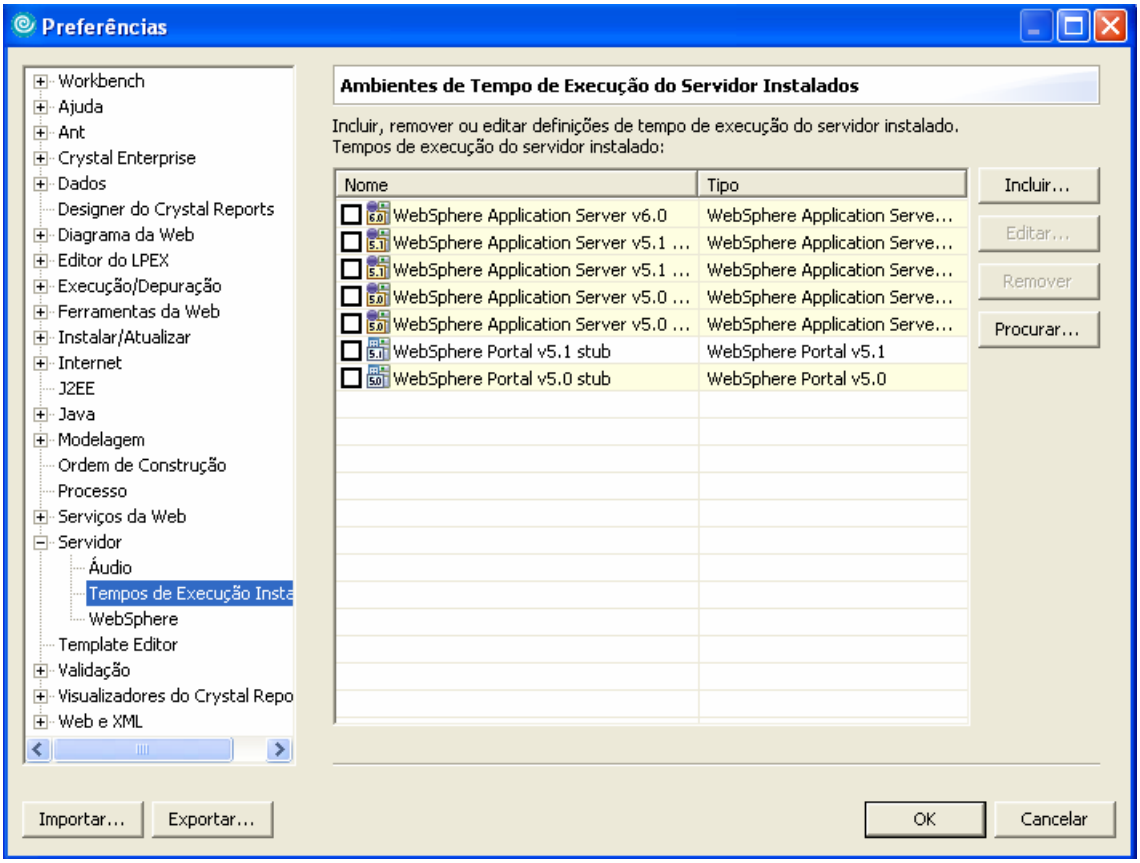


Figura 55 - Inclusão do Servidor Web ao Projeto - ETTK.

Clicando em incluir aparecerá a próxima janela, onde deveremos selecionar o tempo de execução do novo servidor, que será o WebSphere:

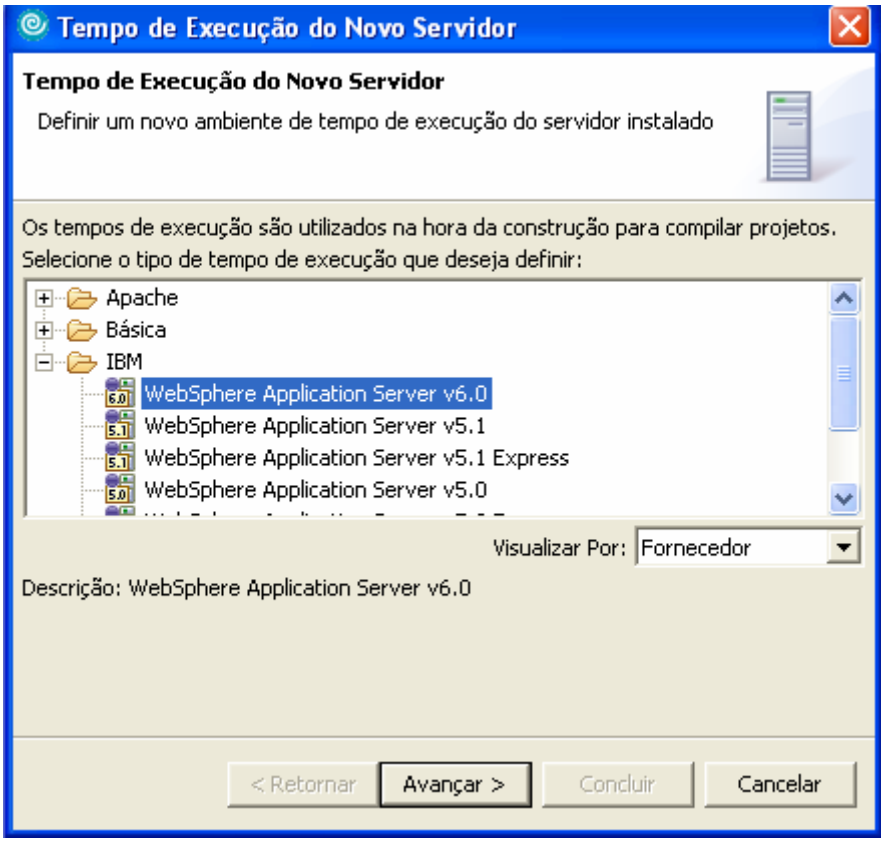


Figura 56 - Definição do Servidor Web - ETTK.

A seguir, devemos inserir o servidor conforme mostra a figura :

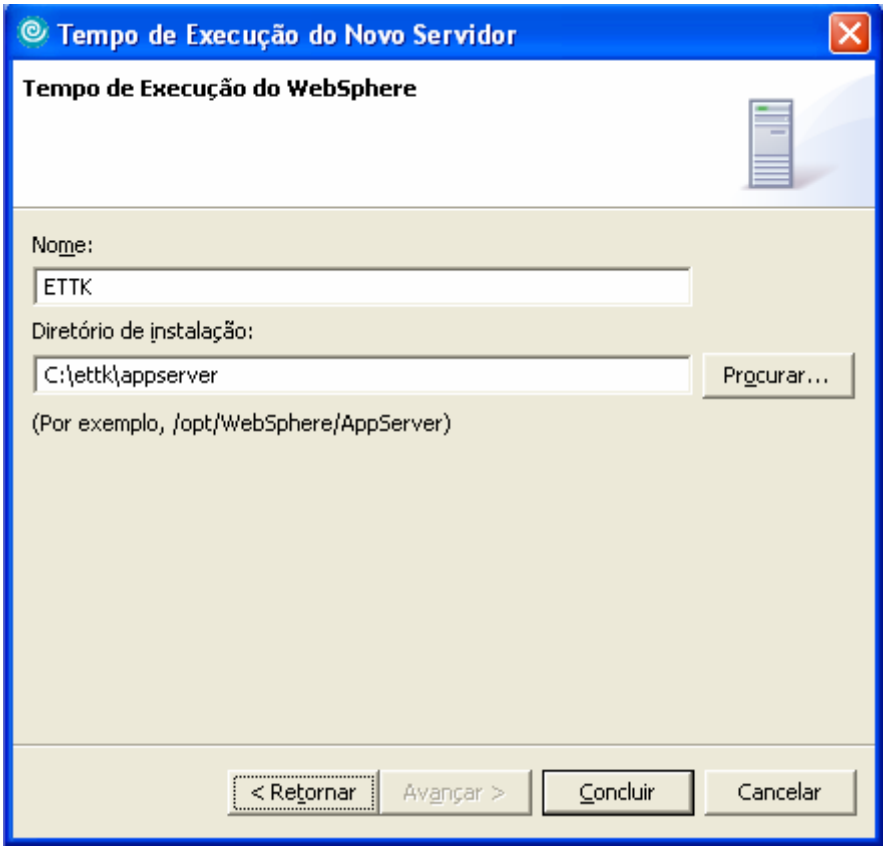


Figura 57 - Definição do Diretório do Novo Servidor - ETTK.

Concluindo este processo, teremos o servidor o ETTK integrado ao RAD. Agora só temos que iniciar o servidor conforme indicado abaixo. O servidor a ser iniciado deve ser o mesmo que foi integrado ao ETTK, no caso Servidor Websphere 6.0 localhost conforme abaixo:

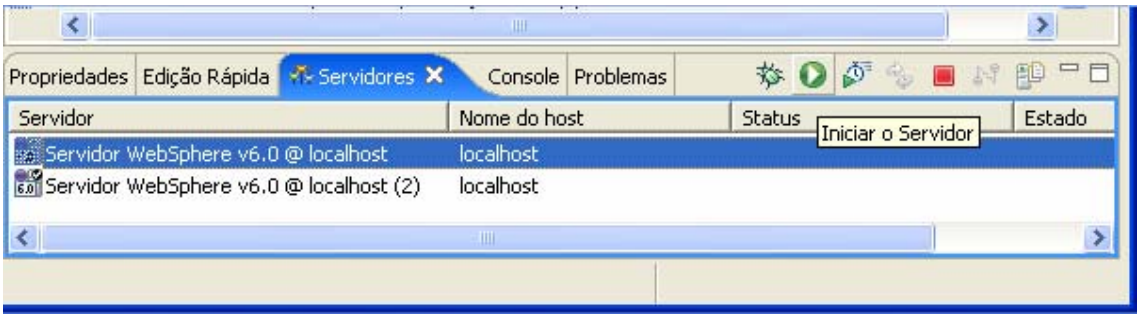


Figura 58 - Iniciando o Servidor Web Através do RAD - ETTK.

17.4 Adicionando as classes

A seguir criaremos as classes JAVA que serão usadas para os processos de criação do Web Service.

Serão duas classes : Calculadora.JAVA e CalcCliente.JAVA. A ferramenta RAD tem algo que iremos utilizar a seguir na sua IDE que é de extrema valia neste momento que é a mudança de perspectiva. Como até agora estávamos criando um projeto Web a perspectiva utilizada era Web, mas como agora iremos criar as classes citadas, faremos a mudança de perspectiva através do item no menu : Janela → Abrir perspectiva → JAVA. Agora temos a IDE do desenvolvedor conforme exemplo a seguir:

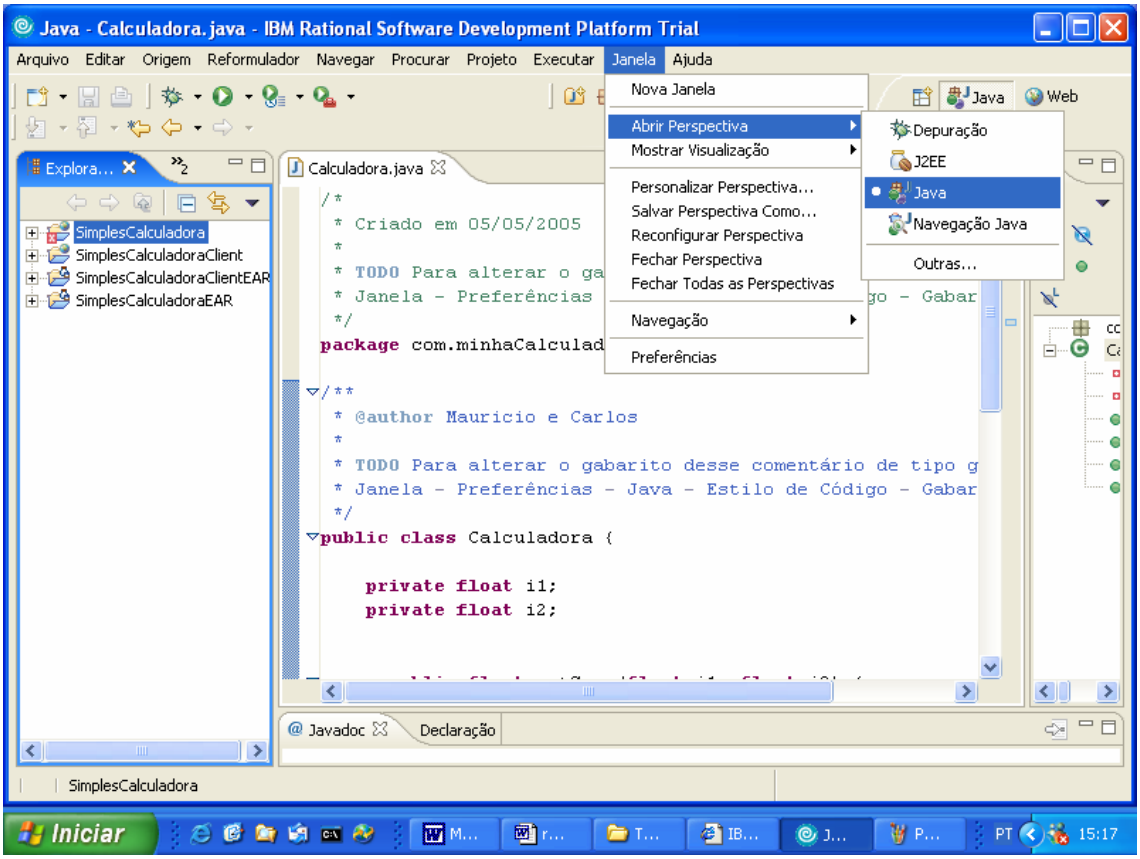


Figura 59 – Definição da Perspectiva das Classes da Aplicação Web - ETTK.

Com a perspectiva JAVA em atividade, agora iremos criar as duas classe clicando com o botão direito no projeto SimplesCalculadora → Novo → Classe. Aqui iremos criar as duas classes já citadas, onde devemos preencher para o pacote : minhaCalculadora e dar o nome da classe Calculadora, os demais campos, são default:

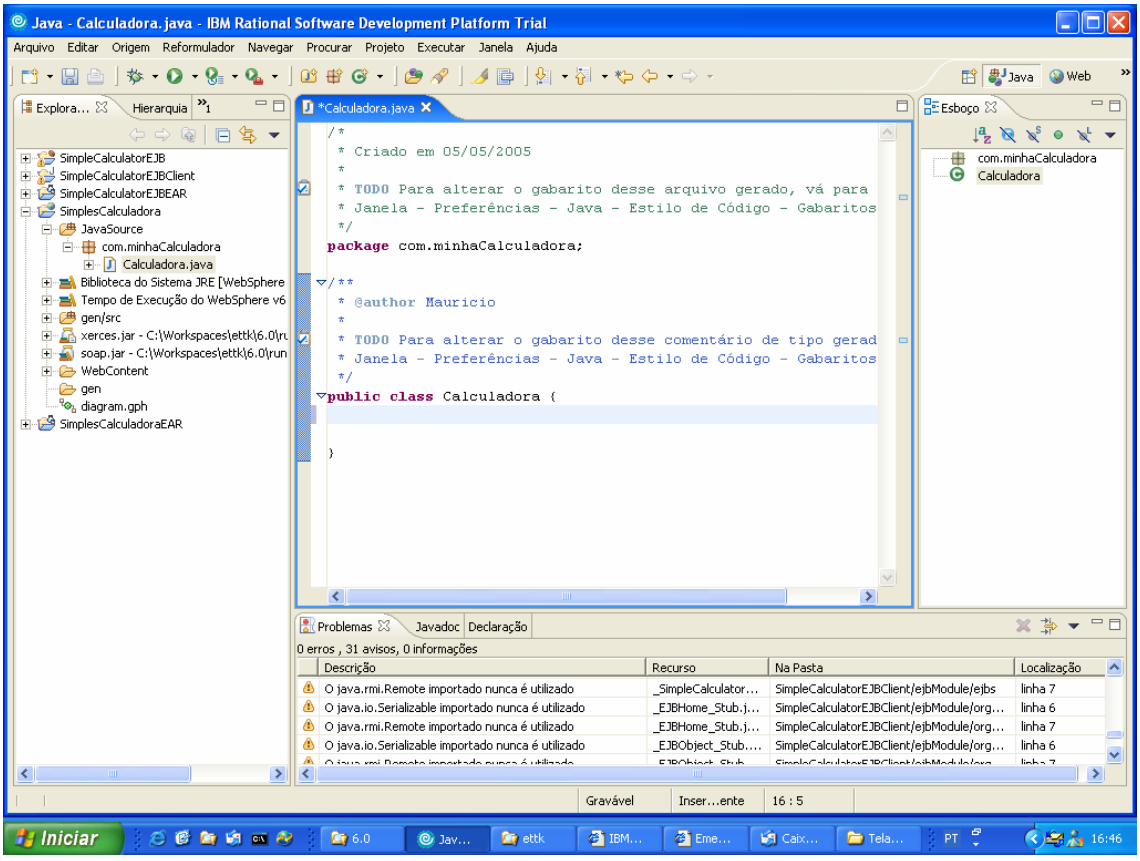


Figura 60 - Criação das Classes da Aplicação Web - ETTK.

