UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

Relatórios Técnicos
do Departamento de Informática Aplicada
da UNIRIO
n°0015/2009

Estudo de Ferramentas da BEA para SOA

Jairo Francisco de Souza
Leonardo Azevedo
Fernanda Baião
Flávia Santoro

Departamento de Informática Aplicada

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
Av. Pasteur, 458, Urca - CEP 22290-240
RIO DE JANEIRO – BRASIL

Projeto de Pesquisa

Grupo de Pesquisa Participante

Patrocínio

PETROBRAS

Relatórios Técnicos do DIA/UNIRIO, No. 0015/2009
Editor: Prof. Sean W. M. Siqueira Agosto, 2009

Arquitetura Orientada a Serviço – Estudo de Ferramentas da BEA para SOA *

Jairo Francisco de Souza[1,3], Leonardo Azevedo[1,2], Fernanda Baião[1,2], Flávia Santoro[1,2]

[1]Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec)
[2]Departamento de Informática Aplicada (DIA) – Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
[3]Departamento de Ciência da Computação (DCC) – Universidade Federal de Juiz de Fora (UFJF)

jairofsouza@gmail.com, azevedo@uniriotec.br, fernanda.baiao@uniriotec.br, flavia.santoro@uniriotec.br

Abstract. Service Oriented Architecture is presented as being more flexible and capable to support services independent of platform and protocol in a distributed environment. In this work, we analyze the tools of BEA that are most relevant for SOA initiative implementation: Aqualogic Service Bus, Aqualogic Service Registry e Aqualogic BPM. Moreover, we describe the most important aspect of SOA provided by the tools.

Keywords: Service Oriented Architecture, service

Resumo. A arquitetura orientada a serviços (SOA – Service Oriented Architecture) apresenta-se como sendo mais flexível e capaz de suportar serviços independentes de plataforma e protocolo em um ambiente distribuído. Neste trabalho analisamos as ferramentas da BEA que são mais relevantes para a implantação de uma iniciativa SOA: Aqualogic Service Bus, Aqualogic Service Registry e Aqualogic BPM. Além disso, relatamos os aspectos mais importantes de SOA que a ferramenta atende.

Palavras-chave: Arquitetura orientada a serviço, serviço.

* Trabalho patrocinado pela Petrobras.

Sumário

Índice de Figuras

1 Introdução

Neste relatório é apresentado o estudo realizado de ferramentas da BEA mais relevantes para a implantação de uma iniciativa SOA: Aqualogic Service Bus, Aqualogic Service Registry e Aqualogic BPM. Estas ferramentas foram analisadas e suas principais funcionalidades foram explicitadas. Além disso, os aspectos mais importantes de SOA que a ferramenta atende foram relatados. As informações apresentadas neste relatório foram baseadas em diversos manuais disponibilizados pela BEA[1].

Este relatório foi produzido pelo Projeto de Pesquisa em SOA como parte das iniciativas dentro do contexto do Projeto de Pesquisa do Termo de Cooperação entre NP2Tec/UNIRIO e a Petrobras/TIC-E&P/GDIEP.

1.1 Conceitos necessários

Esta seção descreve algumas especificações definidas por consórcios internacionais e utilizadas para desenvolvimento de serviços. Ferramentas que fornecem soluções para um ambiente SOA implementam todas ou parte dessas linguagens.

XML Schema é uma linguagem baseada no formato XML para definição de regras de validação ("esquemas") em documentos no formato XML. A XMLSchema foi a primeira linguagem de esquema para XML a obter o status de "recomendação" por parte do W3C. Esta linguagem é uma alternativa ao DTD (Document Type Definition), cuja sintaxe não é baseada no formato XML. Um arquivo contendo as definições na linguagem XML Schema é chamado de XSD (XML Schema Definition). O termo XSD é muitas vezes utilizado como referência à linguagem XML Schema. Quando um documento XML é bem formado, ele pode ser validado com base em regras que podem ser definidas utilizando à linguagem XML Schema [XSD, 2008].

WSDL (Web Services Definition Language) é uma linguagem baseada em XML utilizada para descrever Web Services. Trata-se de um documento escrito em XML que além de descrever o serviço, especifica como acessá-lo e quais as operações ou métodos disponíveis. Foi submetido ao W3C por Ariba, IBM e Microsoft em março de 2001 sendo que seu primeiro rascunho foi disponibilizado em julho de 2002. Atualmente encontra-se na sua versão 2.0 [WSDL, 2008].

XSLT (XML Stylesheet Language Transformations) é uma linguagem de marcação XML usada para transformar documentos XML. É parte de linguagem de transformação XML da especificação XSL (as outras partes são XSL-FO e XPath). Como a XML e a HTML, a especificação XSLT é uma recomendação desenvolvida pela W3C. A especificação XSLT possibilita transformações mais potentes do que as folhas de estilo CSS. Porém, a apresentação de documentos XML é apenas um dos objetivos de XSLT, podendo ser usada também para transformar um documento de um vocabulário diferente para outro vocabulário [XSLT, 2008].

XQuery é uma linguagem de consulta, com alguns recursos de programação, que é projetada para fazer consultas em coleções de dados em XML. A linguagem XQuery é semanticamente similar ao SQL, o que a torna facilmente inteligível para pessoas que

[1] O conjunto de documentações pode ser acessado no endereço http://edocs.bea.com.

trabalham com consultas a banco de dados relacionais. Além de ser uma linguagem de consulta, a linguagem XQuery é utilizada como linguagem para transformação de documentos XML, uma vez que é possível definir o formato de resposta da consulta [XQUERY, 2008].

SOAP (originado do acrônimo Simple Object Access Protocol) é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída, utilizando tecnologias baseadas em XML. Sua especificação define um *framework* que provê maneiras de construir mensagens que podem trafegar através de diversos protocolos e que foi especificado de forma a ser independente de qualquer modelo de programação ou outra implementação específica. Por não se tratar de um protocolo de acesso a objetos, o acrônimo não é mais utilizado. Geralmente servidores SOAP são implementados utilizando-se servidores HTTP, embora isto não seja uma restrição para funcionamento do protocolo. As mensagens SOAP são documentos XML que aderem a uma especificação fornecida pelo órgão W3C [SOAP, 2008].

WS-BPEL (Web Services Business Process Execution Language) é uma linguagem executável para especificar interações com serviços web. A especificação é utilizada para definir a composição de serviços web. Uma composição descrita em WS-BPEL é exposta como um serviço web e possui um documento WSDL que a descreve [BPEL, 2008].

WS-Addressing (Web Services Addressing) é uma especificação para representar endereçamento de mensagens de forma neutra de mecanismos de transporte específicos. A especificação permite que a comunicação entre serviços web use uma referência para o serviço web de destino, ao invés de uma URL física. A especificação consiste de duas partes: uma estrutura para comunicar uma referência de um serviço web e um conjunto de propriedades da mensagem as quais associadam informação de endereçamento a uma mensagem particular [WSA, 2008].

WS-Security (Web Services Security) é um protocolo de comunicação que provê um vocabulário para aplicar regras de segurança a serviços web. Atualmente o padrão é regido pela OASIS e é conhecido como WSS. O protoclo contém especificações para princípios de segurança. Entre eles, estão vocabulários para definir confidencialidade e integridade às mensagens SOAP, adicionar assinaturas digitais, tokens de segurança como certificados X.509 e bilhetes Kerberos [WSS, 2008].

WS-Policy é uma especificação que permite que serviços web utilizem um vocabulário XML para expor suas políticas (de segurança, qualidade de serviço, etc) e que consumidores de serviços especifiquem seus requisitos. WS-Policy atualmente é uma recomendação da W3C desde setembro de 2007. A especificação WS-Policy representa um conjunto de especificações que descrevem capacidades e restrições de políticas de segurança (e outras políticas de negócio) e como associar essas políticas a serviços [WSP, 2008].

SAML (Security Assertion Markup Language) é um padrão baseado em XML para troca de dados de autenticação e autorização entre domínios de segurança, ou seja, entre um provedor de identidades e um provedor de serviços. SAML é uma especificação mantida pela OASIS Security Services Technical Committee. O principal problema que a especificação SAML tenta resolver é o problema de identificação única em ambiente web (Single Sign-On ou SSO) [SAML, 2008].

1.2 Estrutura do relatório

Esse relatório está organizado em 4 capítulos, sendo o capítulo 1 a Introdução.

No capítulo 2 são apresentadas análises das ferramentas estudadas.

Nos capítulos 3 e 4 são apresentadas nossas conclusões sobre as ferramentas e as referências bibliográficas, respectivamente.

Por fim, os anexos apresentam os testes realizados e informações mais detalhadas das ferramentas.

2 Análise das ferramentas da BEA

Neste capítulo descrevemos as funcionalidades relevantes das ferramentas avaliadas da BEA. A definição das ferramentas escolhidas para avaliação se deu pela definição das funcionalidades mais impactantes para adoção de uma iniciativa SOA que necessitam de um suporte computacional adequado. Dentre essas funcionalidades, foram consideradas a disponibilização dos serviços e o monitoramento da troca de mensagens entre esses serviços, o controle e catalogação dos serviços existentes, e a possibilidade de disponibilizar serviços compostos. Baseando-se nestas funcionalidades, foram selecionadas as ferramentas AquaLogic Service Bus, AquaLogic Service Registry e AquaLogic BPM.

2.1 AquaLogic Service Bus

O AquaLogic Services Bus [ALSB, 2008a] (também conhecido como ALSB) funciona em conjunto com o WebLogic Server (também conhecido como WLS), que é o servidor de aplicações da BEA. Assim, o ALSB pode ser considerado um módulo do WLS. O barramento funciona como um ponto de acesso comum para chamadas de serviços web existentes em servidores distintos. Os serviços são organizados no barramento como projetos. Cada projeto pode conter tanto descrições para serviços (WSDL) quanto arquivos criados para completar o projeto (transformações XQuery ou XSLT, definições de tipos em XML Schema, etc), vide Figura 23. Além dos tipos de arquivos comuns de um ambiente SOA (arquivos XMLs como os listados anteriormente), o barramento permite acrescentar arquivos MFL ao projeto. MFL (Message Format Language) [MFL, 2008] é um documento XML usado para descrever o leiaute de dados binários. Este vocabulário XML é proprietário da BEA e é usado para definir regras para transformar dados binários para documentos em formato XML.

Um documento WSDL descreve um ponto de acesso (endpoint) físico do serviço descrito. O barramento ALSB permite definir outros pontos de acesso para o serviço e definir o algoritmo de balanceamento de carga para esses pontos de acesso. Dessa forma, o barramento fica sendo responsável por encaminhar as mensagens recebidas para um dos destinos do serviço para processamento da mensagem. Caso um dos servidores que abriga o serviço esteja sobrecarregado ou fora de condições de uso, o barramento reencaminha a mensagem para outro servidor que possui a réplica do serviço. Entre as opções para balanceamento de carga, estão disponíveis algoritmos round-robin, random ou random-weighted.

Essa troca de mensagens entre o barramento e os serviços pode ser monitorada. Cada serviço disponibilizado no barramento é, por padrão, não monitorado. Assim, o monitoramento deve ser ativado para cada serviço que se deseja através da opção

Monitoring ➔ Enabled. É possível alterar o intervalo de agregação do monitoramento. Esse intervalo corresponde ao intervalo em que as estatísticas do serviço serão agregadas no barramento. Depois de ativado o monitoramento de algum serviço, o barramento começa a gerar um relatório sobre a quantidade de mensagens enviadas/recebidas, quantidade de mensagens de erro, servidores não ativos, tempo de envio de mensagens, etc. Relatórios podem ser gerados também ao definir SLAs (*Service Level Agreements*). Ou seja, o usuário pode definir regras para emissão de alertas e, assim, verificar taxas de erro, sucesso e falha, erros de segurança, tempo de resposta, etc. As mensagens de alerta são exibidas em um *dashboard*, conforme mostra a Figura 56.