A classe Calculadora contém o seguinte código :

Quadro 21 - Código da Classe Calculadora - ETTK.

```
package com.minhaCalculadora;

public class Calculadora (

        private float i1;

        private float i2;

public float getSoma(float i1, float i2) {

        return i1 + i2;

}

public float getSub (float i1, float i2) {

        return i1 – i2;

}

public float getDiv (float i1, float i2) {

        return i1 / i2;

}

public float getMult (float i1, float i2) {

        return i1*i2;

}
```

A classe CalcCliente contém o seguinte código :

Quadro 22 - Código da Classe CalcCliente - ETTK.

```
package com.minhaCalculadora/

public class CalcClient {

        float resultado;

        public float criarNovaCalculadora (float n1, float n2) {

        Calculadora calculadora = new Calculadora();
```

```
        calculadora.getSoma(2,2);
        calculadora.getSub(11,7);
        calculadora.getDiv(75,25);
        calculadora.getMult(12,12);
    }
}
```

17.5 Criando o Web Service

Agora que já possuímos uma aplicação, podemos criar o nosso Web Service exemplo. Devemos ir ao menu arquivo → novo → outros → Serviços da Web → Serviços da Web. Será aberta uma janela, onde deverá ser escolhido em tipo de serviço da Web, como Java Bean e os demais campos como segue o exemplo abaixo:

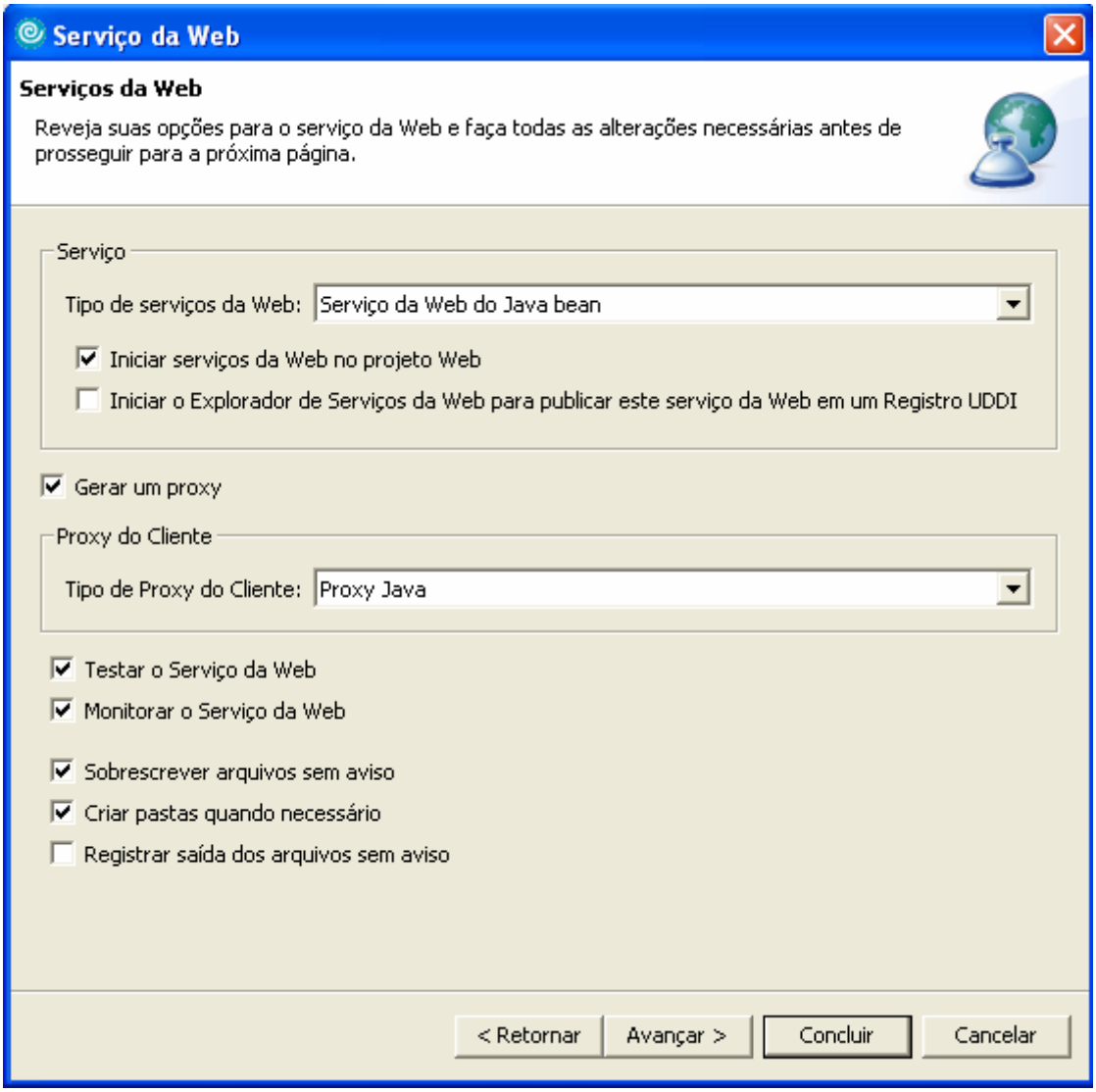


Figura 61 - Criação do Web Service - ETTK.

Na tela a seguir, deverá ser preenchido o nome da classe que será utilizada para criação do Web Service, como o package em questão, como mostra a seguir:

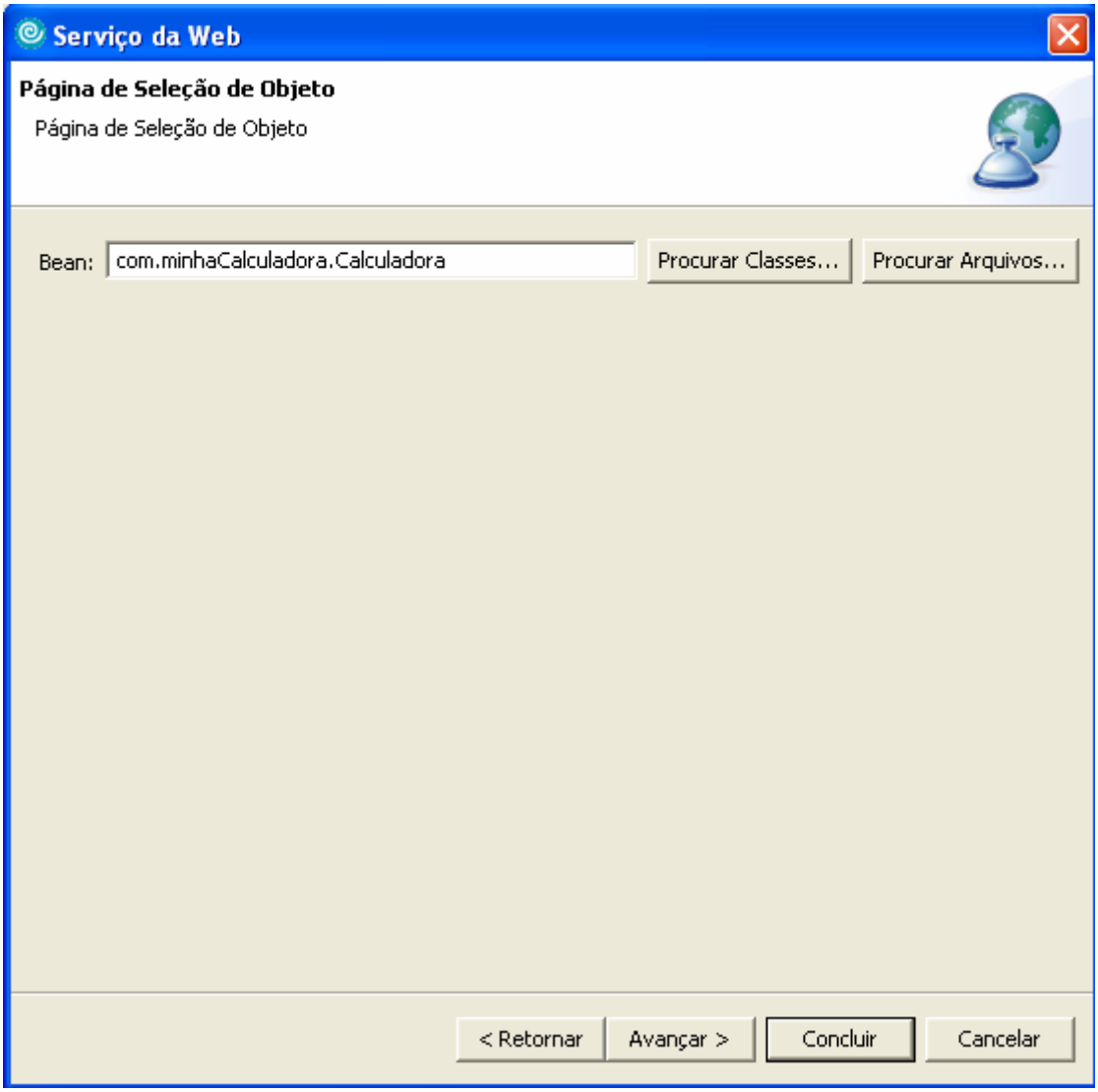


Figura 62 - Seleção dos Objeto do Web Service - ETTK.

Ao selecionar avançar, será apresentada uma janela com os projetos antes criados, onde o usuário deverá selecionar avançar para continuar a criação do Web Service.

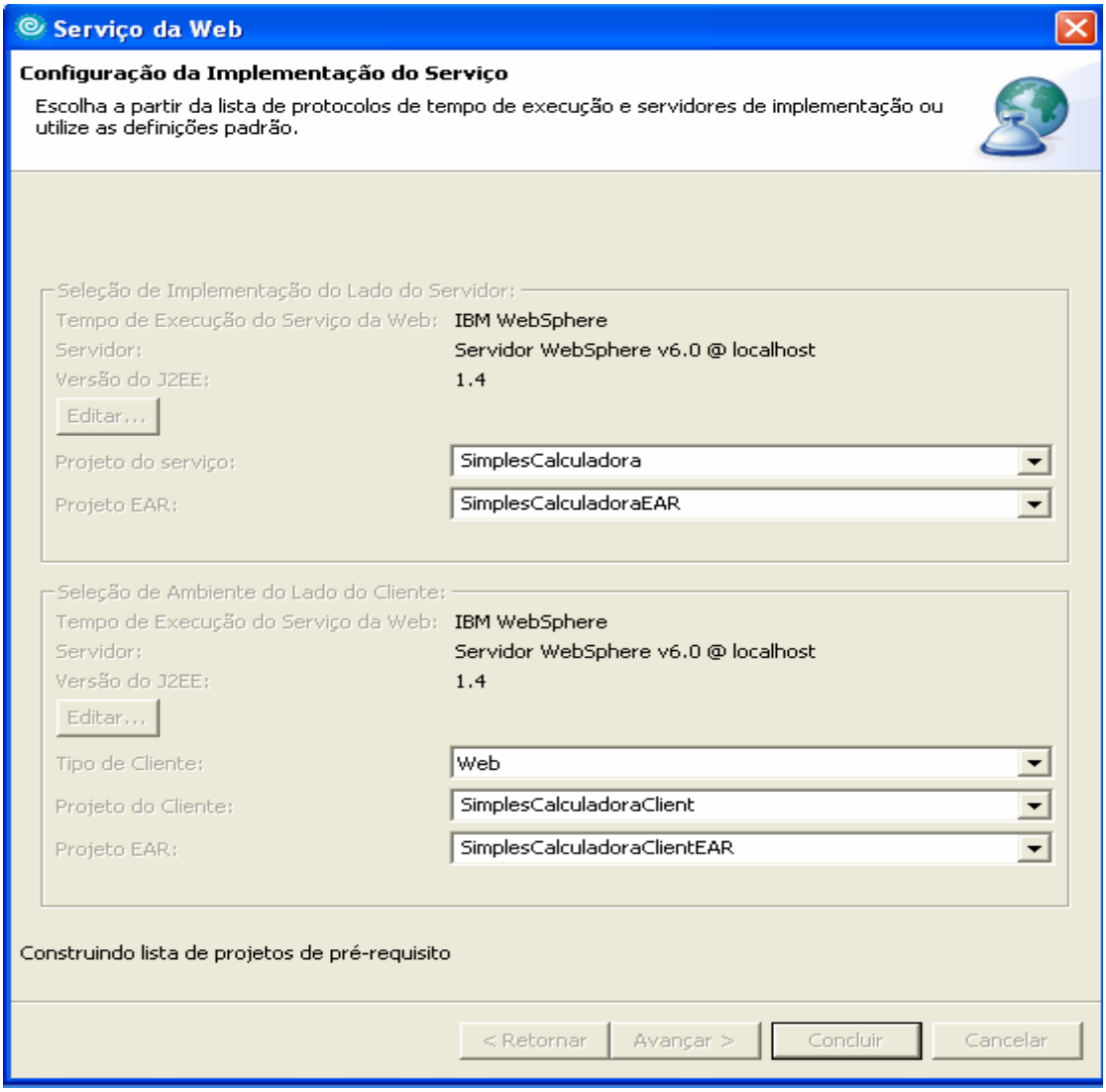


Figura 63 - Configuração da Implementação do Web Service - ETTK.

Para a próxima tela, devemos aceitar a classe que será criada pelo RAD para o nosso bean criado.

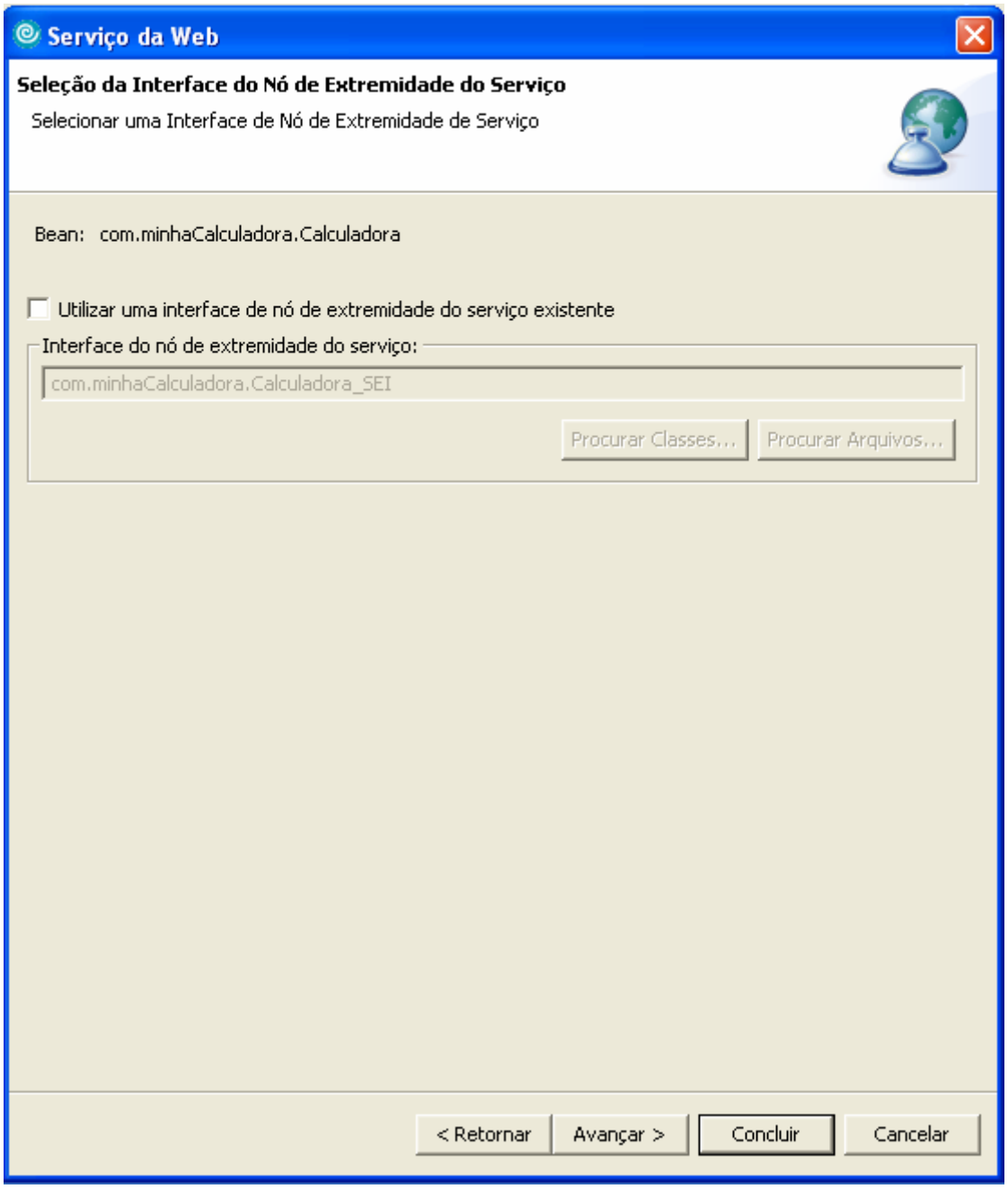


Figura 64 - Seleção da Interface do Nó de Extremidade do Web Service - ETTK.

A próxima etapa é a criação do arquivo wsdl para nossa classe Calculadora, onde os métodos da mesma são apresentados. Devemos colocar como nome do arquivo Calculadora.wsdl e avançar

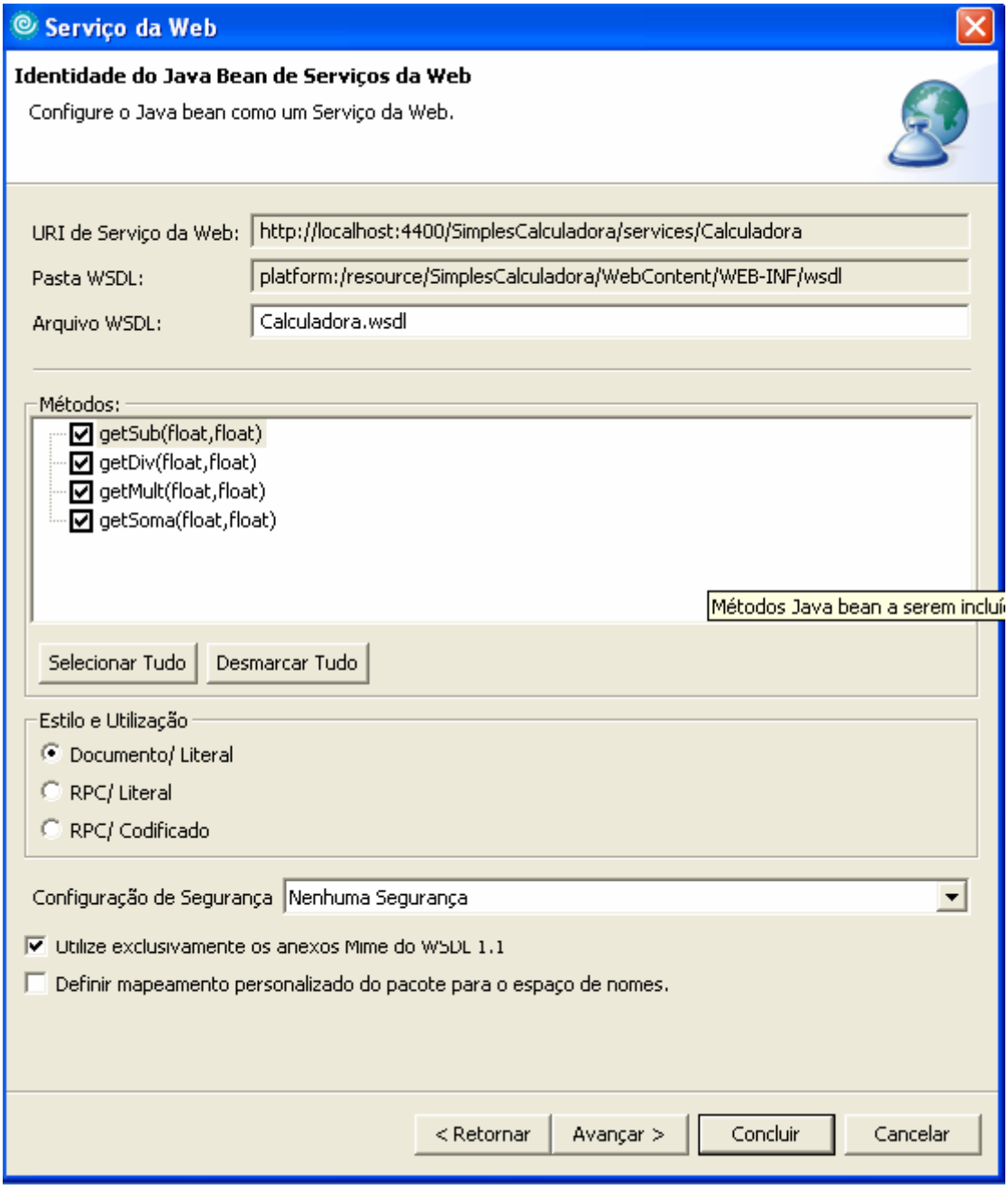


Figura 65 - Criação do Arquivo WSDL - ETTK.

Na próxima janela (não será mostrada) será escolhido o serviço padrão para teste, onde escolheremos serviços Web e avançamos.

A janela que segue, mostra a página para geração do Proxy, onde deveremos selecioná-lo e avançar.

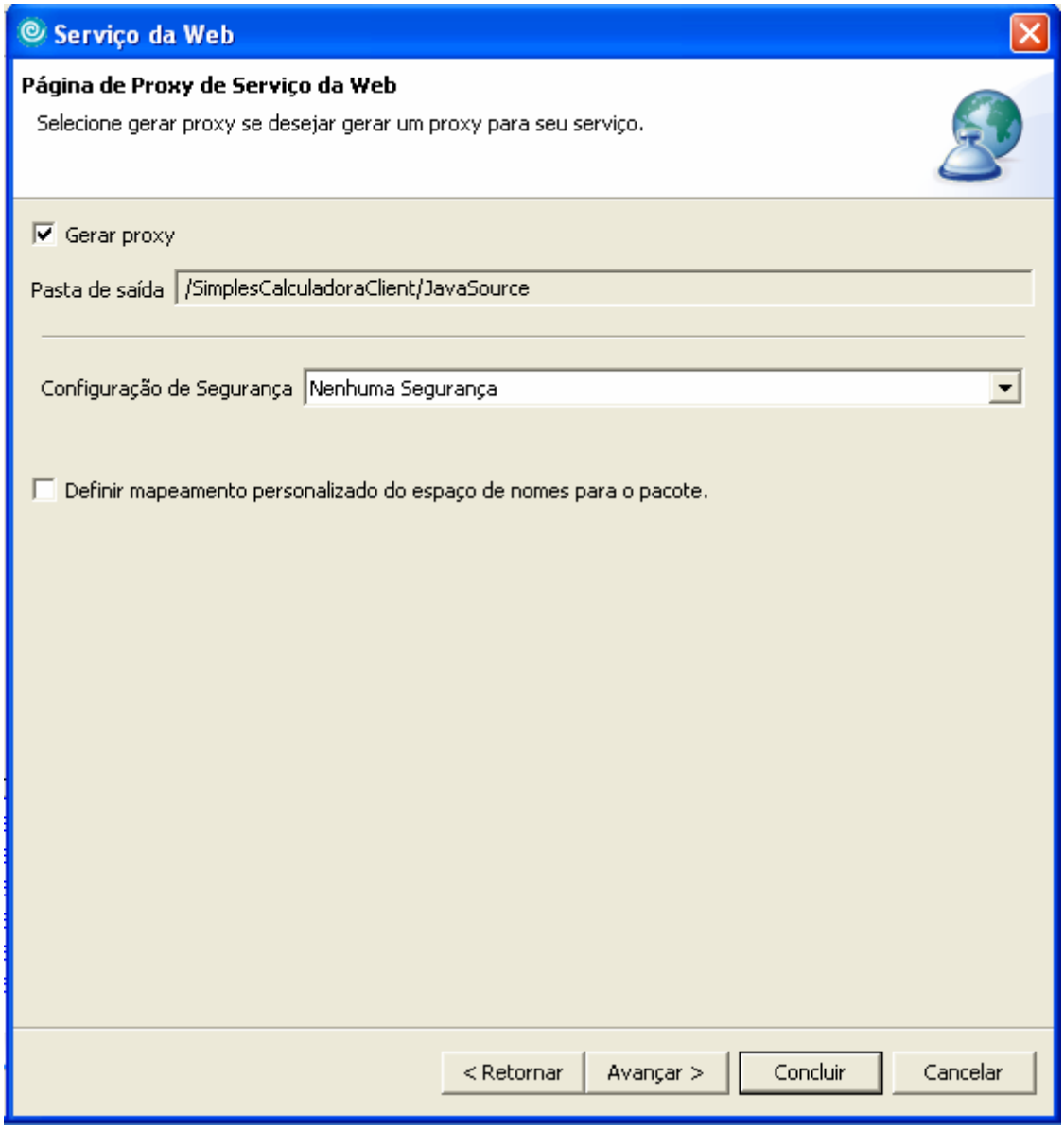


Figura 66 - Geração do Proxy do Web Service - ETTK.

A próxima tela exibe a opção para testarmos o Proxy gerado através de uma página Web que será criada. Devemos clicar no check para teste, bem como no check que executa os testes no servidor e avançar.

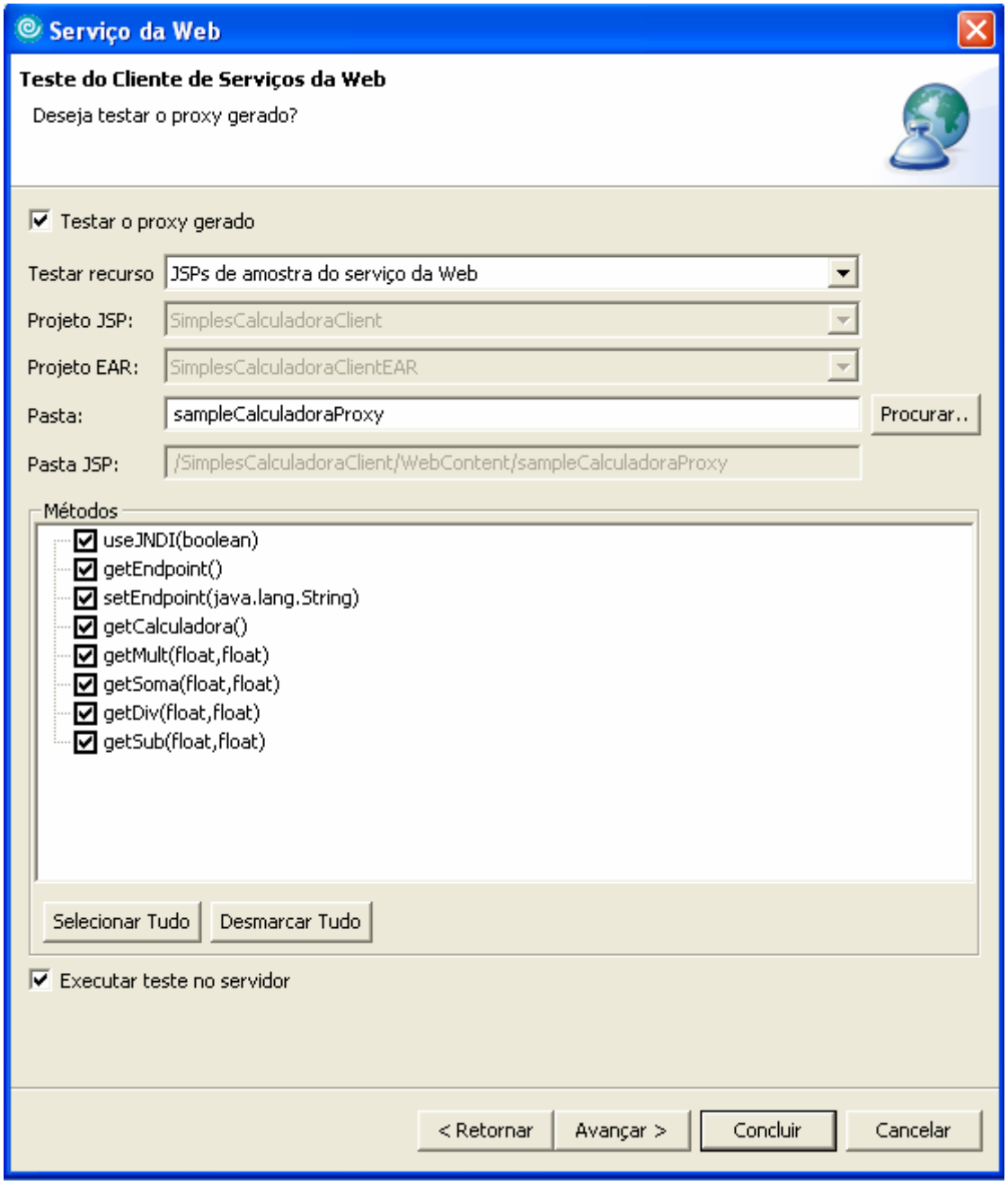


Figura 67 - Teste do Cliente do Web Service - ETTK.

A próxima tela nos pergunta se desejamos publicar o Web Service em um registro UDDI. Para nosso exemplo, não vamos publicar. Após isto podemos clicar avançar ou concluir, que o código será criado e o servidor iniciado. Isto poderá demorar alguns minutos. No final é exibido um navegador no painel central, onde

devemos escolher um dos métodos da calculadora para que tenhamos o resultado. Na janela que mostraremos abaixo, selecionamos o método getSoma().

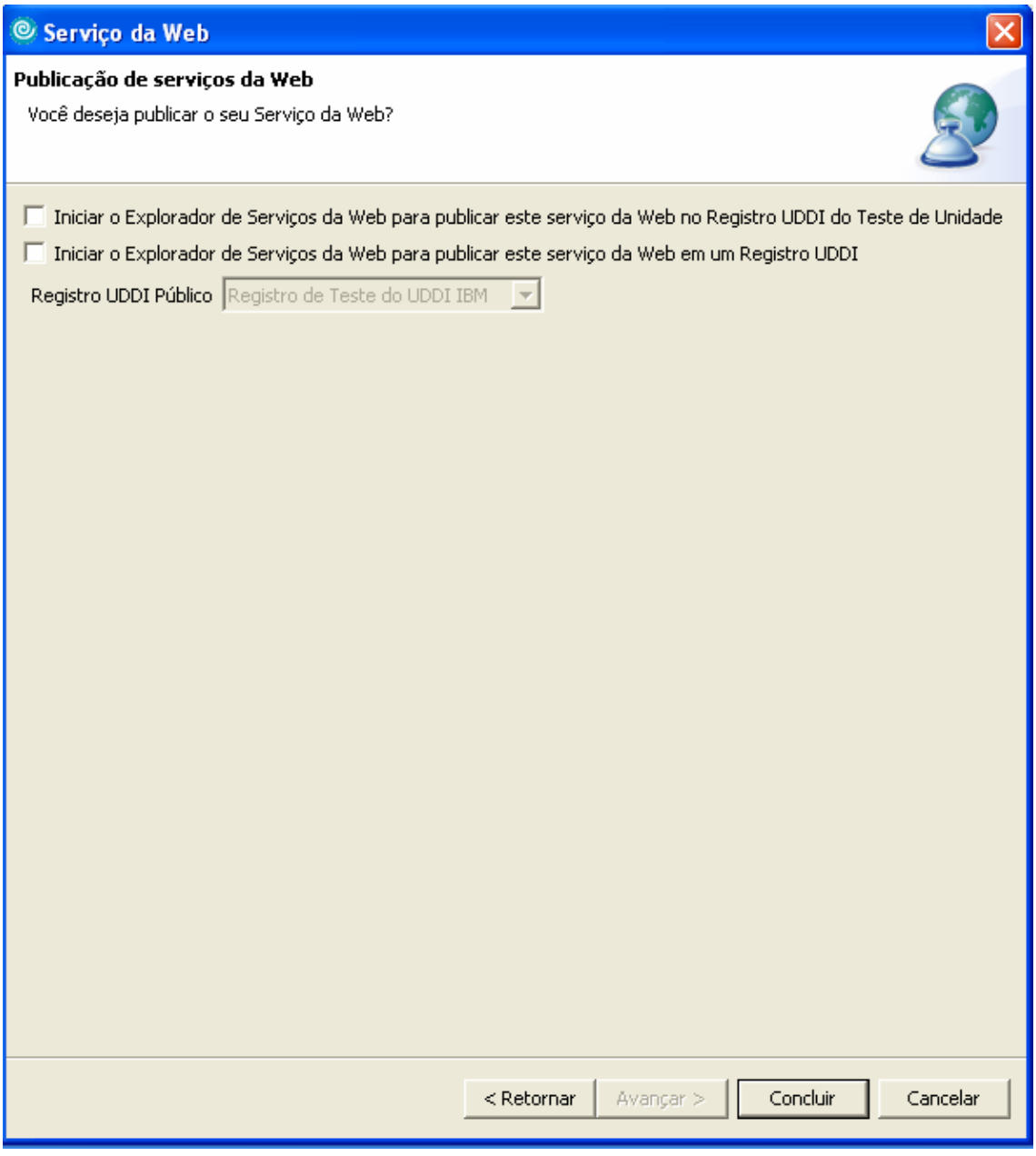


Figura 68 - Publicação do Web Service - ETTK.