Além da possibilidade de acrescentar novos pontos de acesso a serviços, fazendo com que seja transparente o roteamento da mensagem recebida, o barramento permite ainda criar roteamentos inteligentes baseados em conteúdo de mensagens. Esse tipo de roteamento, chamado de Serviço de Proxy, permite que aplicações simples de fluxo de mensagens sejam especificadas. Com o roteamento baseado em conteúdo, o barramento permite analisar o conteúdo de uma mensagem SOAP recebida, determinar seu caminho dependendo do valor contido na mensagem e realizar ações na mensagem antes de enviá-la para seu destino, como transformações, remoções ou adições de conteúdo. Essas transformações podem ser feitas utilizando os padrões XSLT e XQuery.

O roteamento de mensagens é uma forma simplificada de criar um fluxo de chamada de serviços, sendo uma alternativa ao padrão WS-BPEL em casos mais simples. É importante ressaltar que a interpretação de documentos XML com vocabulário WS-BPEL não é uma funcionalidade ainda disponibilizada por esse barramento.

A troca de mensagens no barramento pode ser feita utilizando os protocolos HTTP/SOAP e EJB/RMI nos formatos de mensagem WS-Addressing, SOAP versão 1.1 e 1.2 e WS-I[2]. O transporte dessas mensagens pode ser protegido com protocolo SSL ou conter definições WS-Security e WS-Policy para definição de regras de seguranças aplicáveis às mensagens trafegadas pelo barramento. Além disso, o barramento permite descrição de credenciais de usuários utilizando vocabulário SAML ou tecnologias PKI e OOTB [OOTB, 2008].

A troca de mensagens, principalmente após a definição de roteamentos, pode ser testada no próprio barramento através de uma interface de teste (vide Figura 51). Essa interface de teste permite analisar as mensagens de envio, as mensagens de resposta e, principalmente, verificar o caminho pelo qual a mensagem trafegou.

Como funcionalidade importante de gerência do barramento, todas as alterações realizadas no barramento são feitas dentro de uma sessão de alteração. Assim, antes de realizar uma ação no barramento, é necessário criar uma sessão e, ao terminar o trabalho, ativar a seção. A alteração realizada será implementada somente após a ativação da sessão. Uma vez que as sessões armazenam ações de alterações realizadas pelo administrador do barramento, é possível verificar o histórico de sessões. Cada sessão possui sua data, descrição, identificação do usuário que ativou a sessão e o conjunto de operações que foram realizadas na sessão. Caso as operações realizadas na

[2] WS-I (Web Service Interoperability) (WSI, 2008) é um consórcio que não define padrões, mas provê um conjunto de revisões de outras especificações, chamadas de Profile, que definem a forma com que as especificações devem ser usadas para criar serviços web interoperáveis.

sessão não tenham o efeito esperado, é possível desfazer a sessão (opção *undo*). Tal funcionalidade facilita o gerenciamento e rastreamento das intervenções que o barramento sofreu por usuários administradores.

Por fim, o barramento permite a sincronização dos serviços com um repositório de serviços utilizando o vocabulário UDDI. Uma vez que o vocabulário UDDI é um padrão, teoricamente, é possível utilizar qualquer servidor UDDI com o barramento ALSB. A BEA, por sua vez, disponibiliza um servidor UDDI chamado de AquaLogic Service Registry, descrito na seção 2.2.

2.2 AquaLogic Service Registry

Essa é uma aplicação que pode rodar como parte do WebLogic Server ou, caso não tenha o WLS, pode-se utilizar um servidor HTTP/HTTPS próprio do ALSR. Além disso, é necessário ter acesso a um banco de dados para armazenamento dos registros. O ALSR permite utilizar Oracle, HSQL, MSSQL, DB2, PostgreSQL ou Sybase ASE.

A ferramenta é compatível com o padrão UDDI versão 3 [UDDI, 2004], porém algumas partes da especificação não foram implementadas, como os requisitos de replicação, descrição de políticas e canonização XML para assinaturas digitais [CANONICALIZATION, 2002]. O mapeamento entre WSDL e UDDI é feito seguindo o padrão especificado pela OASIS [WSDL-UDDI, 2004], o qual contempla o novo vocabulário WSDL versão 2.0 [WSDL, 2008].

É possível utilizar vários repositórios ALSR com papéis distintos caso seja do interesse da corporação criar uma arquitetura de repositórios visando à aprovação dos dados publicados, numa arquitetura análoga à usada em desenvolvimento de versões de *softwares*, com servidores de desenvolvimento, homologação e produção. No caso dos repositórios UDDI, pode-se ter um repositório único, o caso mais simples, um repositório de publicação (análogo a um servidor de homologação) e um de pesquisa (análogo a um servidor de produção), como mostra as figuras Figura 60 e Figura 61. O caso mais completo seria a criação de três repositórios para a implantação de uma arquitetura que condizente com um processo de aprovação em duas fases (Figura 62).

Os usuários (humanos ou *softwares*) que interagem com o repositório devem se autenticar. Para autenticação, a ferramenta permite que os dados de autenticação sejam armazenados no mesmo banco de dados utilizado para armazenar os dados dos serviços web, permite utilizar um servidor LDAP ou um repositório externo que será acessado através de uma API (Figura 65).

O ALSR é um repositório que permite armazenar não somente a definição de serviços web, isto é, arquivos WSDL, mas também os artefatos necessários para a comunicação com esses serviços, como arquivos XSLT, tipos definidos em XML Schemas e outros arquivos em formato XML. Dessa forma, o ALSR pode ser usado não só como um repositório de serviços, mas também como um repositório de tipos de dados sendo, este último, uma funcionalidade importante para a gerência dos dados que serão transportados entre os serviços.

O interessante de um repositório de serviços que siga a especificação UDDI é que o repositório permite também associar elementos técnicos, por exemplo, serviço com elementos de negócio como registros de departamentos da empresa e funcionários responsáveis. Assim, é possível consultar todos os serviços que são de responsabilidade de uma dada área da empresa, por exemplo. Além dessa visão gerencial dos serviços, é possível armazenar e consultar os dados mais técnicos do

serviço como seus pontos de acesso, seus tipos de dados e informações estruturais do WSDL como bindings, ports etc.

Por fim, o ALSR permite gerenciar vários aspectos do repositório (vide Figura 74), tendo como opções principais: o acesso aos usuários e criar grupos de usuários com permissões distintas, gerenciar a taxonomia aplicada aos serviços e gerenciar a replicação do repositório, que pode ser usada por medida de segurança ou para balancear o acesso ao servidor, evitando a sobrecarga.

2.3 AquaLogic BPM

O AquaLogic BPM (também conhecido como ALBPM) é a solução BPMS da BEA [ALBPM, 2008a]. BPMS (acrônimo para BPM *System* ou BPM *Suíte*) é um ambiente para execução de processos de negócio que contempla desde a concepção do processo de negócio até sua execução e monitoramento, implementado sobre a plataforma Eclipse [ECLIPSE, 2008]. Além dessas funcionalidades, ambientes BPMS permitem também realizar simulações dos processos, visando à melhoria dos mesmos. O ALBPM permite modelagem dos processos em uma notação proprietária e a utilização de uma máquina para execução destes processos.

A ferramenta permite a importação de processos no formato XPDL. O XPDL (*eXtended Process Definition Language*) [XPDL, 2008] é um vocabulário XML que representa a estrutura de leiaute de um fluxograma. O formato XPDL é padronizado pelo Workflow Management Coalition (WfMC) para intercambiar diferentes formatos de *workflow*. A Figura 1 apresenta um exemplo da tela de visualização e edição de processos no ALBPM.

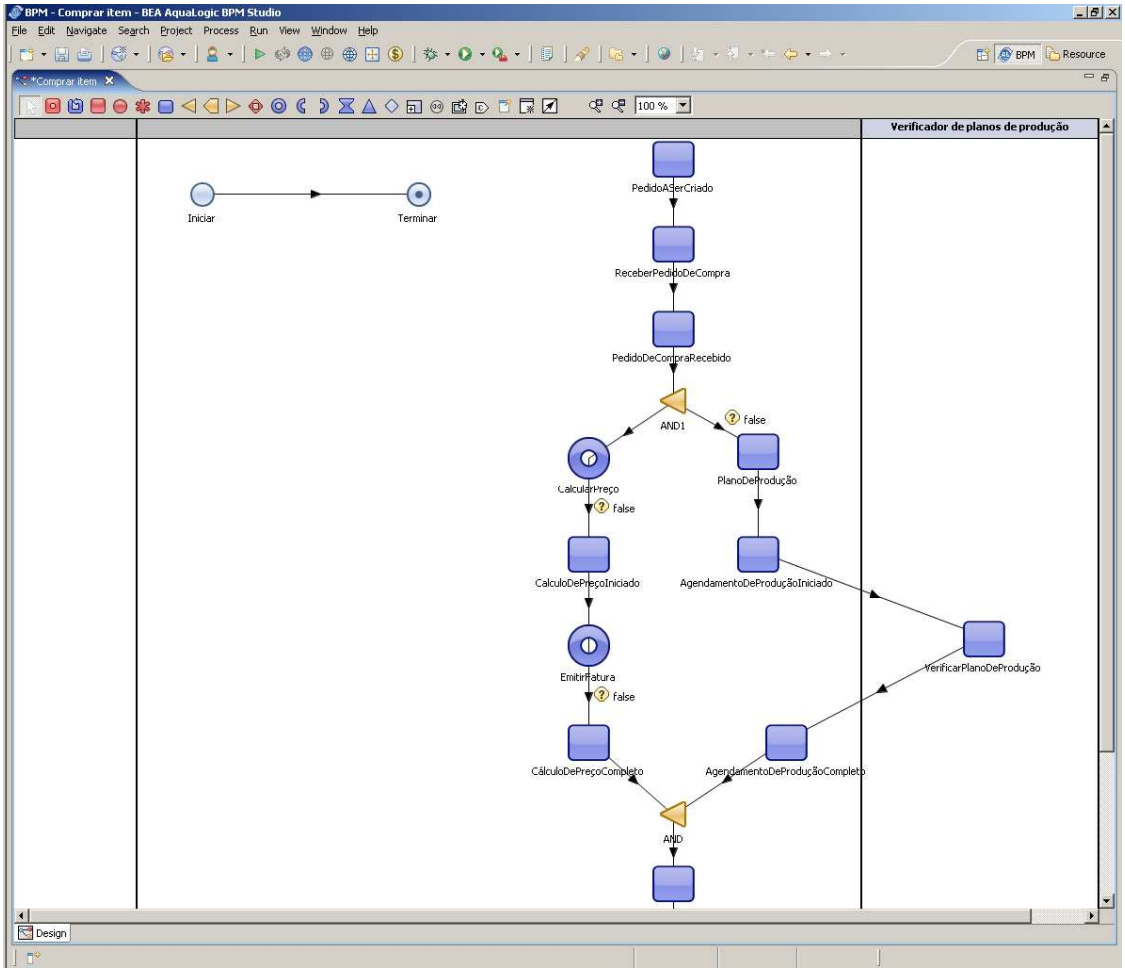

Figura 1 – Processo ARIS importado no ALBPM via vocabulário XPDL

As versões mais recentes do ALBPM permitem a criação de processos BPEL e sua execução. A execução dos processos BPEL é feita pelo AquaLogic BPM Enterprise Server. O *ALBPM* Enterprise Server pode ser executado no WLS ou mesmo em uma máquina Java (JVM) *stand-alone*. Se o ALBPM Enterprise Server estiver executando em um servidor de aplicações, será possível utilizar todos os recursos oferecidos pelo servidor de aplicação, tais como *clustering*, *pool* de conexões, balanceamento de carga, tratamento automático de falhas, etc.

Segundo documentação da BEA [ALBPM, 2008b], existem três diferentes edições distintas do ALBPM: uma versão *standalone*, uma versão para o WLS e uma versão para o IBM WebSphere Application Server. A versão *standalone*, já mencionada anteriormente, permite executar o Process Execution Engine como uma aplicação Java fora de qualquer servidor de aplicação. Esta edição utiliza um servidor de aplicação apenas para as interfaces HTML para usuários (este servidor de aplicação é uma versão do TomCat incluído durante a instalação). A versão para o WLS permite rodar o Process Execution Engine em cima do BEA WebLogic Server e, assim, permite utilizar as capacidades do container JEE como gerência de transações, *clustering* e administração centralizada. A versão para o IBM WebSphere Application Server é similar à versão para WLS, porém sendo executada no servidor de aplicação da IBM.

A criação de processos BPEL no ALBPM é feita utilizando um diagrama específico chamado BPEL Process. Este diagrama não está disponível após a instalação da suíte. Para habilitá-lo, é necessário ativar o perfil Developer, ir à opção Preferências,

selecionar BPEL, ativar a caixa “BPEL Support” e clicar em OK. Após reiniciar o editor do ALBPM, chamado de AquaLogic BPM Studio, será possível criar diagramas BPEL. Os diagramas BPEL no ALBPM Studio são editados utilizando um plugin do Eclipse para edição dos processos BPEL. Informações sobre o plugin podem ser encontradas na página oficial, em [BPEL-PLUGIN, 2008]. A Figura 2 apresenta um exemplo da tela de edição de processos BPEL da ferramenta ALBPM.

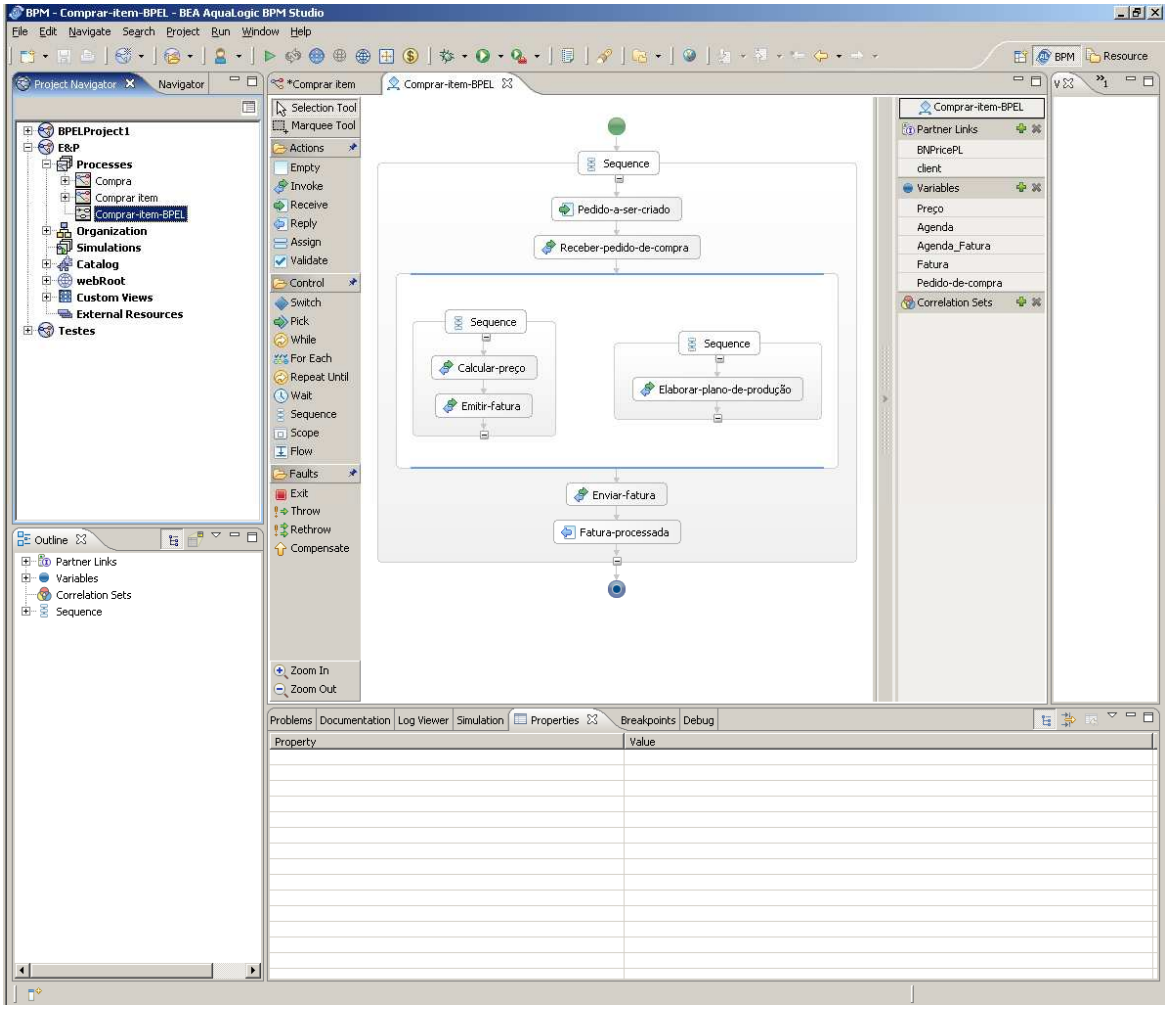

Figura 2 – Processo BPEL criado na ferramenta ALBPM

Os processos BPEL no ALBPM são encapsulados em um projeto na ferramenta ALBPM Studio. Este projeto conterá o(s) processo(s) BPEL, os arquivos WSDL, bibliotecas Java que forem necessárias, arquivos XSD e quaisquer outros recursos necessários para o projeto. Para disponibilizá-los em um servidor, o projeto pode ser exportado do ALBPM no formato EAR típico de aplicações Java e importado para o servidor de aplicação que possui o Business Process Engine.

Por fim, como o ALBPM é uma ferramenta implementada sobre a plataforma Eclipse, ele herda suas potencialidades, como a inclusão de *plugins* para Eclipse e as opções nativas da plataforma, como as opções de uso de repositório de versões SVN ou CSV para armazenar versões do projeto, conforme mostra Figura 3.

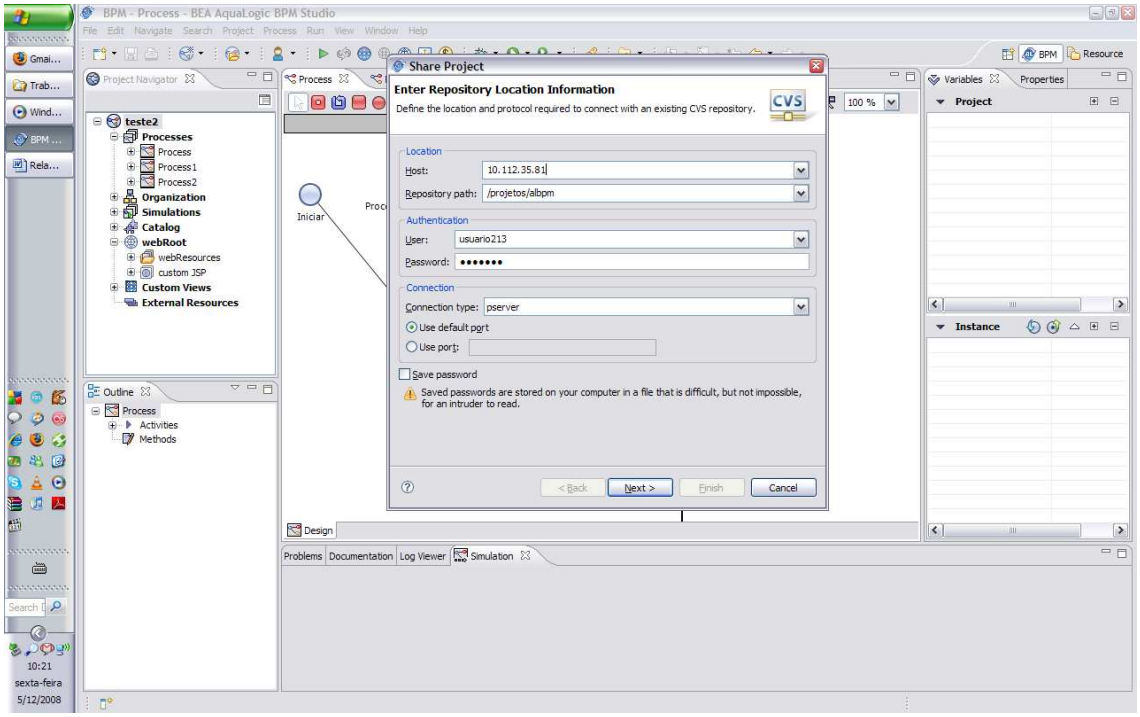

Figura 3 – Cadastrando repositório de versões no ALBPM

O versionamento do projeto inclui também o versionamento do modelo. Contudo, esse versionamento é baseado no conteúdo alfanumérico dos documentos XML que representam o modelo, o que permite o controle de alterações do modelo, mas não distingue versões com alterações de estrutura do modelo de versões com alterações puramente visuais. Diversos *plugins*, principalmente para desenvolvimento, podem ser encontrados na página da ferramenta [ECLIPSE, 2008] e outros sites especializados[3] em extensões para a plataforma Eclipse.

3 Conclusões

Este relatório apresentou a análise feita em três ferramentas da fornecedora BEA: AquaLogic BPM, AquaLogic ESB e AquaLogic Service Registry, as quais foram consideradas as ferramentas mais relevantes para implantação técnica de uma iniciativa SOA.

Uma das características mais importantes de ferramentas para adoção em iniciativas SOA é a sua adequação aos padrões definidos pela OASIS ou W3C, permitindo, assim, que a solução seja mais facilmente portável e independente de tecnologias proprietárias. O barramento da BEA, ALSB, suporta transformações em XQuery e XSLT, interpreta documentos WSDL, implementa o padrão WS-Addressing, SOAP 1.1 e 1.2, SAML, WS-Security, WS-Policy e está de acordo com as boas práticas definidas pelo WS-I. A ferramenta ALSR implementa o padrão UDDI v3, utiliza o padrão XML Canonicalization para canonizar assinaturas digitais em XML, além de seguir o padrão especificado pela W3C para mapeamento da estrutura WSDL em UDDI. Foi identificado que a ferramenta ALBPM suporta o padrão XPDL (que não é um padrão da OASIS ou W3C, mas que é importante para a transferência de modelos

[3] Acesse, por exemplo, <www.eclipseplugincentral.com/> ou <www.eclipse-plugins.info/>.

de processo de negócio) e os padrões necessários para desenvolvimento de serviços orquestrados: BPEL, WSDL, XML Schema, XSLT etc.

Agradecimentos

Este trabalho não seria possível sem a contribuição de pesquisadores em Sistemas de Informação e da parceria com a Petrobras, principalmente a área TIC/TIC-E&P/GDIEP. Em especial, agradecemos aos professores e alunos que colaboraram nas discussões e desenvolvimento de pesquisas, testes e desenvolvimentos necessários ao projeto. Dentre os agradecimentos à academia, se destaca o papel dos profissionais do NP2Tec[4] que contribuíram, técnica ou administrativamente, para o sucesso de nossas atividades.

A condução e os resultados deste trabalho são uma exemplar evidência de como a relação entre as universidades e as empresas pode contribuir para a geração de conhecimento útil e, desta forma, contribuir para nossa sociedade.

Referências Bibliográficas

ALBPM, Documentação sobre a ferramenta BEA AquaLogic BPM Entreprise, disponível em <http://edocs.bea.com/albsi/docs60/>, 2008a.

ALBPM, Documentação sobre edições do ALBPM, disponível em <http://edocs.bea.com/albsi/docs60/config_wls/index.html>, 2008b.