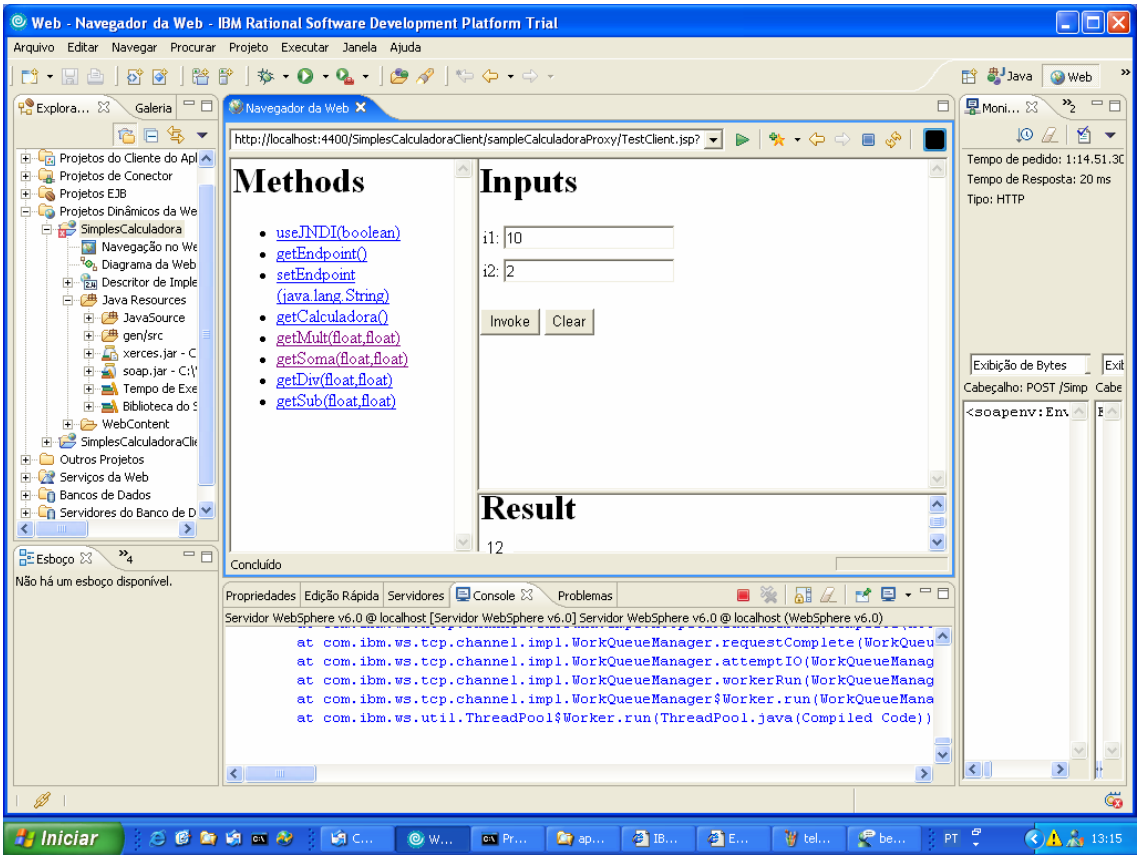


Figura 69 - Teste do Web Service - ETTK.

O arquivo Proxy foi gerado é mostrado no anexo 3.

Este arquivo foi chamado através da página JSP que foi gerada automaticamente pelo RAD. O arquivo WSDL gerado é mostrado no anexo 4.

18 CONCLUSÕES E TRABALHOS FUTUROS.

18.1 Conclusões.

As plataformas avaliadas foram selecionadas com o objetivo de fornecerem uma visão abrangente do processo de desenvolvimento de web services.

Na proposta inicial definimos os seguintes parâmetros de avaliação:

- Desempenho;
- Desenvolvimento de Aplicações Específicas;
- Utilização;
- Instalação;
- Ambiente de Desenvolvimento.

Durante o transcorrer do trabalho os parâmetros foram reavaliados e observamos a necessidade da redefinição dos mesmos. Assim decidimos por não avaliar a performance e a segurança das aplicações desenvolvidas, uma vez que seria necessária a criação de cenários específicos e idênticos e envolveriam aspectos que não são foco deste trabalho, como por exemplo, as configurações de hardware, sistemas operacionais, estrutura de rede, etc.

No caso da avaliação da segurança no desenvolvimento de web services, as plataformas avaliadas não fornecem uma documentação bem definida a respeito, sendo que todas fazem, de alguma forma, referência aos padrões os quais citamos no capítulo 9 de nosso trabalho. Assim sendo, optamos por não efetuar a avaliação deste parâmetro, até mesmo porque, tal esforço demandaria tempo e pesquisa que iriam extrapolar os prazos definidos para a conclusão do presente trabalho.

Assim sendo, optamos pelos seguintes parâmetros de avaliação:

- Instalação;

- Documentação;
- Dificuldade de Aprendizado;
- Velocidade de Desenvolvimento;
- Custo.

A instalação das plataformas não chega a ser um fator determinante para a escolha de um dos produtos. Apesar da instalação de algumas apresentarem-se mais complexas, não necessitam de muita experiência.

A plataforma AXIS não possui um arquivo instalador, sendo necessário conhecimento sobre a configuração e estrutura de diretórios do servidor web Tomcat. As plataformas JWSDP, GLUE e ETTK possuem um arquivo instalador, sendo que o ETTK permite também uma instalação sem uma interface gráfica, contudo, dificultando seu uso e a configuração do servidor web websphere. É necessário salientar que o recurso GUI do ETTK necessita da ferramenta RAD, cujos arquivos de instalação possuem 2,82 Gigabytes. Com o uso de uma conexão banda larga ADSL de 300 KBps, foi necessário em torno de 10 horas para completar o download dos arquivos.

Segundo nossa avaliação, consideramos as plataformas AXIS, GLUE e JWSDP como de baixa complexidade para instalação e a plataforma ETTK com média complexidade.

Outro aspecto avaliado foi a documentação. Este é um parâmetro vital para qualquer plataforma de desenvolvimento. Observamos que a documentação das plataformas ainda deixa a desejar. Entretanto um fator favorável está em que todas elas se baseiam e utilizam o ambiente JAVA, que é muito bem documentado.

Encontramos vasto material de referência, tanto no site da SUN Microsystems (http://www.sun.com), na vasta bibliografia existente assim como na internet.

Neste aspecto sobressaiu a plataforma JWSDP, com documentação bem estruturada e de fácil navegação.

A plataforma AXIS, apesar da grande quantidade de documentação, já não apresenta a mesma qualidade em sua estrutura e navegação, o que pode ser decorrente do fato de ser uma plataforma aberta, com grande quantidade de colaboradores e desenvolvedores, o que dificulta a manutenção e atualização da documentação (POTTS&KOPACK, 2003).

Uma característica observada em todas as plataformas foi a pouca quantidade de livros publicados sobre cada uma delas, sendo a maior parte da documentação encontrada nos sites dos fornecedores.

Também avaliamos a Dificuldade de aprendizado em cada uma das plataformas e observamos que todas necessitam de uma experiência razoável em desenvolvimento. Com algumas delas é possível colocar um web service simples em funcionamento sem muito esforço. Em outras, o desenvolvimento requer um aprendizado maior. As plataformas comerciais (ETTK e GLUE) oferecem um maior suporte à interface gráfica. Contudo com o lançamento da IDE Netbeans 4.1RC2 (http://www.nerbeans.org), com o JWSDP incorporado, tornou esta uma opção interessante entre as plataformas gratuitas.

Em relação à velocidade de desenvolvimento, podemos defini-la como sendo o tempo que um desenvolvedor necessita para terminar uma determinada tarefa, após possuir o domínio de uma ferramenta.

Quanto maior a automação da ferramenta, maior será sua velocidade de desenvolvimento. Contudo observamos que seu custo aumenta na mesma proporção (POTTS&KOPAC, 2003). Também observamos que a migração entre plataformas gratuitas tem um impacto menor do que em plataformas comerciais.

O custo é um fator relevante em qualquer organização. No entanto não pode ser considerado isoladamente quando da aquisição de uma plataforma de desenvolvimento. Outros aspectos necessitam ser considerados, sendo muitas vezes mais importantes. Podemos citar o treinamento da equipe de desenvolvimento, suporte, estrutura de hardware e manutenção a longo prazo dos sistemas desenvolvidos (POTTS&KOPACK, 2003). Como não foi possível medir os outros fatores citados, nossa avaliação se restringiu ao custo de aquisição das plataformas.

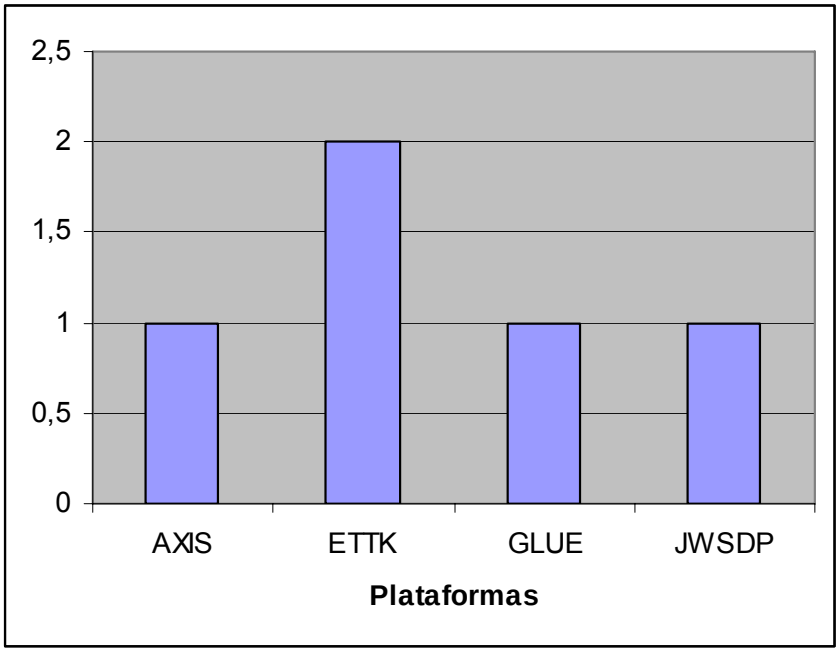
Na Tabela 2, mostramos uma comparação entre as plataformas, considerando-se os parâmetros avaliados.

Tabela 2 - Comparação Entre as Plataformas de Desenvolvimento

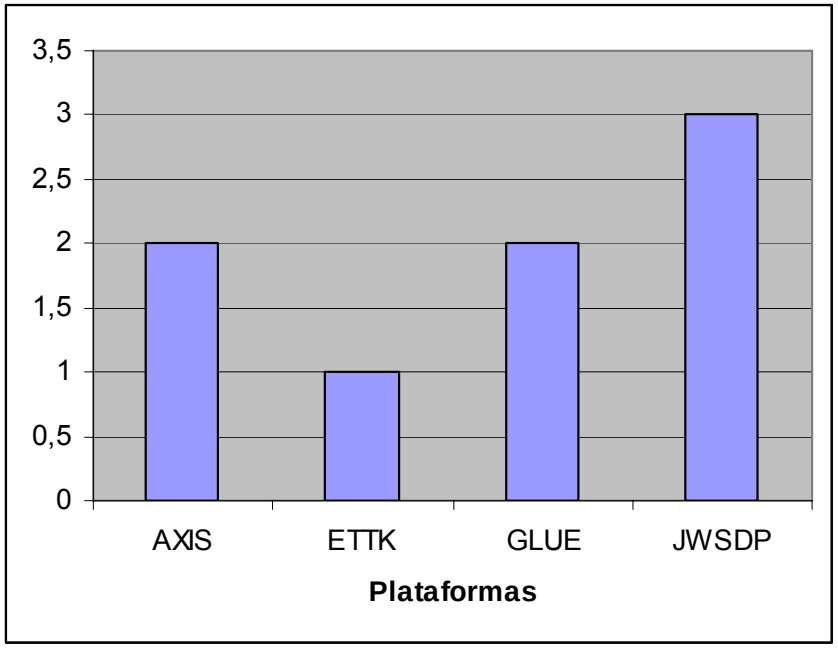
Plataforma	Complexidade da Instalação	Qualidade da Documentação	Dificuldade de Aprendizado	Velocidade de Desenvolvimento	Custo
AXIS	(1) Baixa	(2) Regular	(2) Média Para Programadores Java	(2) Média Para Programadores Java	Gratuito
ETTK	(2) Média	(1) Ruim	(1) Fácil	(3) Rápida Para Programadores Java	$3.000,00
GLUE	(1) Baixa	(2) Regular	(1) Fácil Para Programadores Java	(2) Média Para Programadores Java	Gratuito
JWSDP	(1) Baixa	(3) Boa	(2) Média Para Programadores Java	(3) Rápida Para Programadores Java	Gratuito

Considerando os valores das avaliações das plataformas, criamos gráficos comparativos dos parâmetros avaliados, como mostrados nos quadros a seguir:

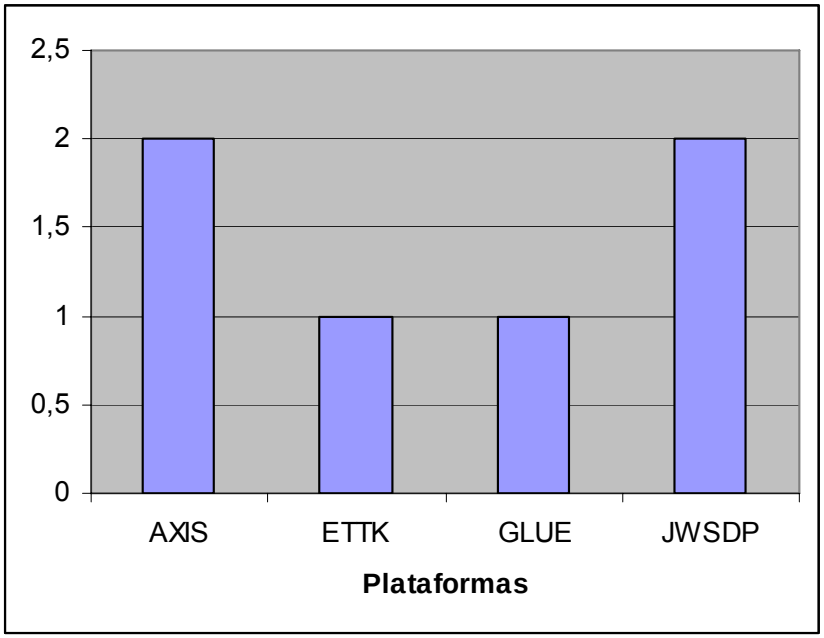
Quadro 23 - Complexidade da Instalação.



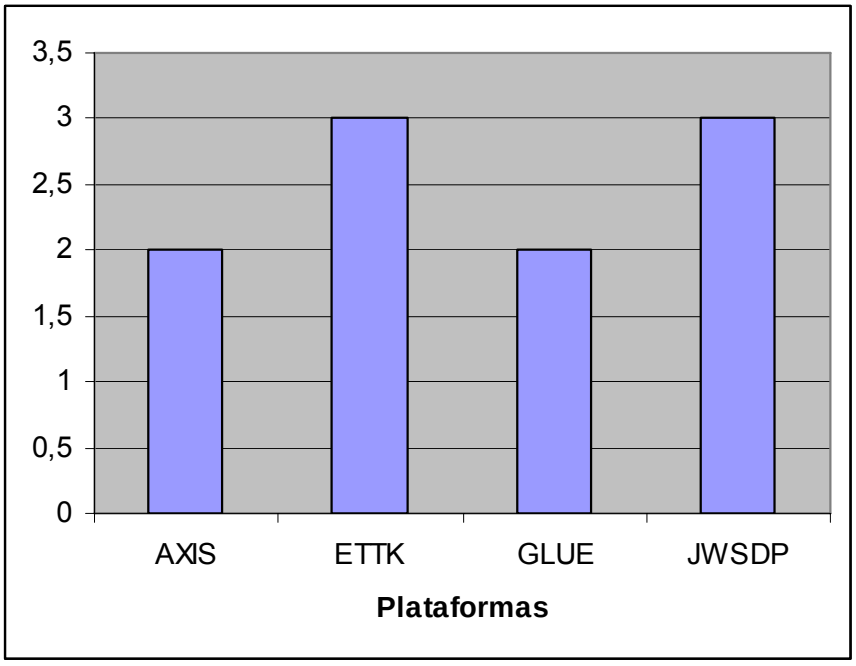
Quadro 24 - Qualidade da Documentação.