ALSB, Documentação sobre conceitos e arquitetura da ferramenta AquaLogic Service Bus, disponível em <http://edocs.bea.com/alsb/docs30/concepts/index.html>, 2008a.

ALSB, Documentação de instalação da ferramenta AquaLogic Service Bus, disponível em <http://edocs.bea.com/alsb/docs30/install/index.html>, 2008b.

ALSB, Tutorial para iniciantes da ferramenta AquaLogic Service Bus, disponível em <http://e-docs.bea.com/alsb/docs30/tutorial/tutGettingStarted.html>, 2008c.

ALSR, Documentação de instalação da ferramenta AquaLogic Service Registry, disponível em <http://edocs.bea.com/alsr/docs30/install/index.html>, 2008.

BPEL, Business Process Execution Language, disponível em <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel>, 2008.

BPEL-PLUGIN, Informações sobre o plugin do Eclipse para criação de diagramas BPEL, disponível em <http://www.eclipse.org/bpel/users.php>, 2008.

CANONICALIZATION, Exclusive XML Canonicalization Version 1.0, disponível em <http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/>, 2002.

ECLIPSE, What is Eclipse and the Eclipse Foundation?, disponível em <http://www.eclipse.org/org/#about>, 2008.

MFL, Message Format Language, disponível em <http://e-docs.bea.com/alsb/docs20/consolehelp/mfls.html>, 2008.

OOTB, Security Test and Evaluation Labs: BEA WebLogic Server 7.0, disponível em

[4] Site do NP2Tec: http://www.uniriotec.br/~np2tec

<http://www.cygnacom.com/labs/pfSEL0142b.htm>, 2008.

SAML, Security Assertion Markup Language Specification, disponível em <http://saml.xml.org/saml-specifications/>, 2008.

SOAP, SOAP Specification, disponível em <http://www.w3.org/TR/soap/>, 2008.

UDDI, Universal Description, Discovery & Integration Specification version 3, disponível em <http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3>, 2004.

WSA, Web Service Addressing Specification, disponível em <http://www-128.ibm.com/developerworks/library/specification/ws-add/>, 2008.

WSDL, Especificação da Web Service Description Language, disponível em <http://www.w3.org/TR/wsdl>, 2008.

WSDL-UDDI, Using WSDL in a UDDI Registry, version 2.0.2, Technical Note, disponível em <http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm>, 2004.

WSI, Web Service Interoperability Specifications, disponível em <http://www.ws-i.org/>, 2008.

WSP, Web Service Policy Specifications, disponível em <http://www.w3.org/TR/ws-policy/>, 2008.

WSS, Web Service Security Specifications, disponível em <http://www-128.ibm.com/developerworks/library/specification/ws-secure/>, 2008.

XPDL, eXtended Process Definition Language version 2.1 Complete Specification, disponível em <http://www.wfmc.org/xpdl-developers-center.html>, 2008.

XQUERY, XML Query (XQuery), disponível em <http://www.w3.org/XML/Query>, 2008.

XSD, XML Schema, disponível em <http://www.w3.org/XML/Schema>, 2008.

XSLT, XML Stylesheet Language Transformations, disponível em <http://www.w3.org/TR/xslt20/>, 2008.

Anexo 1 - AquaLogic Service Bus

A.1.1 Instalação do ALSB

O AquaLogic Services Bus [ALSB, 2008a] (também conhecido como ALSB) funciona em conjunto com o WebLogic Server (também conhecido como WLS), que é o servidor de aplicações da BEA. Assim, o ALSB pode ser considerado um módulo do WLS. A tela de administração do ALSB e WLS, contudo, são distintas. O endereço default para acesso à tela de administração do ALSB e do WLS é http://localhost:7001/sbconsole/ e http://localhost:7001/console, respectivamente. O usuário/senha inicial para acesso ao ambiente de administração é weblogic/weblogic.

A.1.2 Publicação de serviços no barramento

Os exemplos utilizados nesse capítulo foram baseados no tutorial fornecido pela BEA [ALSB, 2008c].

A publicação de serviços no barramento compreende duas etapas. Em primeiro lugar, é necessário realizar o *deploy* das aplicações no servidor de aplicação WLS e, em seguida, publicar as interfaces de serviços (WSDL) no ALSB. O primeiro passo é necessário, uma vez que o serviço será executado no mesmo WLS que está a sua interface. Para uma aplicação distribuída, contudo, torna-se necessário salientar que o *deploy* do serviço (seu código executável) será realizado em qualquer outro servidor de aplicação (JBoss, WebSphere, etc).

A.1.2.1 Descrição do exemplo

Utilizaremos como exemplo um processo de empréstimo. O exemplo é baseado em uma agência financiadora (Mortgage Broker) que usa o ALSB para rotear pedidos de empréstimos aos serviços de negócio apropriados. Os pedidos de empréstimos são roteados para diferentes serviços de negócio depedendo de critérios de qualificação, como a taxa de juros e o principal (montante em dinheiro) requisitado. Quando o principal é maior que U$25 milhões, a informação da notação de crédito do requerente é requerida para completar o empréstimo.

O exemplo possui noções de mediação de serviços, roteamento de mensagens, transformação e validação.

A.1.2.2 Deployment dos serviços

Os serviços serão carregados no servidor de aplicação WLS. Para tal, temos já implementados os seguintes pacotes: creditLoan_jws_basic_ejb.jar, largeLoan_jws_basic_ejb.jar, manager_jws_basic_ejb.jar e normalLoan_jws_basic_ejb.jar. Cada pacote .jar corresponde a uma implementação de serviço que será carregada no servidor de aplicação. Vamos apresentar como carregar um pacote no WLS e o mesmo se repetirá para os restantes.

Para realizar o *deployment*, é necessário entrar no console de administração do WLS, a qual disponibilizará a tela, apresentada na Figura 4, no navegador.

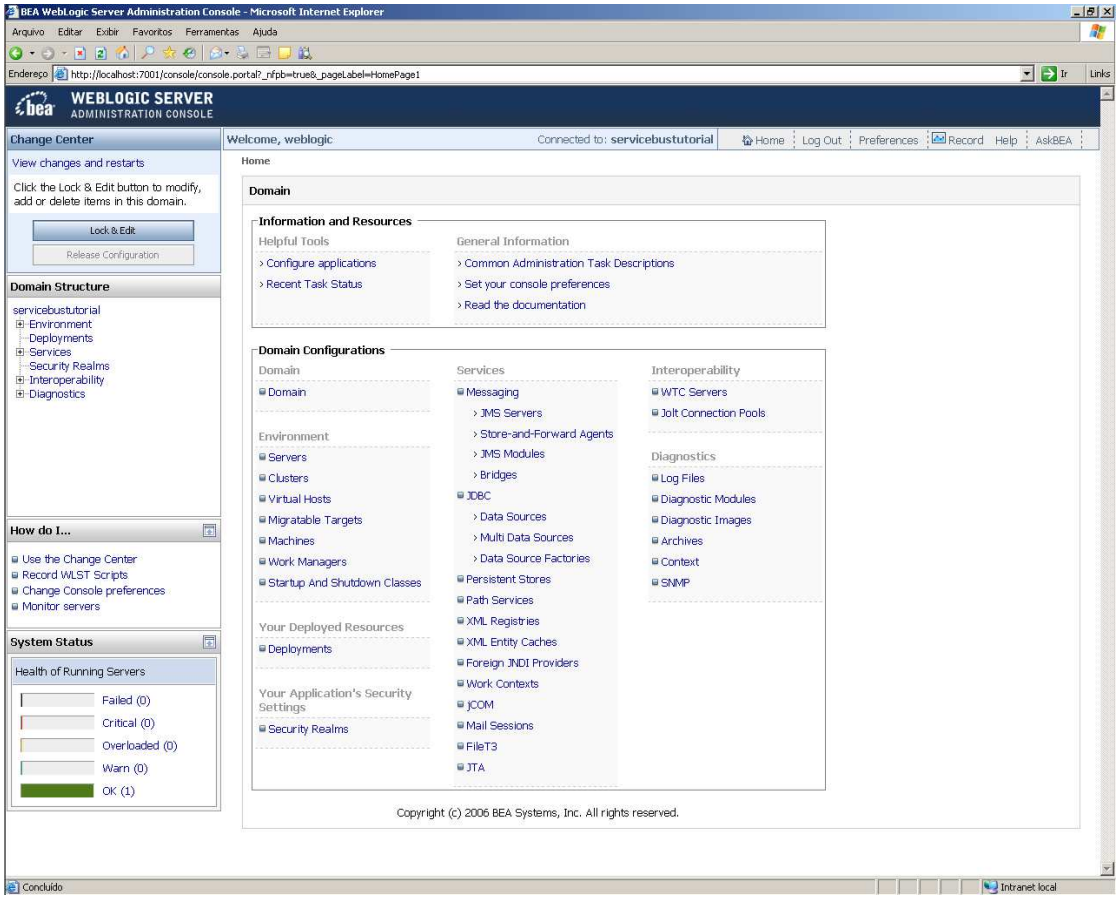

Figura 4 – Tela principal do WLS

Vamos clicar na opção Lock&Edit do Change Center para iniciar uma nova seção de mudanças no servidor. Em seguida, clicaremos na opção Deployments do Domain Structure, a qual exibirá a tela apresentada na Figura 5. Nesta tela, conseguimos visualizar todas as aplicações que estão sendo fornecidas pelo servidor. Para incluirmos os serviços de empréstimo, clicaremos na opção Install e usaremos o Install Application Assistant.

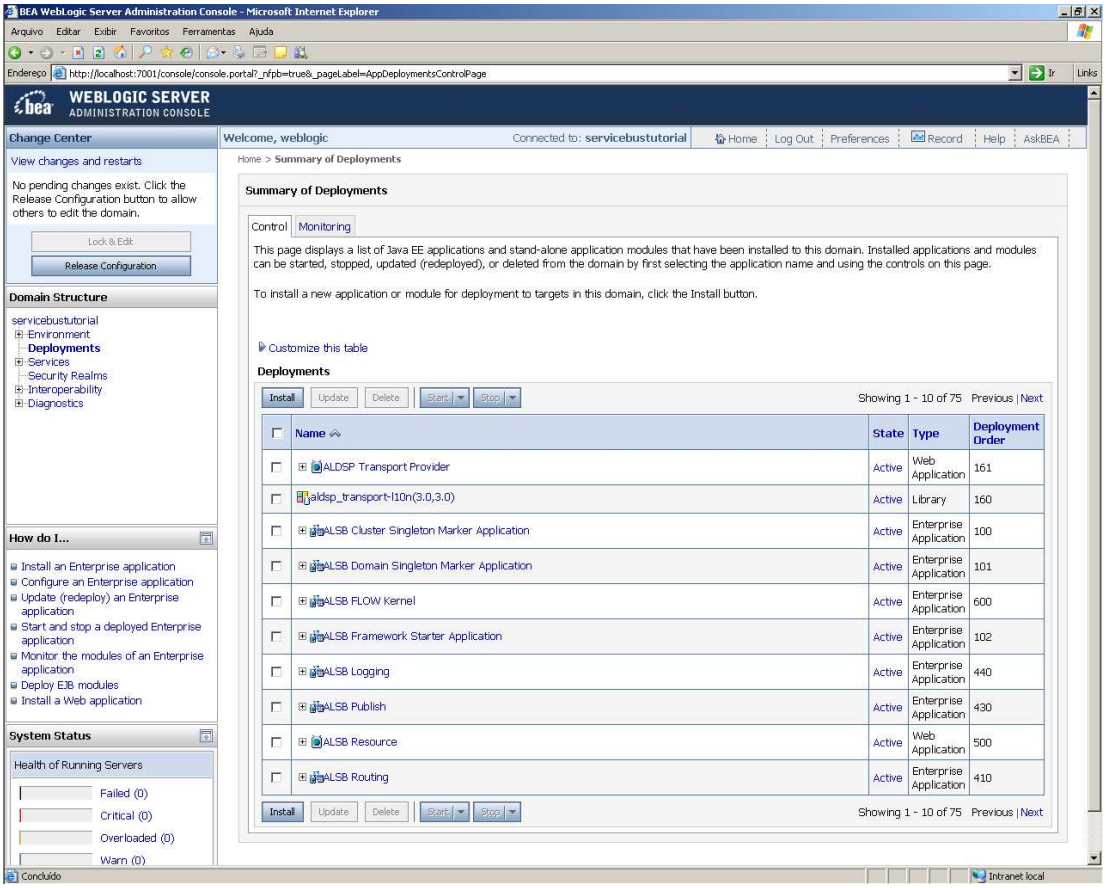

Figura 5 – Aplicações disponíveis no WLS

Caminharemos pelas pastas até encontrar os arquivos .jar e marcaremos o primeiro deles para iniciar o deploy (Figura 6). Assim como outros servidores de aplicação, o *deploy* de aplicações é feito sequencialmente, ou seja, é necessário que escolhamos os pacotes um a um. No momento, escolheremos o arquivo creditLoan_jws_basic_ejb.jar (Figura 7).

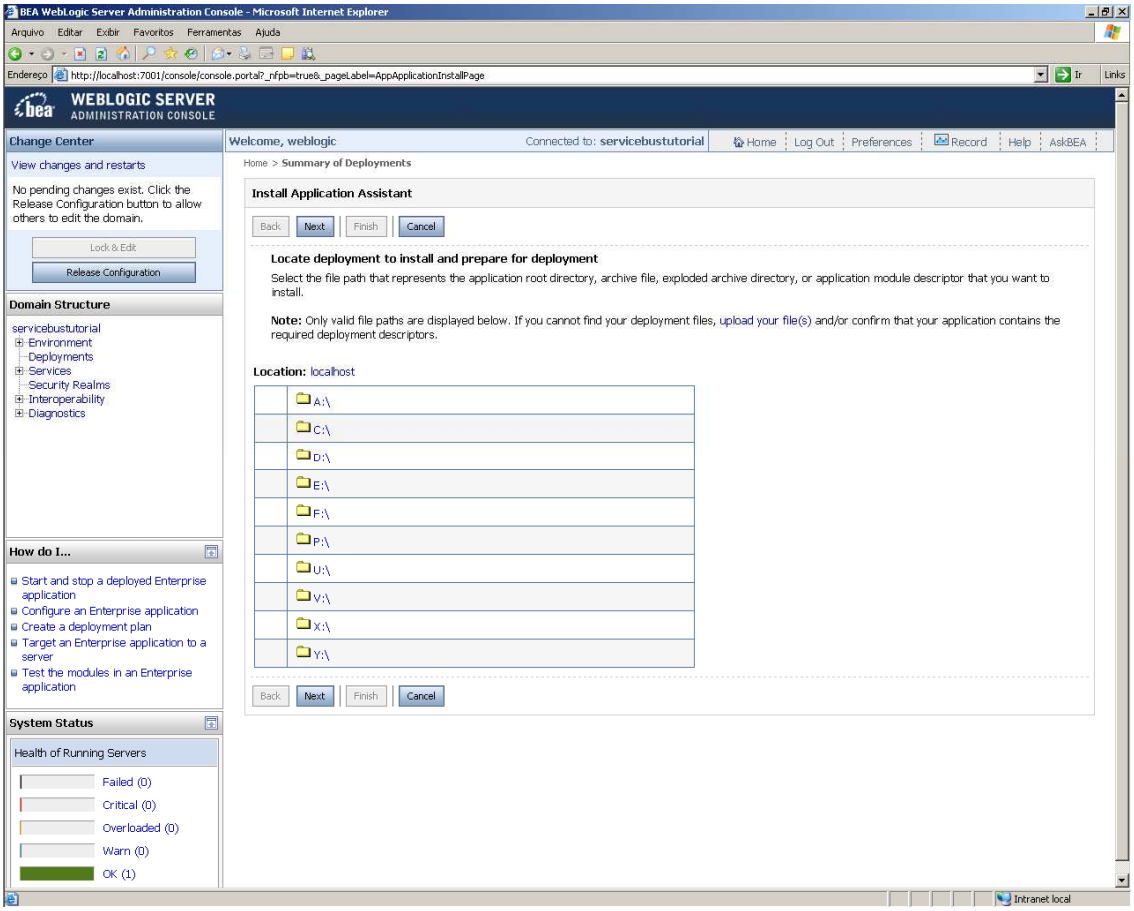

Figura 6 – Disponibilizando nova aplicação

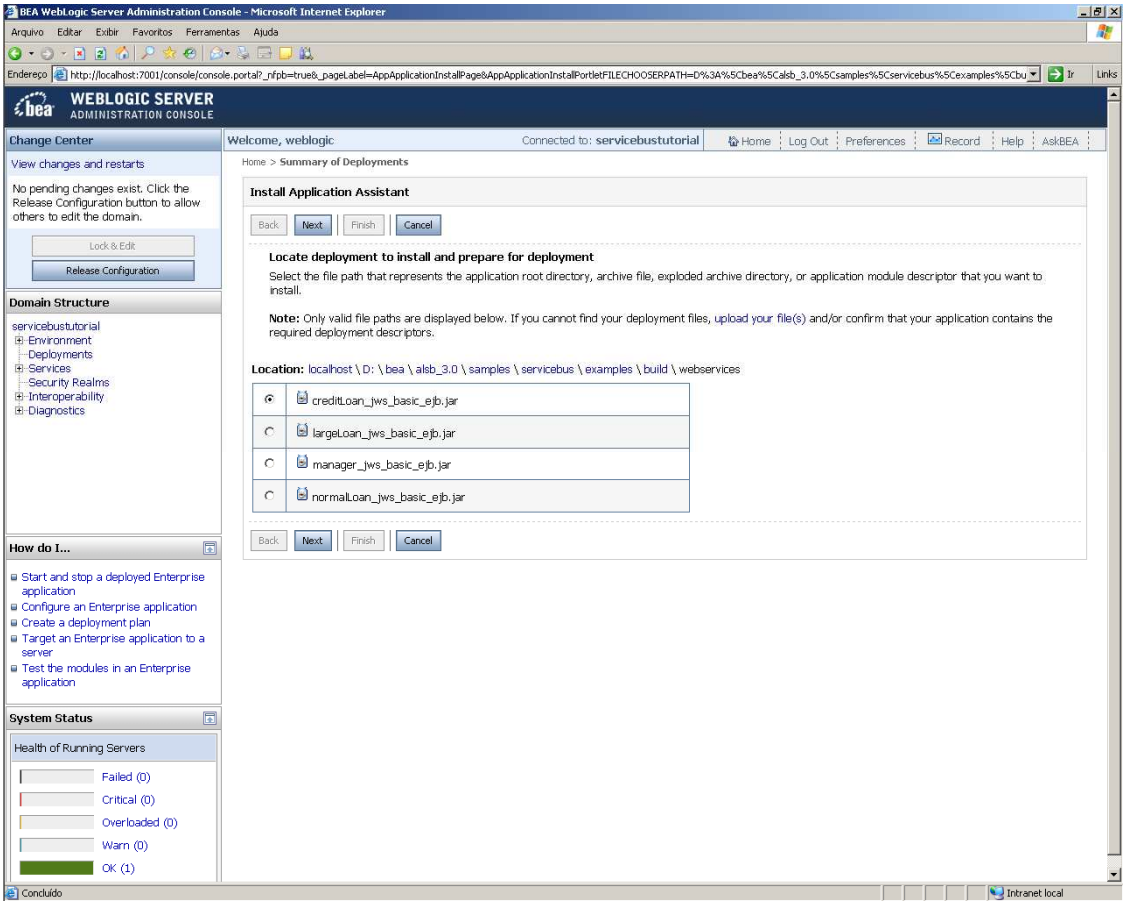

Figura 7 – Inicializando aplicação no servidor

Em seguida, podemos escolher se o pacote será registrado como uma aplicação ou como uma biblioteca do servidor (Figura 8). Uma aplicação pode ser acessada por clientes (internos ou externos) e uma biblioteca é um conjunto de funcionalidades que serão acessadas por aplicações dentro do próprio servidor.

Figura 8 – Tipos de deploy de uma aplicação

Após escolhermos registrar o pacote como uma aplicação, somos direcionados para alterarmos as configurações opcionais, como mostra Figura 9.