Quadro 25 - Dificuldade de Aprendizado.



Quadro 26 - Velocidade de Desenvolvimento.



Pelo exposto neste trabalho, os autores consideram ter atingido de forma satisfatória os objetivos propostos neste trabalho. Durante o transcorrer do mesmo, alguns ajustes nos parâmetros de avaliação se tornaram necessários, até porque algumas plataformas tornaram disponíveis versões e recursos não existentes no início de nossa pesquisa.

Concluímos, portanto, considerando-se o desenvolvimento baseado no ambiente JAVA e o conjunto de parâmetros usados em nossa avaliação, que a melhor opção para o desenvolvimento de web services encontra-se na adoção da plataforma JWSDP.

18.2 Trabalhos Futuros.

O uso de web services vem aumentando consideravelmente pelas empresas que desejam tornar suas informações disponíveis na internet. É um campo de pesquisa contínuo e este trabalho poderá ser estendido para a avaliação de outras plataformas, assim como a análise de outras características não abordadas nesta pesquisa.

19 BIBLIOGRAFIA.

ALPHAWORKS. **Emerging Technologies Toolkit.** Disponível em http://www.alphaworks.ibm.com/tech/ettk

ALVES, Maria Bernadete Martins; ARRUDA, Suzana Margareth. **COMO FAZER REFERÊNCIAS: bibliográficas, eletrônicas e demais formas de documentos.** Disponível em: http://bu.ufsc.br/framerefer.htm

ARMSTRONG, Eric; et. al. – **The Java Web Services Tutorial**.July 25, 2003.

AXIS**. Axis Documentation.** Disponível em http://ws.apache.org/axis/java/index.html

_____.**Axis Architecture Guide.** Disponível em http://ws.apache.org/axis/java/architecture-guide.html

CERAMI, Ethan – **Web Services Essentials. Distribuited Applications with XML-RPC, UDDI & WSDL.** O´Reilly. Janeiro 2003.

FOSTER, Jay; et. al.- **Developing Web Services With Java APIs form XML Using WSDP.** Syngres. 2002. Disponível em http://www.syngres.com
CHAPPELL, David; JEWEL, Tyler. **Java Web Services.** O´Reilly. Março 2002.

GROS, Christian – **Web Service Programming Using Axis.** O´Reilly, Open Source Convention. Julho 7 – 11, 2003.

POTTS, Stephen; KOPACK, Mike. **Aprenda em 24 horas Web Services.** Tradução de Marcos Vieira. Campus. Rio de Janeiro. 2003.

SILVA, Edna Lúcia da; MENEZES, Estera Muszkat. **Metodologia da Pesquisa e Elaboração de Dissertação.** UFSC/PGED/LED. 3 ed. Florianopolis. 2001.

SUN. **The Java Web Service Tutorial 1.0**. Disponível em http://java.sun.com/webservices/docs/1.0/tutorial/

TECHMETRIX RESEARCH. **AXIS: The New Incarnation of Apache SOAP – A Technical White Paper.** Disponível em http://www.techmetrix.com

TOPLEY, Kim – **Java Web Services in Nutshell.** O´Reilly, Junho 2003.

XML FROM THE INSIDE OUT. **XML development, XML resources, XML specifications.** Disponível em http://www.xml.com/index.csp

WEBMETHODS. **Glue 5.0.2 User Guide.** Disponível em http://www1.webmethods.com/docs/glue/guide/index.html

W3C**. World Wide Web Consortium.** Disponível em http://www.w3.org

ANEXO 1 – Configuração das Variáveis de Ambiente da Plataforma AXIS.

Nos S.O. WindowsNT/2000/XP, a criação das variáveis de ambiente pode ser feita da seguinte forma : “Meu Computador”→”Propriedades”→”Avançado” →”Variáveis de Ambiente”

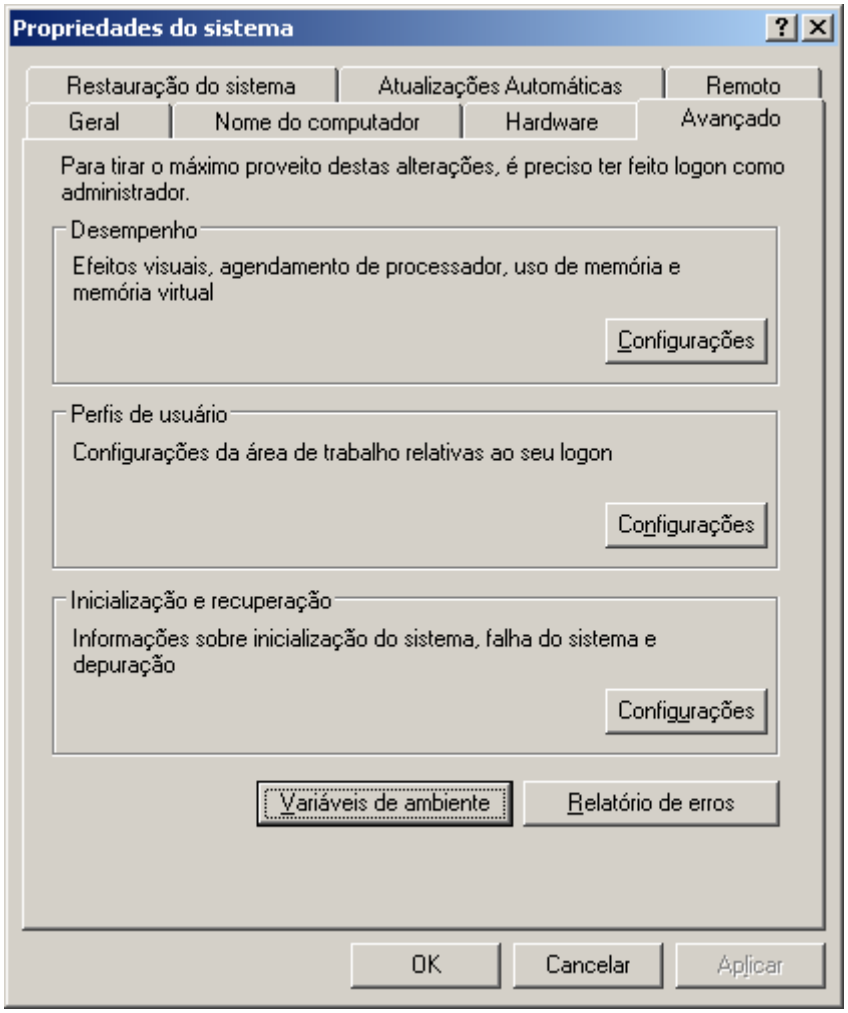


Figura 70 – Propriedades do Sistema.

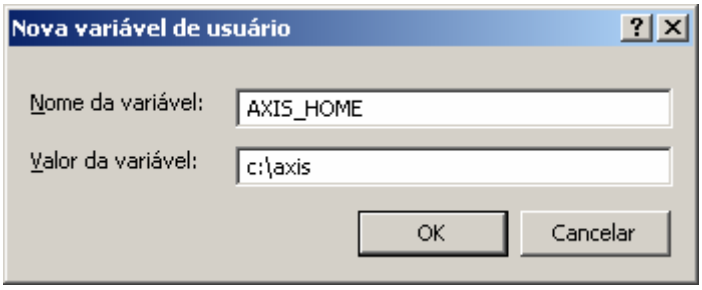


Figura 71 – AXIS_HOME.

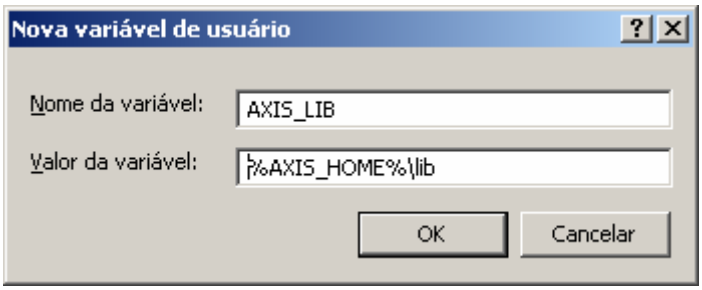


Figura 72 – AXIS_LIB.

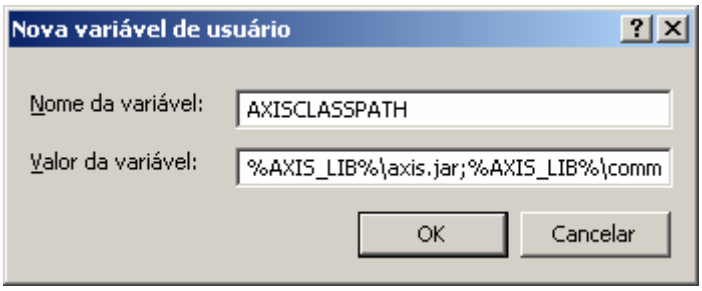


Figura 73 – AXISCLASSPATH.

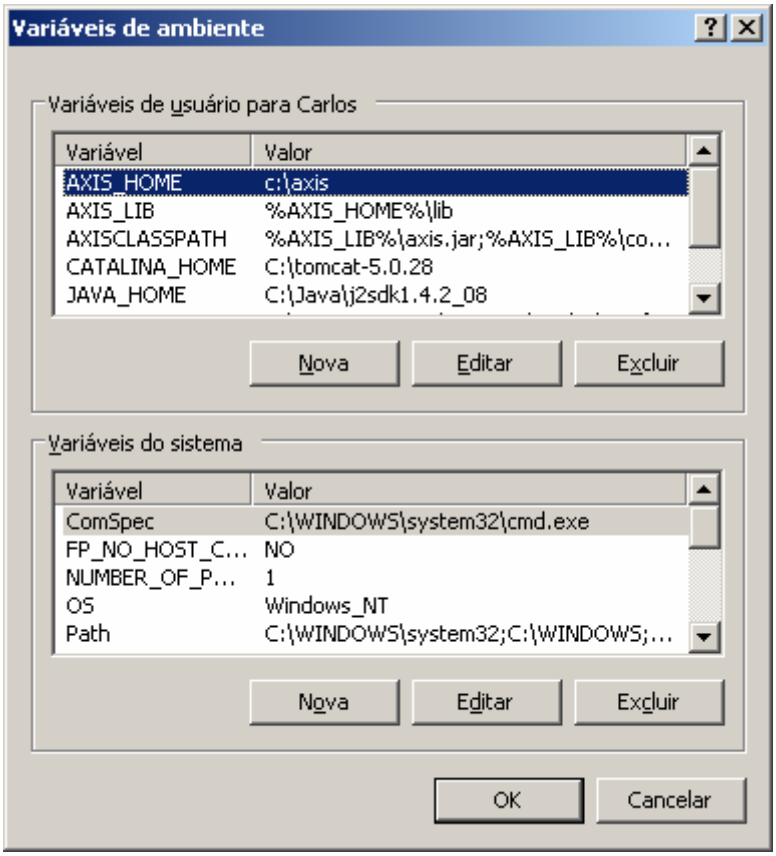


Figura 74 – Variáveis de Ambiente.

Outro método que pode ser utilizado é a criação de um arquivo executável que crie as mesmas variáveis de ambiente, como por exemplo:

Quadro 27 - Código de Inicialização das Variáveis de Ambiente AXIS.

```
set AXIS_HOME=c:\axis
set AXIS_LIB=%AXIS_HOME%\lib
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\activation.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\ant.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\axis-ant.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\axis.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\commons-discovery.jar
```

```
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\commons-logging.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\jaxrpc.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\jsse.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\mail.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\saaj.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\uddi4j.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\wsdl4j.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\xalam.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\log4j-1.2.8.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\xmlsec-1.2.1.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\xercesSamples.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\xml-apis.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\xmlParserAPIs.jar
set AXISCLASSPATH=%AXISCLASSPATH%;%AXIS_LIB%\xercesImple.jar
set CLASSPATH=%AXISCLASSPATH%
```

ANEXO 2 – Configuração das Variáveis de Ambiente da Plataforma GLUE.

Nos S.O. WindowsNT/2000/XP a criação destas variáveis pode ser feita da seguinte forma : “Meu Computador”→”Propriedades”→”Avançado” →”Variáveis de Ambiente”