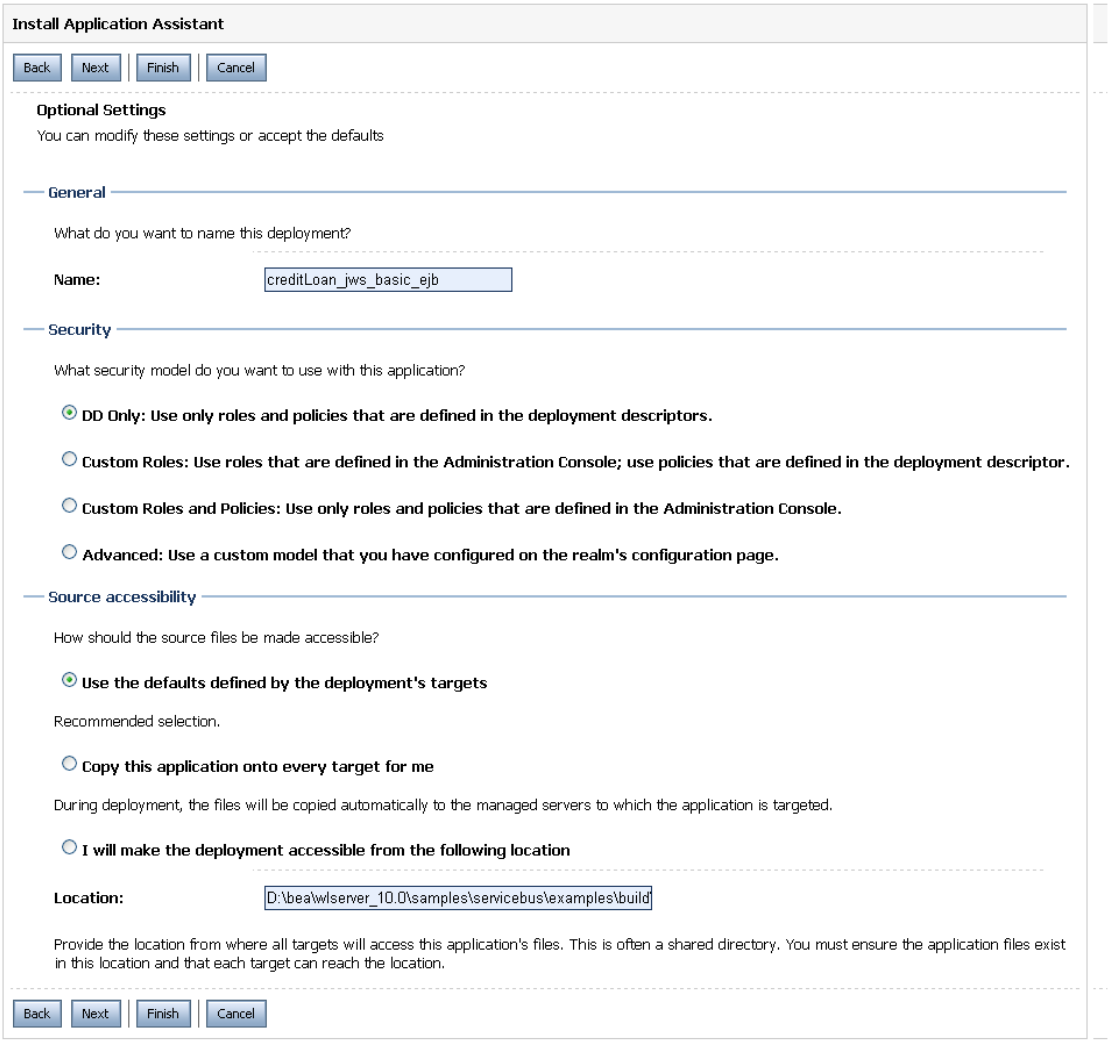

Figura 9 – Configurações opcionais de uma aplicação

Não alteraremos as opções, somente clicaremos em Finish para iniciar o deploy do serviço. O deploy será iniciado e pode ser acompanhado pelo console do servidor (Figura 10).

```
<26/09/2008 10h50min30s BRT> <Notice> <Server> <BEA-002613>
<Channel "Default[1]
" is now listening on 127.0.0.1:7001 for protocols iiop, t3,
ldap, snmp, http.>
<26/09/2008 10h50min30s BRT> <Notice> <WebLogicServer> <BEA-
000331> <Started Web
Logic Admin Server "AdminServer" for domain
"servicebustutorial" running in Deve
lopment Mode>
<26/09/2008 10h50min30s BRT> <Warning> <Server> <BEA-002611>
<Hostname "localhos
t", maps to multiple IP addresses: 10.25.138.63, 127.0.0.1>
<26/09/2008 10h50min32s BRT> <Notice> <WebLogicServer> <BEA-
000365> <Server stat
e changed to RUNNING>
<26/09/2008 10h50min32s BRT> <Notice> <WebLogicServer> <BEA-
```

```
000360> <Server star
ted in RUNNING mode>
<26/09/2008 11h03min22s BRT> <Warning> <netuix> <BEA-423420>
<Redirect is execut
ed in begin or refresh action. Redirect url is
/console/console.portal?_nfpb=tru
e&_pageLabel=EJBApplicationOverviewPage&EJBApplicationOverviewP
ortlethandle=com.
bea.console.handles.AppDeploymentHandle%28%22com.bea%3AName%3Dc
reditLoan_jws_bas
ic_ejb%2CType%3DAppDeployment%22%29.>
```

Figura 10 – Arquivo de Log

No WLS, após uma aplicação ser cadastrada no servidor, ela encontra-se no estado "Preparada para inicialização" e necessita que sua inicialização seja aprovada pelo administrador. Para tal, iremos na tela de Deployment novamente e procuraremos o pacote que acabamos de inserir (Figura 11).

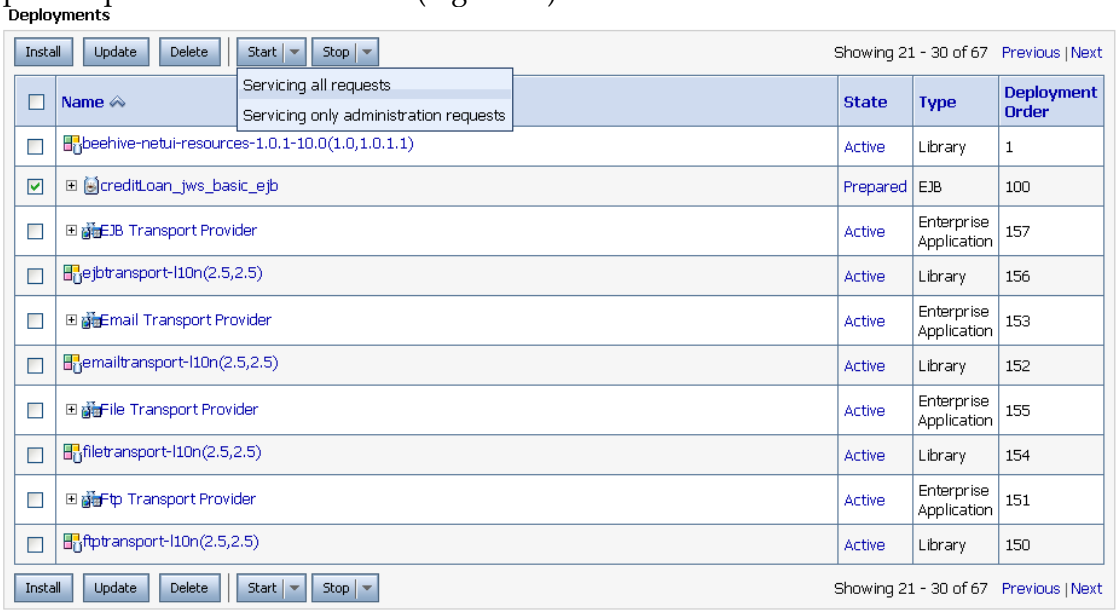

Figura 11 – Aprovando a incialização da aplicação

Para iniciar a aplicação, escolhemos o serviço e selecionamos a opção Servicing all requests no menu Start. Repare que, incialmente, o estado da aplicação creditLoan_jws_basic_ejb está como Prepared. Após clicarmos em Servicing all requests, a tela apresentada na Figura 12 é disponibilizada para que aceitemos a mudança da aplicação do estado Prepared para Active.

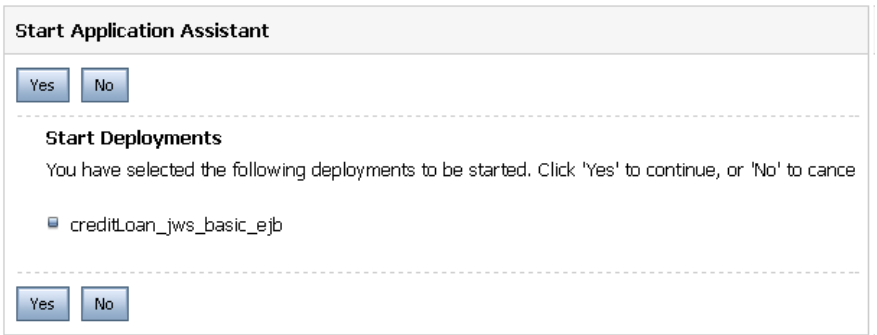

Figura 12 – Aceitando a inicialização de uma aplicação

O mesmo processo acima deve ser realizado para todas as outras 3 aplicações que serão utilizadas nesse exemplo.

A.1.2.3 Publicação de aplicações no ALSB

Até esse ponto, temos um servidor de aplicação com algumas aplicações disponíveis. Agora iremos disponibilizar essas aplicações como serviços web, publicando suas interfaces no ESB. Adicionalmente, mostraremos outros recursos do AquaLogic Service Bus.

No cenário da aplicação, a companhia de financiamento usa o ALSB para rotear os pedidos de empréstimo para os serviços de negócio[5] apropriados baseado na taxa de juros requisitada. Uma requisição com uma taxa menor que 5% requer aprovação de um gerente e é roteada para um serviço para processamento. Todos os outros pedidos de empréstimo são roteadas para o serviço de negócio que não precisa de aprovação. A Figura 13 apresenta a arquitetura dos serviços expostos no barramento.

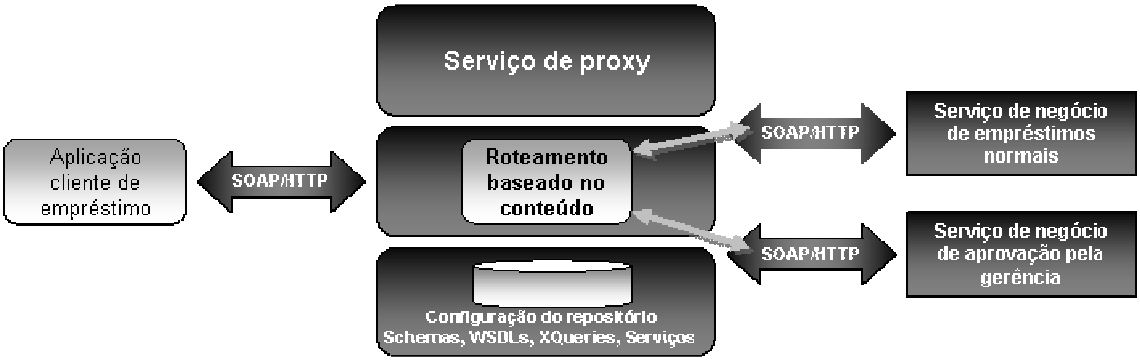

Figura 13 – Expondo a aplicação de empréstimo via ALSB

O ALSB funcionará, então, da seguinte forma: um cliente envia uma requisição de empréstimo para um serviço de Proxy chamado LoanGateway1. O serviço de Proxy tem um estágio de roteamento condicional que checa o valor da taxa de juros requisitada no empréstimo. Se a taxa de juros for menor que 5%, o empréstimo é roteado para o serviço chamado ManagerLoadReview; caso contrário, é roteado para o serviço chamado NormalLoan. Os dois serviços que podem processar a requisição retornam mensagens parecidas, como mostra Figura 14 o envelope SOAP que será enviado como resposta. A única diferença entre as mensagens de resposta dos dois serviços é que, caso o empréstimo seja processado pelo serviço ManagerLoadReview, a palavra NORMAL será substituída pela palavra MANAGER na resposta gerada.

[5] O termo "serviço de negócio" é usado por esse barramento para identificar todos os serviços que foram criados e importados pelo barramento. Ou seja, todo serviço que não é nativo do barramento é denominado serviço de negócio.

Response Document

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Header/>
    <soapenv:Body>
        <m:processLoanAppResponse xmlns:m="http://example.org">
            <return>
                <java:Name xmlns:java="java:normal.client">Smith</java:Name>
                <java:SSN xmlns:java="java:normal.client">1234567</java:SSN>
                <java:Rate xmlns:java="java:normal.client">5.3</java:Rate>
                <java:Amount xmlns:java="java:normal.client">9000000</java:Amount>
                <java:NumOfYear xmlns:java="java:normal.client">10</java:NumOfYear>
                <java:Notes xmlns:java="java:normal.client">
                    APPROVED BY THE &lt;i&gt;&lt;b&gt;NORMAL&lt;/b&gt;&lt;/i&gt; LOAN APPLICATION PROCESSING SERVICE
                </java:Notes>
            </return>
        </m:processLoanAppResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Figura 14 – Mensagem SOAP de resposta dos serviços de aprovação de empréstimo

A.1.2.3.1 Publicação de recursos em um projeto

Para disponibilizarmos essas funcionalidades no ESB, precisaremos cadastrar cada recurso. Porém, antes do cadastramento, é preciso entender como funcionam as configurações no ALSB. A cada alteração de configuração realizada no ESB, é necessário criar uma sessão. Após a criação da sessão, todas as alterações realizadas no barramento são armazenadas em arquivos temporários e somente são aplicadas após clicar na opção Activate para ativar a sessão. Caso o administrador saia da tela de administração sem ativar a sessão, as alterações não serão aplicadas.

A Figura 15 apresenta a tela principal do AquaLogic Service Bus. No canto superior está o menu Change Center e a opção Edit, utilizada para iniciar uma nova sessão.

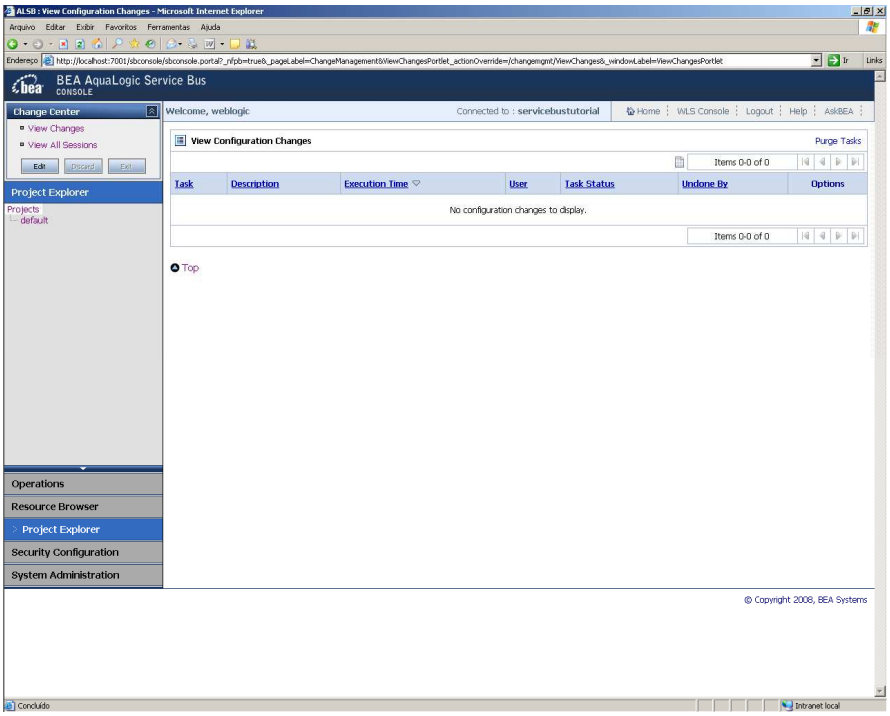

Figura 15 – Tela principal do ALSB

Após clicarmos na opção Edit, o Change Center aparecerá modificado, como mostra a Figura 16. A partir desse momento, é permitido alterar configurações do barramento. Ao final das alterações, o botão Activate deve ser acionado.

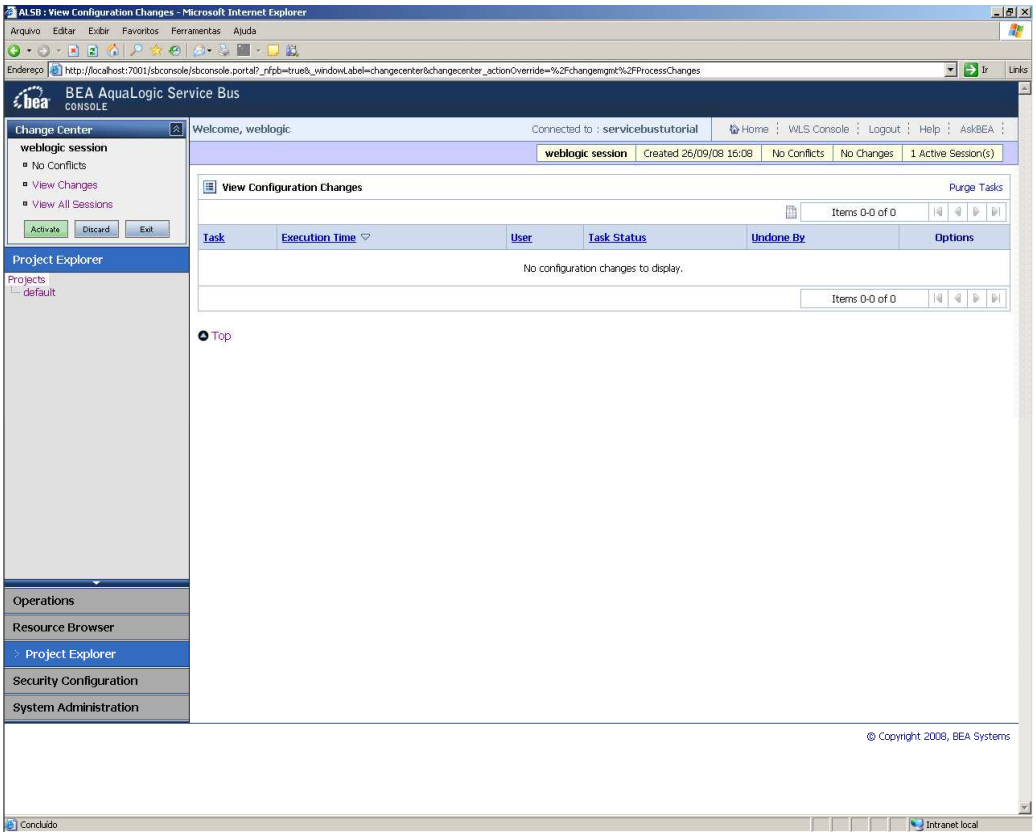

Figura 16 – Tela principal do ALSB com seção de alterações criada

No ambiente do ALSB, os recursos (WSDL, XSD, etc) são agrupados por projetos. Esses projetos podem ser visualizados no Project Explorer. No Project Explorer da Figura 16 acima ainda não existe nenhum projeto definido e, portanto, temos somente o item default. Cada projeto é representado como uma pasta no console. Pode-se adicionar pastas e navegar nessa árvore de pastas criada. Existem tipos de recursos predefinidos no ALSB, como interfaces WSDL, transformações XSLT, XQuery, serviços de conta, etc (veja Figura 23). Cada tipo de recurso é representado por uma subpasta dentro da pasta do projeto. Pode-se criar e configurar os recursos dentro das suas respectivas pastas.

Vamos criar um projeto no barramento, chamado MontgageBroker, e criar 3 pastas para os recursos pré-definidos: ProxyService, BusinessService e WSDL. A primeira pasta conterá o serviço de Proxy, o qual fará o roteamento das mensagens. A segunda pasta conterá os serviços de negócio que processarão as mensagens e a última pasta conterá as interfaces dos serviços.

Para criar o projeto, clicaremos no ProjectExplorer, o qual disponibilizará o campo para digitarmos o nome do projeto, conforme mostra a Figura 17.

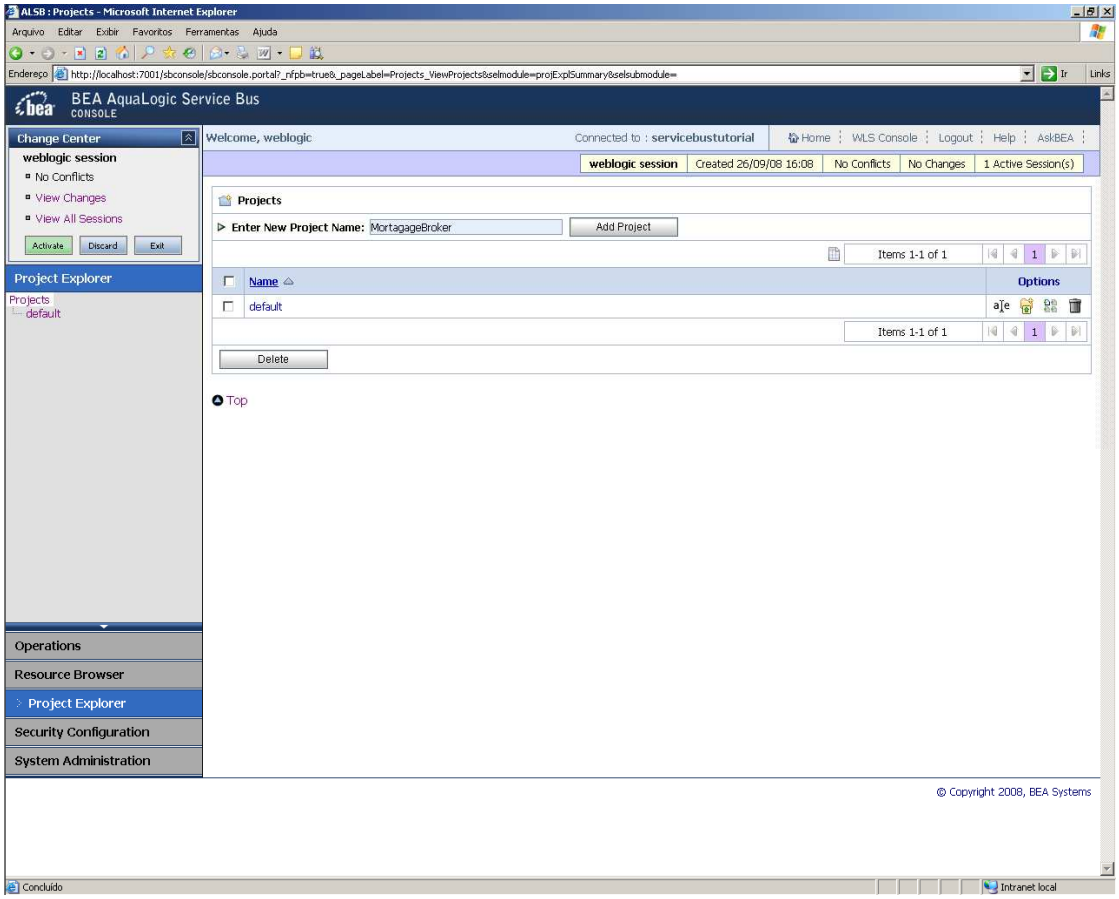

Figura 17 – Tela para criação de projetos no barramento

Ao clicarmos na opção Add Project, o projeto será criado e aparecerá no Project Explorer (Figura 18).

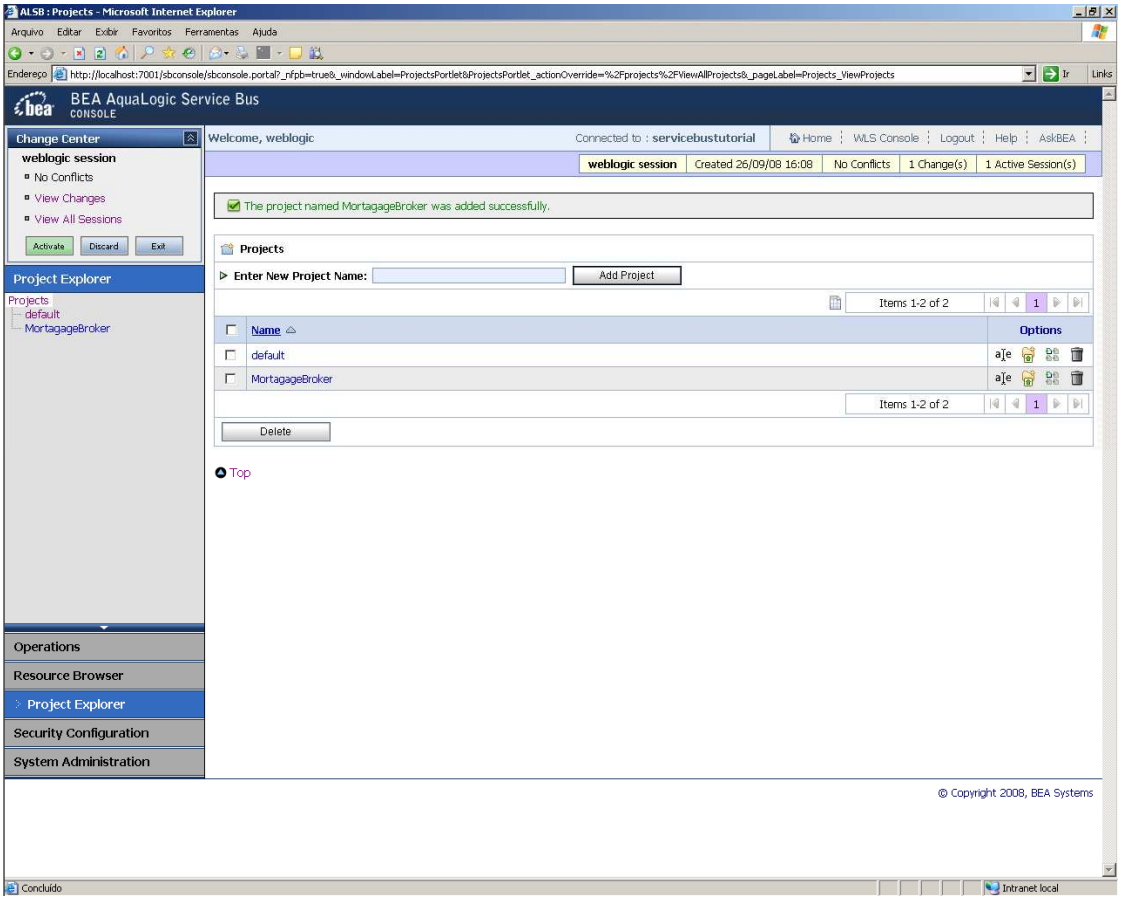

Figura 18 – Tela com projeto criado no barramento

Para criarmos as pastas dentro do projeto, primeiro clicaremos no nome do projeto e a tela apresentada na Figura 19 ficará disponível.