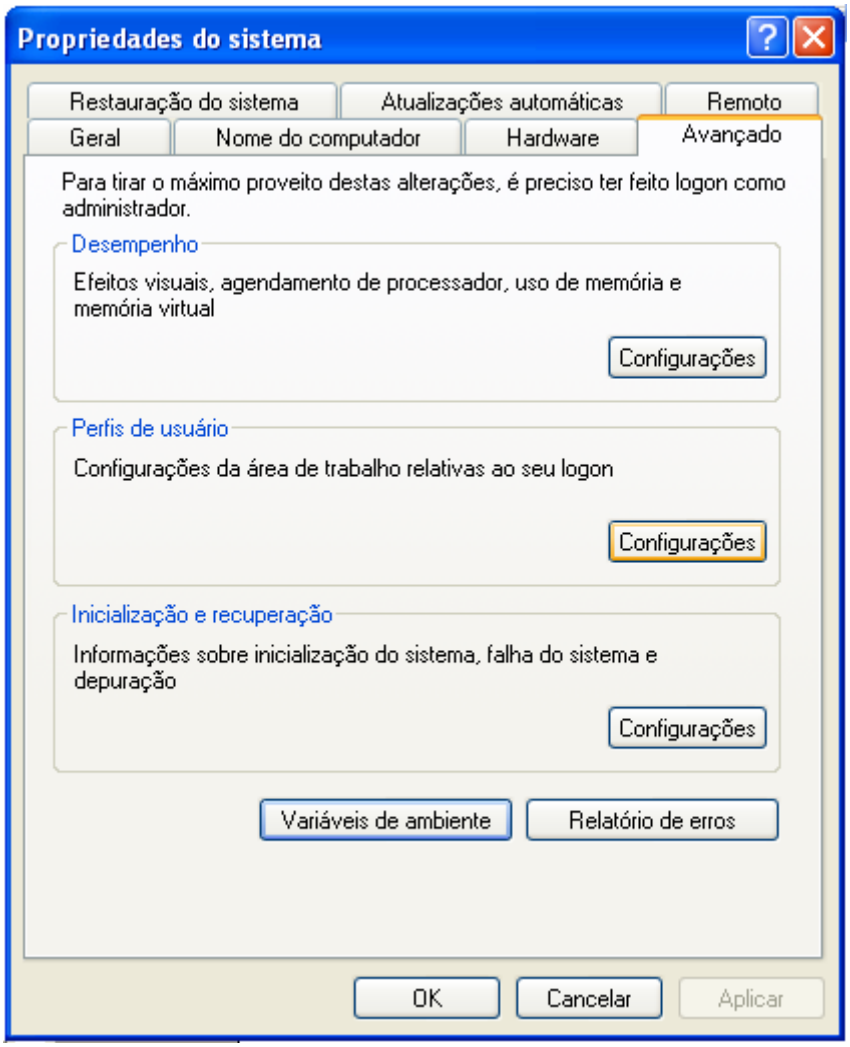


Figura 75

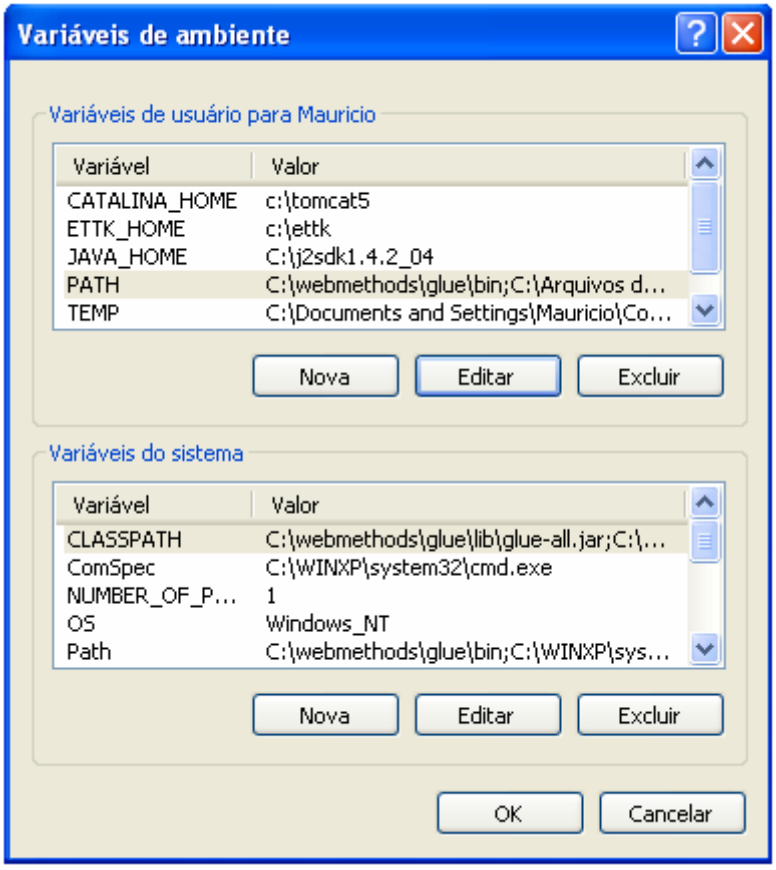


Figura 76

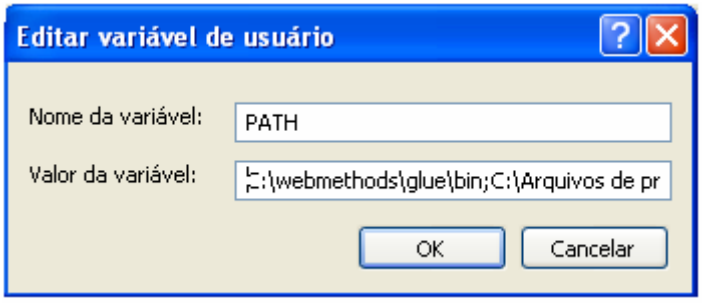


Figura 77

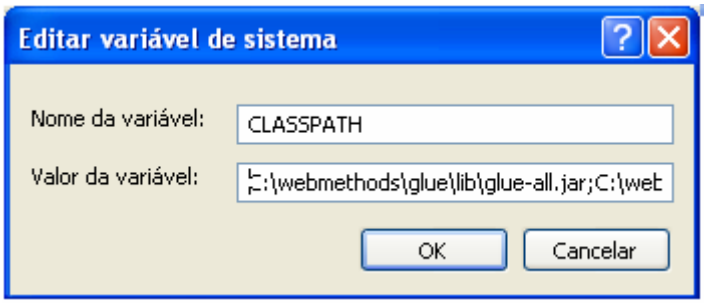


Figura 78

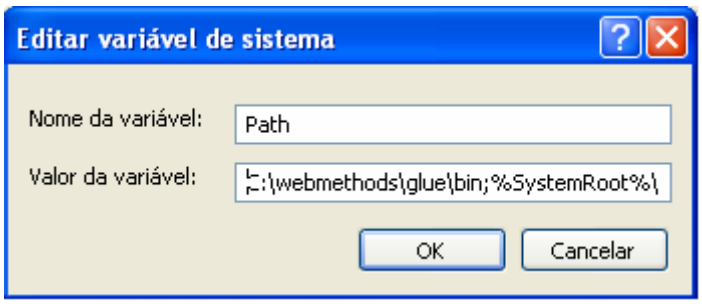


Figura 79

ANEXO 3 – Proxy ETTK.

Quadro 28 - Código do Proxy - ETTK.

```
package com.minhaCalculadora;

public class CalculadoraProxy implements com.minhaCalculadora.Calculadora{

private boolean_useJNDI = true;

private String-endpoint = null;

private com.minhaCalculadora.Calculadora calculadora = null;

public CalculadoraProxy () {

  _initCalculadoraProxy();

}

private void _initCalculadoraProxy() {

 if (_useJNDI) {

  try {

     javax.naming.InitialContext ctx = new javax.naming.InitialContext();

     calculadora =

((com.minhaCalculadora.CalculadoraService)ctx.lookup("Java:comp/env/service/Calc

uladoraService")).getCalculadora();

} catch (javax.naming.NamingException namingException) {}

   Catch (javax.xml.rpc.ServiceException service Exception) {}

}

If (calculadora == null) {

 try{

  calculadora = (new

com.minhaCalculadora.CalculadoraServiceLocator()).getCalculadora();

} catch (javax.xml.rpc.ServiceException serviceExpeption) {}
```

```
}
  if (calculadora != null) {
   if (_endpoint != null) {
((javax.xml.rpc.Stub)calculadora)._setProperty("javax.xml.rpc.service.endpoint.addre
ss", _endpoint);
else _endpoint
(String)((javax.xml.rpc.Stub)calculadora)._getProperty(“javax.xml.rpc.service.endpoin
t.address”);
}}
public void useJNDI(boolean useJNDI) {
  _useJNDI = useJNDI;
  calculadora = null;
}
public String getEndpoint() {
 return _endpoint;
}
public void setEndpoint(String endpoint) {
  _endpoint = endpoint;
 if (calculadora != null)
((javax.xml.rpc.Stub)calculadora)._setProperty(“javax.xml.rpc.service.endpoint.addre
ss”,_endpoint);
}
public com.minhaCalculadora.Calculadora getCalculadora() {
  if (calculadora == null)
   _initCalculadoraProxy();
```

```
return calculadora;
}
public float getMult(float i1, float i2) throws java.rmi.RemoteException {
  if (calculadora == null)
    _initCalculadoraProxy();
   return calculadora.getMult(i1,i2);
}
public float getSoma (float i1, float i2) throws java.rmi.RemoteException {
  if (calculadora == null)
   _initCalculadoraProxy();
   return calculadora.getSoma(i1,i2);
}
public float getDiv(float i1, float i2) throws java.rmi.RemoteException {
 if (calculadora == null)
  _initCalculadoraProxy();
 return calculadora.getDiv(i1, i2);}
}
public float getSub(float i1, float i2) throws java.rmi.Remote.Exception {
  if (calculadora == null)
   _initCalculadoraProxy();
  return calculadora.getSub(i1,i2)
} }
```

ANEXO 4 - Arquivo WSDL – ETTK.

Quadro 29 - Código do Arquivo WSDL - ETTK.

```
<wsdl:definitions targetNamespace="http://minhaCalculadora.com"
xmlns:impl="http://minhaCalculadora.com"
xmlns:intf="http://minhaCalculadora.com"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsi="http://ws-i.org/profiles/basic/1.1/xsd"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <wsdl:types>
  <schema targetNamespace="http://minhaCalculadora.com"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:impl="http://minhaCalculadora.com"
xmlns:intf="http://minhaCalculadora.com"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <element name="getMultResponse">
    <complexType>
     <sequence>
      <element name="getMultReturn" type="xsd:float"/>
     </sequence>
    </complexType>
   </element>
   <element name="getSoma">
    <complexType>
     <sequence>
      <element name="i1" type="xsd:float"/>
      <element name="i2" type="xsd:float"/>
     </sequence>
    </complexType>
   </element>
   <element name="getSomaResponse">
    <complexType>
     <sequence>
      <element name="getSomaReturn" type="xsd:float"/>
     </sequence>
    </complexType>
   </element>
   <element name="getDiv">
    <complexType>
     <sequence>
      <element name="i1" type="xsd:float"/>
      <element name="i2" type="xsd:float"/>
     </sequence>
    </complexType>
   </element>
   <element name="getDivResponse">
```

```
 <complexType>
  <sequence>
   <element name="getDivReturn" type="xsd:float"/>
  </sequence>
 </complexType>
</element>
<element name="getSub">
 <complexType>
  <sequence>
   <element name="i1" type="xsd:float"/>
   <element name="i2" type="xsd:float"/>
  </sequence>
 </complexType>
</element>
<element name="getSubResponse">
 <complexType>
  <sequence>
   <element name="getSubReturn" type="xsd:float"/>
  </sequence>
 </complexType>
</element>
<element name="getMult">
 <complexType>
  <sequence>
   <element name="i1" type="xsd:float"/>
   <element name="i2" type="xsd:float"/>
  </sequence>
 </complexType>
</element>
</schema>
</wsdl:types>
 <wsdl:message name="getMultResponse">
    <wsdl:part element="impl:getMultResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="getMultRequest">
    <wsdl:part element="impl:getMult" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="getDivResponse">
    <wsdl:part element="impl:getDivResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="getDivRequest">
    <wsdl:part element="impl:getDiv" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="getSomaResponse">
    <wsdl:part element="impl:getSomaResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="getSubResponse">
    <wsdl:part element="impl:getSubResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="getSubRequest">
```

```
    <wsdl:part element="impl:getSub" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="getSomaRequest">
    <wsdl:part element="impl:getSoma" name="parameters"/>
  </wsdl:message>
  <wsdl:portType name="Calculadora">
    <wsdl:operation name="getMult">
      <wsdl:input message="impl:getMultRequest" name="getMultRequest"/>
      <wsdl:output message="impl:getMultResponse"
name="getMultResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getSoma">
      <wsdl:input message="impl:getSomaRequest"
name="getSomaRequest"/>
      <wsdl:output message="impl:getSomaResponse"
name="getSomaResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getDiv">
      <wsdl:input message="impl:getDivRequest" name="getDivRequest"/>
      <wsdl:output message="impl:getDivResponse"
name="getDivResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getSub">
      <wsdl:input message="impl:getSubRequest" name="getSubRequest"/>
      <wsdl:output message="impl:getSubResponse"
name="getSubResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="CalculadoraSoapBinding" type="impl:Calculadora">
    <wsdlsoap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getMult">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getMultRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getMultResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getSoma">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getSomaRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSomaResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getDiv">
```

```
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getDivRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getDivResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getSub">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getSubRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getSubResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="CalculadoraService">
      <wsdl:port binding="impl:CalculadoraSoapBinding" name="Calculadora">
        <wsdlsoap:address
location="http://localhost:4400/SimplesCalculadora/services/Calculadora"/>
      </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

ANEXO 5 - ARTIGO.

AVALIAÇÃO DE PLATAFORMAS PARA O DESENVOLVIMENTO DE WEB SERVICES

1. Introdução

A maioria das aplicações distribuídas e dos sistemas distribuídos é baseada no paradigma cliente/servidor, aonde o servidor anuncia um conjunto de serviços ao qual ele provê acesso para alguns recursos. O código que executa estes serviços é hospedado localmente no servidor que executa cada serviço e pode utilizar a capacidade de seu processador, ou seja, o servidor possui os recursos o know-how e o processador.

Se o cliente quiser usar algum recurso oferecido pelo servidor o cliente escolhe um ou mais serviços oferecidos pelo servidor.

A arquitetura cliente/servidor pode ser classificada como: cliente/servidor com processos, cliente/servidor com objetos distribuídos (RMI, CORBA, DCOM,...), cliente/servidor com objetos para web (XML, RPC/XML, WSDL, SOAP/XML, UDDI).

A estrutura cliente/servidor, com objetos distribuídos, pospõe limitações para funcionar na web. A web foi concebida para tirar vantagem da Internet e a computação distribuída como foi concebida, não foi pensado para a web.

A web é basicamente desconectada, isto é, as conexões são temporárias.

Os serviços de computação distribuída, como segurança e transações, que necessitam de conexões ao nível de transporte, precisam ser projetadas para oferecer tais funcionalidades para as características da web desconectada.

Na web as partes podem se conectar sem conhecimento uma da outra, seguindo os links URL e observando poucas regras básicas. Nas tecnologias de computação distribuída o acoplamento cliente/servidor se dá de forma mais firme e não pode, portanto, tirar vantagem da web hoje existente.

Na computação distribuída convencional, as partes precisam conhecer as outras seguindo as referências de objetos-servidores passadas por esses para o lado cliente, ou providas por um servidor de nomes remoto (que não está no lado cliente), o qual fornece referências, e através dos quais, objetos de aplicação são localizados e podem então ser acessados.

Os Web Services adotam o modelo de publicação na web, sendo então possível um ponto específico (endereço do serviço), usando uma definição de interface de serviços para a web, não sendo necessário um tipo específico de cliente para acessar este serviço.

Para os Web Services qualquer cliente pode acessar qualquer serviço disponível na web, desde que as informações sobre o serviço estejam disponíveis e sejam compreensíveis e possam ser geradas mensagens, através de processadores XML.

Este paradigma possui a vantagem de permitir a integração cooperativa de aplicações. Os clientes podem desenvolver e integrar aplicações posteriormente.

A próxima geração da web é voltada para a busca de informações exatas sobre strings de texto embutidas em páginas web. Soluções para a comunicação entre aplicações devem derivar das tecnologias atuais da Internet. Os pontos finais da web, os endereços URL, irão proporcionar

serviços para o processamento de dados XML, da mesma maneira que os browser atuais processam texto html.

Os serviços web devem fazer referência dinâmica a URL's e também ler e gerar esquemas XML automaticamente.

Grandes empresas já se dedicam a desenvolver os tipos de serviços padrões que são acessíveis para qualquer programa. O objetivo é a integração de aplicações que utilizam serviços pré-definidos na web.

Para isto é necessária uma padronização no desenvolvimento destes serviços. Já existem iniciativas, tais como do W3C para web.

Os web services necessitam de diversas tecnologias baseadas em XML, tais como:

- XML (extended markup language) – Representa uma série de especificações publicadas e mantidas pelo W3C;
- WSDL (web services description language) – Tecnologia baseada em XML para especificação de interfaces de web services, dados e tipos de mensagens, modelos de interação e mapeamento de protocolos;
- SOAP (simple object access protocol) – Tecnologias baseadas em XML que define um envelope para a comunicação de serviços na web;
- UDDI (universal description, Discovery and integration) – Permite o registro e descoberta de serviços na web.

2. Escopo

Tanto o avanço da Internet como a utilização de web services, fez com que várias ferramentas e plataformas de desenvolvimento destes serviços surgissem no mercado.

O projeto proposto visa analisar algumas ferramentas para o desenvolvimento de web services, estabelecendo um comparativo entre elas.

As ferramentas e plataformas analisadas serão as seguintes:

- ETTK (Emerging Techlogies ToolKit – IBM);
- AXIS – Projeto Jakarta (Apache);
- GLUE – The Mind Eletric;
- JWSDP – Java Web Services Developer Pack.

3. Objetivos

Avaliar várias plataformas de desenvolvimento de web services, através de parâmetros pré definidos e criar uma metodologia para uma análise comparativa das plataformas citadas, avaliando como principais aspectos:

- Desempenho;
- Desenvolvimento de aplicações específicas tais como B2B;

- Utilização;
- Instalação;
- Ambiente de Desenvolvimento.

4. Problema a ser resolvido

Como as plataformas de desenvolvimento de web services são fornecidas por diversas empresas, nem sempre é uma tarefa fácil fazer uma escolha sobre qual utilizar.

O problema principal está focado na falta de métricas de comparação entre elas, tornado a escolha, muitas vezes, baseada em parâmetros abstratos ou também na experiência de utilização dos desenvolvedores, quando são ignoradas as características benéficas e as vantagens técnicas de cada plataforma.

5 Uma visão geral dos Serviços Web

Nos dias de hoje seria difícil imaginar a computação em rede sem o uso da internet. O sucesso da difusão da internet se baseia em dois pontos básicos: sua simplicidade e presença em todo o mundo. Sob a visão de um fornecedor de serviços, com a criação de uma página na web ele pode ser visto em todo o mundo. Do ponto de vista do usuário, basta possuir um computador conectado à web para acessar uma vasta gama de serviços com facilidade.

Por serviços web pode-se entender a união e reutilização de vários serviços que estão disponíveis na internet que executam tarefas específicas. Podemos citar como exemplo um site de comparação de preços de produtos que se utiliza dos serviços web de fornecedores, obtendo informações de suas bases de dados corporativas e concentrando o resultado em uma única página.

Os serviços web são uma nova geração de aplicações. São aplicações modulares, independentes e auto-explicativas, as quais podem ser publicadas, localizadas e invocadas através da internet. Os serviços web executam funções que podem ir de uma simples até complexos processos de negócios. Uma vez que um serviço web é distribuído, outras aplicações (e outros serviços web) podem encontrá-lo, invocá-lo e utilizar seus recursos.

5. A Plataforma de Serviços WEB

A plataforma básica dos serviços web é composta pelo XML mais o protocolo HTTP. O protocolo HTTP se encontra em qualquer lugar, executando praticamente tudo na internet. O XML fornece uma meta linguagem na qual podemos escrever códigos especializados para representar iterações complexas entre clientes e serviços ou entre componentes que formam um serviço. Por trás da fachada de um servidor web, uma mensagem XML é convertida para uma requisição que conhecida pelo servidor e o resultado convertido novamente para XML.

É claro que o processo de requisição e resposta vai além do procedimento descrito. A plataforma de serviços web completa envolve outras tecnologias e procedimentos, tais como:

- SOAP – Chamadas Remotas;
- WSDL – Definição das características do serviço;
- UDDI. – Serviço de publicação e procura de serviços.

5.1. SOAP

O SOAP é a especificação de um protocolo que define uma maneira para a passagem de dados codificados em XML (Ethan, 2002). Também define maneiras para facilitar a chamada remota de funções via Internet, permitindo que dois programas se comuniquem de forma muito semelhante à invocação de páginas Web. Submetido em 2003 ao W3C pela IBM, Microsoft, Userland e DevelopMentor, o futuro do desenvolvimento do SOAP agora esta entregue aos cuidados do grupo de trabalho do protocolo XML do W3C (W3C's Protocols Working Group). Efetivamente, isto significa que a atual versão do SOAP está parada e estável até que o grupo de trabalho do W3C forneça uma nova especificação.

5.2. UDDI (Universal Description, Discovery and Integration Service).

O UDDI fornece um mecanismo para a descoberta dinâmica de serviços web. Utilizando a interface do UDDI, negócios podem se conectar dinamicamente a outros serviços fornecidos por parceiros externos. Um registro UDDI é similar a um registro CORBA, ou pode ser imaginado com um serviço DNS para uma aplicação de negócios. Um registro UDDI possui dois tipos de clientes: negócios que querem publicar um serviço (e sua interface), e clientes que querem obter serviços de certo tipo. O UDDI é desenvolvido sobre o SOAP e assume que as requisições e respostas são objetos UDDI enviados como mensagens SOAP. Abaixo é oferecida uma visão geral do que é fornecido pelo UDDI.

Tabela 3 - Serviços UDDI. (Ethan, 2002, p. 135-136).

Informação	**Operação**	**Informação Detalhada** (suportada por API de baixo nível)
Páginas Brancas: Fornece o nome, endereço, telefone e outras informações de contato de um determinado negócio.	**Publicação:** Como um fornecedor de serviço registra suas informações.	**Informação do negócio:** Armazenada em um objeto *BusinessEntity*, o qual contém informações sobre os serviços, categorias, contatos, URLs e outros dados necessários para interagir com a empresa.
Páginas amarelas: Informação que determina a categoria do negócio.	**Descoberta:** Como uma aplicação descobre um serviço web em particular.	**Informação do Serviço:** Descreve um grupo de serviços web. Estes estão contidos em um objeto *BusinessService.*
Páginas verdes: Informações técnicas sobre um serviço web.	**Conexão:** Como uma aplicação conecta e interage com um serviço web após sua descoberta.	**Informação de conexão:** Os detalhes técnicos necessários para invocar um serviço web. Isto inclui URLs, informações sobre os nomes dos métodos, tipos de argumentos e assim por diante. Estes dados são representados no objeto *BindingTemplate.* **Detalhes de Especificação de Serviço:** São os meta dados sobre as várias especificações implementadas por um serviço web. São chamados *tModels* na especificação UDDI.

Fonte: Adaptado de Ethan (2002).

5.3. WSDL (Web Services Definition Language).

O WSDL fornece uma maneira para que os provedores de serviços descrevam o formato básico dos serviços requisitados sobre diferentes protocolos e codificações (Ethan, 2002). O WSDL é usado para descrever o que um serviço web é capaz de fazer, onde ele está hospedado e como invocá-lo. Enquanto a declaração do SOAP/HTTP pode ser feita utilizando várias especificações, o WSDL faz mais sentido por assumir SOAP/HTTP/MIME como mecanismo de invocação de objetos remotos. Os registros UDDI descrevem vários aspectos dos serviços web, incluindo detalhes obrigatórios do serviço. O WSDL se ajusta como um subsistema da descrição de serviço UDDI.

O WSDL define os serviços com coleções de pontos finais ou portas na rede. Em WSDL a definição abstrata de pontos finais e mensagens são separadas pela distribuição concreta da rede ou pelo formato obrigatório dos dados. Isto permite a reutilização da definição abstrata de mensagens, as quais são descrições abstratas dos dados que estão sendo trocados, e dos tipos de portas, as quais são coleções abstratas de operações. O protocolo concreto e especificação do formato dos dados constituem uma ligação reutilizável. Uma porta é definida pela associação de um endereço na rede com uma ligação reutilizável; uma coleção de porta define um serviço. E, assim, um documento WSDL utiliza os seguintes elementos para a definição de serviços:

- Tipos – um repositório para definição de tipos de dados utilizando alguns tipos de sistemas (tal como XSD);
- Mensagem – Um tipo abstrato de definição dos dados que estão sendo comunicados;
- Operação – Uma descrição abstrata de uma ação suportada pelo serviço;
- Tipo de Porta – Um conjunto abstrato de operações suportadas por um ou mais pontos finais da rede;
- Ligação – Um protocolo concreto e especificação de formato de dado para um tipo particular de porta;
- Porta – Um ponto final simples definido como uma combinação de uma ligação e um endereço na rede;
- Serviço – Uma coleção de pontos finais relacionados.

Em resumo, o WSDL é um modelo para mostrar como os serviços podem ser descritos e conectados pelos clientes.

6. As Plataformas

6.1 – AXIS

A plataforma Axis consiste basicamente de vários subsistemas

funcionando conjuntamente e pretendemos a seguir dar uma visão geral de como isto funciona.

Visto de uma forma simples, Axis consiste basicamente do processamento de mensagens. Quando o processamento central do Axis inicia, uma séria de manipuladores de mensagens são invocados em ordem. A ordem em que são chamados depende de 2 fatores – a configuração em que são desenvolvidos e distribuídos e se a configuração funciona como um cliente ou servidor.

O objeto que é enviado para cada manipulador de mensagens é um "MessageContext". Um "MessageContext" é uma estrutura que contem várias partes: a) uma mensagem de solicitação, b) uma mensagem de resposta, e c) um conjunto de propriedades.

Há basicamente duas maneiras de invocar o Axis:

3) Como um servidor, um "transport listener", que irá criar um "MessageContext" e irá chamar o framework do Axis sempre que for solicitado;
4) Como um Cliente, neste caso uma aplicação que irá gerar um "MessageContext" e irá chamar o framework do Axis sempre que for solicitado;

Em qualquer uma das situações, o framework do Axis irá simplesmente enviar o "MessageContext" resultante através dos manipuladores de mensagens configurados, onde cada um deles irá processar as ações determinadas sobre o "MessageContext".

A figura abaixo mostra o lado servidor. Os cilindros menores representam manipuladores de mensagens e o cilindro maior, com cilindro menores em seu interior, represente uma corrente de manipuladores de mensagens (coleção de manipuladores ordenados de forma pré-determinada).

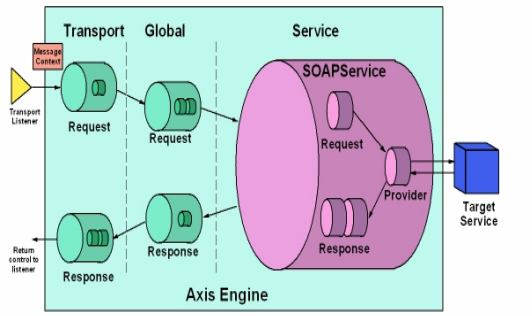


Figura 80 - Mecanismo do Axis – Servidor. Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

Uma mensagem chega (em um manipulador específico de protocolo) através de um "Transport Listener". Neste caso assumiremos que o "Listener" é um servlet http. Este é o processo do "Listener" para "empacotar" os dados de um protocolo específico em um objeto Mensagem (org.apache.axis.Message), e colocar a mensagem em um "MessageContext". O "MessageContext" é carregado com várias propriedades pelo "Listener" – como por

exemplo a propriedade “http.SOAPAction” será atribuída como valor do atributo SOAPAction do cabeçalho http. O “Transport Listener” também atribui a cadeia “transportName” ao “MessageContext”. Uma vez que o objeto “MessageContext” está pronto para ser enviado o “Listener” o passa ao mecanismo do Axis (Gros, 2003).

O primeiro procedimento do mecanismo do Axis é procurar o transporte pelo nome. O Transporte é um objeto que contem uma cadeia “request” e uma cadeia “response” e, em alguns casos, ambas. Uma cadeia consiste é um manipulador de mensagens formado por uma seqüência de manipuladores. Se uma cadeia “request” de transporte existe, esta será chamada passando o “MessageContext” para um método invoke(). Isto resultará na chamada de todos os manipuladores especificados na cadeia “request”.

Após a manipulação do “request” do transporte, o mecanismo do Axis aloca uma cadeia “request” global e então pode invocar qualquer um dos manipuladores configurados no “request” global.

Dentro deste contexto, um dos manipuladores irá especificar o valor do campo “serviceHandler” no objeto “MessageContext” (normalmente isto é feito pelo manipulador “URLMapper” o qual mapeia uma URL como por exemplo, http://localhost:8080/axis/service/AdminService para o serviço “AdminService”). Este campo determina o manipulador que será chamado para executar uma funcionalidade específica, como por exemplo, fazer uma chamada RPC em um objeto back-end. Na plataforma Axis, serviços são comumente instancias de cadeias “request” e “response” e devem possuir um provedor, que é um manipulador de mensagens responsável pela implementação da lógica do serviço.

Para “requests” RPC, o provedor é uma classe org.apache.axis.provider.java.RPCProvider. Também se trata de um manipulador de mensagens que, quando invocado, tenta chamar uma classe java que é determinada pelo parâmetro “className” especificada durante o desenvolvimento. É utilizado o padrão do SOAP RPC para a especificação do método chamado e para verificar se os tipos dos argumentos de entrada, codificados em XML, são compatíveis com os parâmetros resultantes do método chamado.

O caminho da mensagem do lado cliente é similar ao lado servidor, exceto pelo fato de a ordem de envio e recebimento de mensagens, fluírem em sentido oposto.

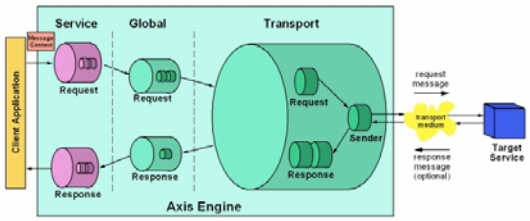


Figura 81 - Mecanismo Axis – Cliente

Fonte: Axis Architecture Guide. http://ws.apache.org/axis/java/architecture-guide.html.

O manipulador “Service” é chamado primeiro – no lado cliente não há um provedor, uma vez que o serviço está sendo disponibilizado por um nó remoto, mas ainda há a possibilidade da existência de cadeias

"request" e "response". As cadeias "request" e "response" existentes na camada "Service" executam qualquer processamento específico na requisição da mensagem para fora do sistema, assim como na mensagem de resposta que retorna ao sistema.

Depois da cadeia "request" da camada "Service", a cadeia "request" da camada "Global" é chamada, seguida pela camada "Transport". O "Transport Sender", um manipulador especial cuja função é obter as mensagens que entram e saem do servidor SOAP, independente da especificação de protocolo. A resposta ("response"), se existir, é colocada no campo "responseMessage" do objeto "MessageContext" e o objeto é propagado através da cadeia "response", passando pelas camadas "Trasnport", "Global" e "Service", nesta ordem.

6.2 – JWSDP

O pacote de desenvolvimento de Serviços web Java (JWSDP) é uma coleção de ferramentas e bibliotecas projetadas para o desenvolvimento de serviços web em Java de forma simples e eficiente. Apresentado pela primeira vez em janeiro de 2002 como um release "Early Access" (EA), o JWSDP trouxe junto a análise de XML, suporte ao SOAP, um recipiente de serviços ("Service Container") e ferramentas integradas que podem ser usadas para criar e utilizar uma grande quantidade de serviços distribuídos baseados em protocolo XML (FOSTER, 2002).

Atualmente encontra-se à disposição o release EA2, o qual contém os seguintes componentes:

- Bibliotecas de processamento e de mensagens JAXP XML;
- Bibliotecas para chamada de procedimentos remotos JAX-RPC XML;
- Bibliotecas JAXR para acesso à registros XML;
- Biblioteca Java de sockets seguros (JSSE);
- Biblioteca de tags padrões JSP;
- Ferramenta para construção Apache/Tomcat Ant;
- O container Apache/Jakarta Tomcat Servlet;
- Um registro UDDI simples (WSDP Registry Tool);
- Uma ferramenta de distribuição de aplicações Web.

É de conhecimento geral que partes do JWSDP não são desenvolvidas pela SUN. Na verdade, o JWSDP é, em sua maioria, uma coleção de produtos já existentes; o JWDSP agrupa estas ferramentas de uma forma conveniente para sua instalação. Também, com o JWSDP, existem versões das ferramentas que funcionam juntas, evitando um grande número de problemas quando tentamos executar algum código.

Como citado, o JWSDP é uma coleção de ferramentas já existentes e, por

certo ponto de vista, uma jogada de marketing da Sun em resposta à plataforma .Net da Microsoft, unindo e colocando em evidência tecnologias já existentes e dando a elas um nome comum.

Na maioria das vezes isto vem em benefício do usuário. Não é mais necessário se preocupar com o que as APIs estão fazendo para sobreviver ao mercado; as APIs existentes no JWSDP estão sob proteção da Sun e possuem boa chance de se tornarem padrões. Também não há necessidade de preocupar-se com as versões e compatibilidade entre diferentes bibliotecas, uma vez que estas vem junto com o JWSDP.

Existem alguns lugares onde a união de bibliotecas resulta em algumas coisas estranhas. Particularmente as bibliotecas JAXM, JAXP e JAX-RPC que são desenvolvidas por iniciativas separadas, pelo projeto Apache XML e pela comunidade Java.

Como são desenvolvidas separadamente, há lugares onde as APIs se sobrepõe. Por exemplo, algumas funcionalidades presentes no JAXM também existem no JAXP.

Outro aspecto que pode ser confuso sobre o JWSDP é que algumas de suas partes estão disponíveis em diferentes formas na Web. Por exemplo, a biblioteca JAXP é distribuída como parte do release 1.4 do JDK. O JWSDP inclui estas bibliotecas para ser compatível com versões anteriores do JDK, especialmente aquelas que irão demorar algum tempo para migrar paras as versões mais novas.

Será dada na seqüência uma visão geral dos componentes do JWSDP.

O JWSDP fornece uma coleção de bibliotecas e ferramentas projetadas para dar ao usuário tudo que ele necessita para iniciar o desenvolvimento e testes de serviços web. Além das bibliotecas de interface padrões , referências às implementações de cada biblioteca são fornecidas. Em alguns casos (JAXP), estas referências estão voltadas à qualidade de desenvolvimento; em outros casos (JAXM), são apenas suficientes ao desenvolvimento sem se preocupar com a qualidade deste. Em todos os casos, as interfaces são projetadas para permitir a mudança de implementações por versões alternativas.

O JWSDP também fornece algumas ferramentas que facilitam o desenvolvimento de serviços web. Elas não substituem servidores de produção (como o WebSphere da IBM ou BEA WebLogic) e outras ferramentas, mas permitem que se inicie no desenvolvimento de serviços web.

Uma vez obtido os arquivos de instalação do JWSDP, o usuário tem tudo que precisa para o desenvolvimento de serviços web.

6.3 – ETTK

A plataforma ETTK (IBM Emerging Technologies Toolkit) é um kit para o desenvolvimento de software voltado ao projeto, desenvolvimento e execução de serviços web. A plataforma ETTK fornece um ambiente no qual é possível executar exemplos de tecnologias emergentes que

demonstram recentes especificações e protótipos das equipes de desenvolvimento e pesquisa em tecnologia da IBM.

Além disso, a plataforma torna disponível material introdutório que permite aos desenvolvedores se familiarizar com o desenvolvimento de tecnologias autônomas e serviços web.

A plataforma ETTK se originou do pacote da IBM conhecido como WSTK ("Web Services Tookit"). Com a mudança de denominação do pacote WSTK para ETTK, o escopo das tecnologias inclusas no pacote foi ampliado para incluir tecnologias independentes. O novo kit de ferramentas reúne tecnologias relacionadas de vários laboratórios de pesquisa e desenvolvimento da IBM. Os programas de demonstração e funções do ETTK podem ser executados tanto no sistema operacional Linux como no Windows®.

6.3.1- Funcionamento

Os componentes de software básico necessários para a criação de serviços web e programas independentes são fornecidos com o ETTK. Junto há uma visão geral da arquitetura de tecnologias independentes, serviços web, programas de exemplo, utilitários e algumas ferramentas que são úteis no desenvolvimento e distribuição de programas independentes e serviços web. O kit de ferramentas também inclui o mecanismo SOAP e um servidor de aplicações.

O ETTK pode ser executado nos sistemas operacionais Windows® 2000, Windows Server 2003, Windows XP, RedHat Enterprise Linux para plataforma Intel 2.1 ou UnitedLinux 1.0.

6.3.2 - Conteúdo do ETTK

O pacote ETTK contém os seguintes recursos:

- Infra-estrutura para serviços webs e programas independentes, exemplos e documentação para que programadores possam criar seus próprios programas independentes;
- Ambiente pré-instalado (versão do WebSphere Application Server-Express e JDK);
- APIs para o lado cliente para comunicação com documentos WSDL ou um registro UDDI; e APIs para publicação de serviços web;
- Implementação de protótipos de especificações recentemente anunciadas de serviços web;
- Demonstrações que podem ser usadas para o teste do ETTK; Publicação de um serviço e a utilização de um cliente que se comunica com um registro UDDI para encontrar o serviço web e invocá-lo;

- Um conjunto de utilitários para serviços web: Perfil de usuário, medições, contabilidade, contrato e notificação. Junto é fornecido um demonstrativo, que utiliza estes utilitários e faz uso do nível de acordo de serviços web (WSLA);
- Uma visão das funções de segurança da tecnologia, tais como WS-Security, gerenciamento de sistemas e tecnologias SOAP;
- Documentação sobre a arquitetura de serviços web, WS-Inspection e especificações WSDL;

Um tutorial para o desenvolvimento de serviços web.

6.4 – GLUE

A plataforma webMethods Glue™ é uma plataforma projetada para o desenvolvimento e distribuição de aplicações com serviços web, JSPs e servlets.

A plataforma Glue pode ser utilizada de duas formas: standalone e modo Hosted. No modo standalone, Glue é um servidor de aplicações que, segundo seus desenvolvedores, procura ser mais simples que a utilização de EJB (Entreprise Java Beans). No modo Hosted pode se conectar a outros servidores de aplicação ou servlet e fornecer aos serviços web.

A figura abaixo fornece uma visão geral da arquitetura da plataforma.

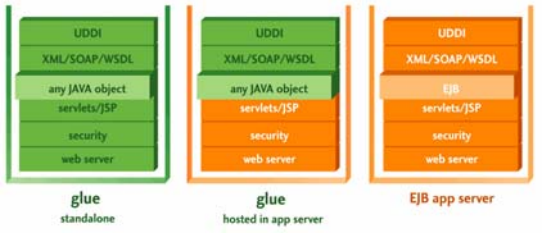


Figura 03 - Arquitetura da Plataforma webMethods Glue

Fonte: WebMethods Glue User Guide http://www1.webmethods.com/docs/glue/guide/index.html

Existem 3 edições da plataforma Glue, e todas, com exceção da edição Standard estão disponíveis para instalação em mainframes.

- Standard : Sem suporte e livre para uso comercial, utilizado para prototipação e projetos de baixo custo
- Professional : Plataforma de classes de negócio, utilizada para projetos robustos e de alta performance de desenvolvimento.
- Enterprise : Possui clusterização de Servlets, JSPs e objetos Java

plenos, evitando a complexidade de EJB.

A plataforma webMethods Glue™ também fornece a ferramenta webMethods Fabric™ que permite às organizações migrarem sua aplicações legadas para a arquitetura orientada a serviços da Glue.

Uma vez realizado o estudo teórico das plataformas de desenvolvimento de serviços web, o objetivo para a próxima etapa é a especificação de uma aplicação que será desenvolvida utilizando as plataformas apresentadas.

Serão analisados aspectos tais como:

- Segurança;
- Ambiente de Desenvolvimento;
- Recursos de Software e Hardware necessários;
- Desempenho;
- Integração com outras tecnologias de computação distribuída;
- Limitações;
- Avaliação das plataformas para o desenvolvimento de serviços web em ambientes corporativos;
- Avaliação das plataformas para o desenvolvimento de B2B.

Vale ressaltar que outras medidas de desempenho poderão surgir ao longo do desenvolvimento do protótipo, em virtude de necessidades ainda não detectadas até o momento.

7. Avaliações

Para todas as ferramentas propostas foram feitas avaliações de acordo com as metas traçadas, através da implementação de uma aplicação simples: Uma calculadora, com as quatro operações básicas – Soma, adição, subtração e multiplicação d acordo com as classes a seguir:

```
public interface ICalculadora extends Remote
{

  public int getSoma(int i1, int i2);

  public int getSub(int i1, int i2);

  public int getDiv(int i1, int i2);

  public int getMult(int i1, int i2);
}
```

```
public class Calculadora implements
ICalculadora {
   /** Creates a new instance of Calculadora
*/
  public int getSoma(int i1, int i2) {
    return i1 + i2;
  } //getSoma
  //-----------------------------------------
  public int getSub(int i1, int i2) {
    return i1 - i2;
  } // getSub
  //-----------------------------------------
```

```
  public int getDiv(int i1, int i2) {
     return i1/i2;
  } // getDiv
  public int getMult(int i1, int i2) {
   return i1*i2;
  } //getMult
} //Calculadora
```

8. Conclusões e Trabalhos futuros

8.1 Conclusões.

As plataformas avaliadas foram selecionadas com o objetivo de fornecerem uma visão abrangente do processo de desenvolvimento de web services.

Na proposta inicial definimos os seguintes parâmetros de avaliação:

- Desempenho;
- Desenvolvimento de Aplicações Específicas;
- Utilização;
- Instalação;
- Ambiente de Desenvolvimento.

Durante o transcorrer do trabalho os parâmetros foram reavaliados e observamos a necessidade da redefinição dos mesmos. Assim decidimos por não avaliar a performance e a segurança das aplicações desenvolvidas, uma vez que seria necessária a criação de cenários específicos e idênticos e envolveriam aspectos que não são foco deste trabalho, como por exemplo, as configurações de hardware, sistemas operacionais, estrutura de rede, etc.

Assim sendo, optamos pelos seguintes parâmetros de avaliação:

- Instalação;
- Documentação;
- Dificuldade de Aprendizado;
- Velocidade de Desenvolvimento;
- Custo.

A instalação das plataformas não chega a ser um fator determinante para a escolha de um dos produtos. Apesar da instalação de algumas apresentarem-se mais complexas, não necessitam de muita experiência.

A plataforma AXIS não possui um arquivo instalador, sendo necessário conhecimento sobre a configuração e estrutura de diretórios do servidor web Tomcat. As plataformas JWSDP, GLUE e ETTK possuem um arquivo instalador, sendo que o ETTK permite também uma instalação sem uma interface gráfica, contudo, dificultando seu uso e a configuração do servidor web websphere. É necessário salientar que o recurso GUI do ETTK necessita da ferramenta RAD, cujos arquivos de instalação possuem 2,82 Gigabytes. Com o uso de uma conexão banda larga ADSL de 300 KBps, foi necessário em torno de 10 horas para completar o download dos arquivos.

Segundo nossa avaliação, consideramos as plataformas AXIS, GLUE e JWSDP como de baixa complexidade para instalação e a plataforma ETTK com média complexidade.

Outro aspecto avaliado foi a documentação. Este é um parâmetro vital para qualquer plataforma de desenvolvimento. Observamos que a documentação das plataformas ainda deixa a desejar. Entretanto um fator favorável está em que todas elas se baseiam e utilizam o ambiente JAVA, que é muito bem documentado.

Encontramos vasto material de referência, tanto no site da SUN Microsystems (http://www.sun.com), na vasta bibliografia existente assim como na internet.

Neste aspecto sobressaiu a plataforma JWSDP, com documentação bem estruturada e de fácil navegação.

A plataforma AXIS, apesar da grande quantidade de documentação, já não apresenta a mesma qualidade em sua estrutura e navegação, o que pode ser decorrente do fato de ser uma plataforma aberta, com grande quantidade de colaboradores e desenvolvedores, o que dificulta a manutenção e atualização da documentação (POTTS&KOPACK, 2003).

Uma característica observada em todas as plataformas foi a pouca quantidade de livros publicados sobre cada uma delas, sendo a maior parte da documentação encontrada nos sites dos fornecedores.

Também avaliamos a Dificuldade de aprendizado em cada uma das plataformas e observamos que todas necessitam de uma experiência razoável em desenvolvimento. Com algumas delas é possível colocar um web service simples em funcionamento sem muito esforço. Em outras, o desenvolvimento requer um aprendizado maior. As plataformas comerciais (ETTK e GLUE) oferecem um maior suporte à interface gráfica. Contudo com o lançamento da IDE Netbeans 4.1RC2 (http://www.nerbeans.org), com o JWSDP incorporado, tornou esta uma opção interessante entre as plataformas gratuitas.

Em relação à velocidade de desenvolvimento, podemos defini-la como sendo o tempo que um desenvolvedor necessita para terminar uma determinada tarefa, após possuir o domínio de uma ferramenta.

Quanto maior a automação da ferramenta, maior será sua velocidade de desenvolvimento. Contudo observamos que seu custo aumenta na mesma proporção (POTTS&KOPAC, 2003). Também observamos que a migração entre plataformas gratuitas tem um impacto menor do que em plataformas comerciais.

O custo é um fator relevante em qualquer organização. No entanto não pode ser considerado isoladamente quando da aquisição de uma plataforma de desenvolvimento. Outros aspectos necessitam ser considerados, sendo muitas vezes mais importantes. Podemos citar o treinamento da equipe de desenvolvimento, suporte, estrutura de hardware e manutenção a longo prazo dos sistemas desenvolvidos (POTTS&KOPACK, 2003). Como não foi possível medir os outros fatores citados, nossa avaliação se restringiu ao custo de aquisição das plataformas.

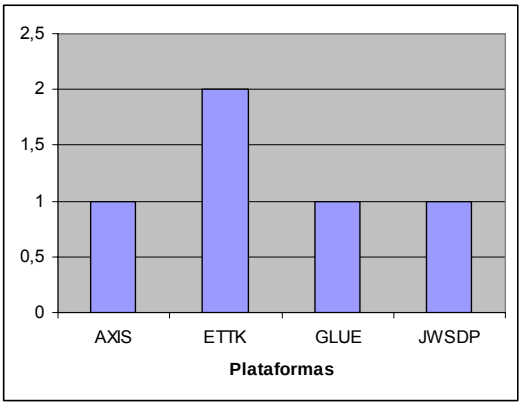
Na Tabela 2, mostramos uma comparação entre as plataformas, considerando-se os parâmetros avaliados.

Tabela 4 - Comparação Entre as Plataformas de Desenvolvimento

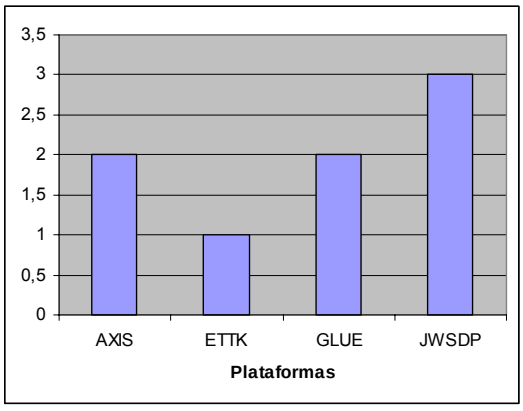
Plataforma	***Complexidade da Instalação***	***Qualidade da Documentação***	***Dificuldade de Aprendizado***	***Velocidade de Desenvolvimento***	Custo
AXIS	(1) Baixa	(2) Regular	(2) Média Para Programadores Java	(2) Média Para Programadores Java	Gratuito
ETTK	(2) Média	(1) Ruim	(1) Fácil	(3) Rápida Para Programadores Java	$3.000,00
GLUE	(1) Baixa	(2) Regular	(1) Fácil Para Programadores Java	(2) Média Para Programadores Java	Gratuito
JWSDP	(1) Baixa	(3) Boa	(2) Média Para Programadores Java	(3) Rápida Para Programadores Java	Gratuito

Considerando os valores das avaliações das plataformas, criamos gráficos comparativos dos parâmetros avaliados, como mostrados nos quadros a seguir:

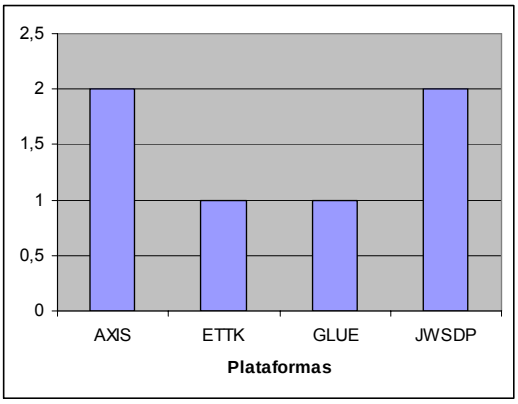
Quadro 01 - Complexidade da Instalação.



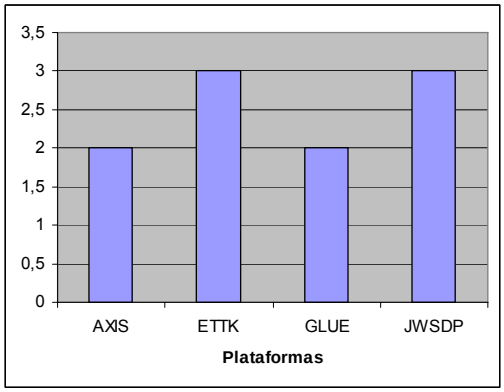
Quadro 02 - Qualidade da Documentação.



Quadro 03 - Dificuldade de Aprendizado.



Quadro 04 - Velocidade de Desenvolvimento.



Pelo exposto neste trabalho, os autores consideram ter atingido de forma satisfatória os objetivos propostos neste trabalho. Durante o transcorrer do mesmo, alguns ajustes nos parâmetros de avaliação se tornaram necessários, até porque algumas plataformas tornaram disponíveis versões e recursos não existentes no início de nossa pesquisa.

Concluímos, portanto, considerando-se o desenvolvimento baseado no ambiente JAVA e o conjunto de parâmetros usados em nossa avaliação, que a melhor opção para o desenvolvimento de web services encontra-se na adoção da plataforma JWSDP.

8.2 Trabalhos Futuros.

O uso de web services vem aumentando consideravelmente pelas empresas que desejam tornar suas informações disponíveis na internet. É um campo de pesquisa contínuo e este trabalho poderá ser estendido para a avaliação de outras plataformas, assim como a análise de outras características não abordadas nesta pesquisa.

9. BIBLIOGRAFIA.

ALPHAWORKS. **Emerging Technologies Toolkit.** Disponível em http://www.alphaworks.ibm.com/tech/ettk

ALVES, Maria Bernadete Martins; ARRUDA, Suzana Margareth. **COMO FAZER REFERÊNCIAS: bibliográficas, eletrônicas e demais formas de documentos.** Disponível em: http://bu.ufsc.br/framerefer.htm

ARMSTRONG, Eric; et. al. – **The Java Web Services Tutorial**.July 25, 2003.

AXIS**. Axis Documentation.** Disponível em http://ws.apache.org/axis/java/index.html

_____.**Axis Architecture Guide.** Disponível em http://ws.apache.org/axis/java/architecture-guide.html

CERAMI, Ethan – **Web Services Essentials. Distribuited Applications with XML-RPC, UDDI & WSDL.** O´Reilly. Janeiro 2003.

FOSTER, Jay; et. al.- **Developing Web Services With Java APIs form XML Using WSDP.** Syngres. 2002. Disponível em http://www.syngres.com

CHAPPELL, David; JEWEL, Tyler. **Java Web Services.** O´Reilly. Março 2002.

GROS, Christian – **Web Service Programming Using Axis.** O´Reilly, Open Source Convention. Julho 7 – 11, 2003.

POTTS, Stephen; KOPACK, Mike. **Aprenda em 24 horas Web Services.** Tradução de Marcos Vieira. Campus. Rio de Janeiro. 2003.

SILVA, Edna Lúcia da; MENEZES, Estera Muszkat. **Metodologia da Pesquisa e Elaboração de Dissertação.** UFSC/PGED/LED. 3 ed. Florianópolis. 2001.

SUN. **The Java Web Service Tutorial 1.0**. Disponível em http://java.sun.com/webservices/docs/1.0/tutorial/

TECHMETRIX RESEARCH. **AXIS: The New Incarnation of Apache SOAP – A Technical White Paper.** Disponível em http://www.techmetrix.com

TOPLEY, Kim – **Java Web Services in Nutshell.** O´Reilly, Junho 2003.

XML FROM THE INSIDE OUT. **XML development, XML resources, XML specifications.** Disponível em http://www.xml.com/index.csp

WEBMETHODS. **Glue 5.0.2 User Guide.** Disponível em http://www1.webmethods.com/docs/glue/guide/index.html

W3C**. World Wide Web Consortium.** Disponível em http://www.w3.org