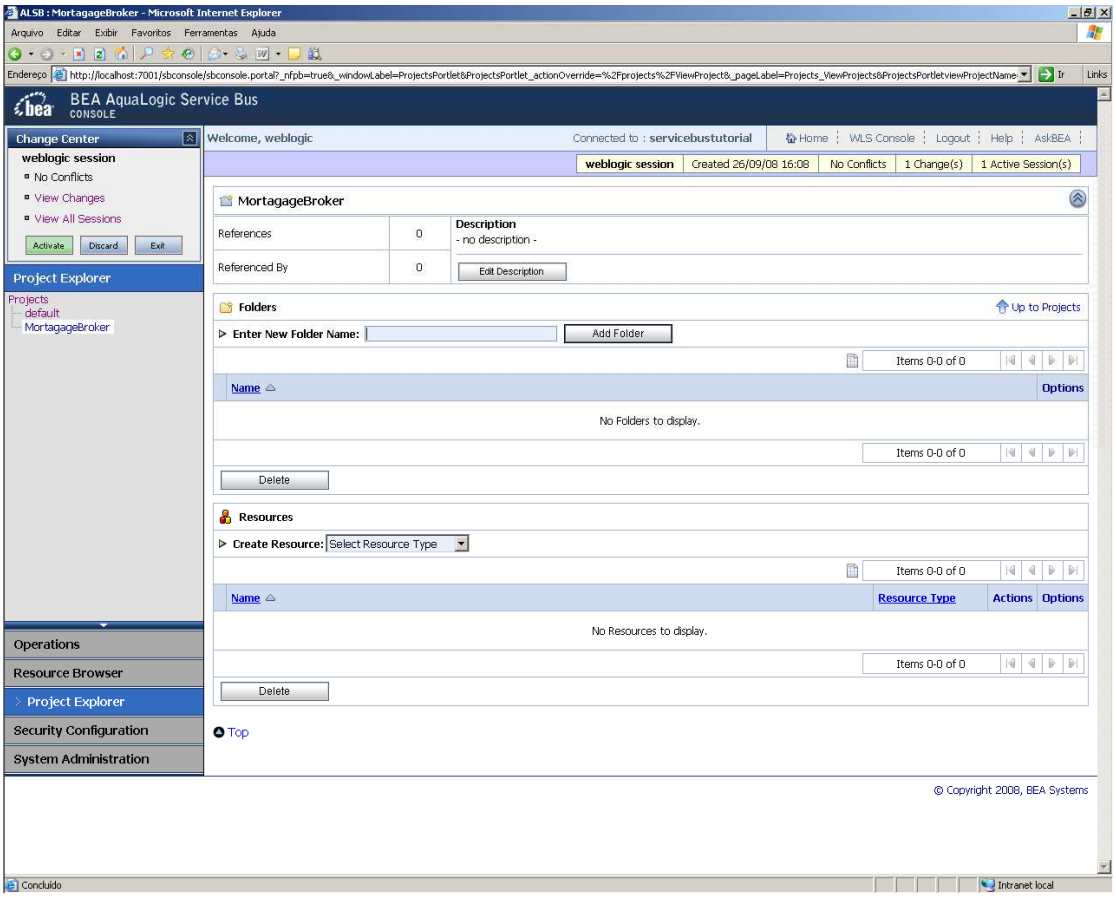

Figura 19 – Tela com detalhamento do projeto no barramento

No campo Enter New Folder Name, colocaremos os nomes das nossas pastas e clicaremos em Add Folder. Ao inserirmos as três pastas, nossa tela ficará como apresentado na Figura 20.

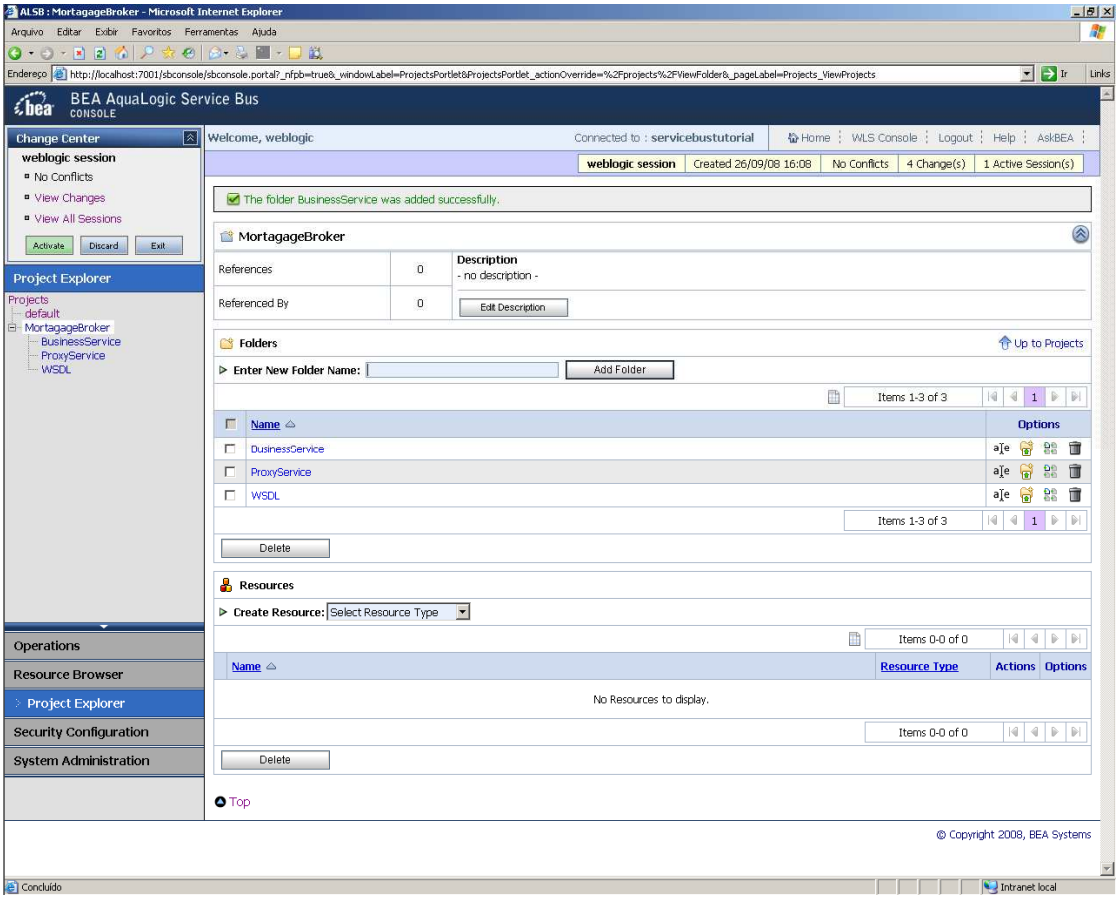

Figura 20 – Tela com pastas do projeto

Em seguida, adicionaremos os recursos. Adicionaremos em primeiro lugar os arquivos WSDL [WSDL, 2008]. Um arquivo WSDL define o contrato público (especificação da interface) entre um cliente e um serviço. O arquivo contém a descrição formal de um serviço web. Um documento WSDL é usado para descrever qual a interface do web service, onde ele reside e como invocá-lo. A criação do arquivo WSDL é necessária ser feita antes de qualquer outro registro de recurso, uma vez que as tarefas de registro dependem da descrição WSDL. O documento WSDL é usado posteriormente para registrar serviços de negócio com serviços de Proxy. Vale lembrar também que o barramento não é um editor e, neste caso, é necessário que os arquivos WSDL já tenham sido criados num ambiente de desenvolvimento. A tarefa do usuário administrador do barramento não é desenvolver aplicações, mas configurar as aplicações desenvolvidas anteriormente e prover soluções de serviço para algumas dessas aplicações (como o nosso serviço de Proxy que será criado mais a frente neste documento).

Nas Figura 21 e Figura 22, estão descritos o conteúdo dos arquivos WSDL.

```
<?xml version='1.0'?>
<definitions                                name="NormalLoanApprovalServiceDefinitions"
targetNamespace="http://example.org"           xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:s0="java:normal.client"                          xmlns:s1="http://example.org"
xmlns:s2="http://schemas.xmlsoap.org/wsdl/soap/">
  <n1:types xmlns:n1="http://schemas.xmlsoap.org/wsdl/">
    <xs:schema     attributeFormDefault="unqualified"     elementFormDefault="qualified"
targetNamespace="java:normal.client"          xmlns:n1="http://schemas.xmlsoap.org/wsdl/"
xmlns:s0="java:normal.client"                          xmlns:s1="http://example.org"
xmlns:s2="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
      <xs:complexType name="LoanStruct">
        <xs:sequence>
          <xs:element minOccurs="0" name="Name" nillable="true" type="xs:string"/>
          <xs:element minOccurs="0" name="SSN" nillable="true" type="xs:string"/>
          <xs:element minOccurs="0" name="Rate" nillable="false" type="xs:double"/>
          <xs:element minOccurs="0" name="Amount" nillable="false" type="xs:long"/>
          <xs:element minOccurs="0" name="NumOfYear" nillable="false" type="xs:int"/>
          <xs:element minOccurs="0" name="Notes" nillable="true" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
  </n1:types>
  <n2:message xmlns:n2="http://schemas.xmlsoap.org/wsdl/" name="processLoanApp">
    <n2:part name="loanRequest" type="s0:LoanStruct"/>
  </n2:message>
  <n3:message xmlns:n3="http://schemas.xmlsoap.org/wsdl/" name="processLoanAppResponse">
    <n3:part name="return" type="s0:LoanStruct"/>
  </n3:message>
  <n4:portType xmlns:n4="http://schemas.xmlsoap.org/wsdl/" name="myPortType">
    <n4:operation name="processLoanApp" parameterOrder="loanRequest">
      <n4:input message="s1:processLoanApp"/>
      <n4:output message="s1:processLoanAppResponse"/>
    </n4:operation>
  </n4:portType>
  <n5:binding                                  xmlns:n5="http://schemas.xmlsoap.org/wsdl/"
name="NormalLoanApprovalServiceSoapBinding" type="s1:myPortType">
    <s2:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <n5:operation name="processLoanApp">
      <s2:operation soapAction="http://example.orgprocessLoanApp" style="rpc"/>
      <n5:input>
        <s2:body               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://example.org" use="encoded"/>
      </n5:input>
      <n5:output>
        <s2:body               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://example.org" use="encoded"/>
      </n5:output>
    </n5:operation>
  </n5:binding>
  <n6:service                                  xmlns:n6="http://schemas.xmlsoap.org/wsdl/"
name="NormalLoanApprovalService">
    <n6:port binding="s1:NormalLoanApprovalServiceSoapBinding" name="helloPort">
      <s2:address location="http://localhost:7021/njws_basic_ejb/NormalSimpleBean"/>
    </n6:port>
  </n6:service>
</definitions>
```

Figura 21 – Conteúdo do arquivo normalLoanApprovalService.wsdl

```
<?xml version='1.0'?>
<definitions                                    name="ManagerApprovalServiceDefinitions"
targetNamespace="http://example.org"            xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:s0="java:normal.client"                                xmlns:s1="http://example.org"
xmlns:s2="http://schemas.xmlsoap.org/wsdl/soap/">
  <n1:types xmlns:n1="http://schemas.xmlsoap.org/wsdl/">
    <xs:schema     attributeFormDefault="unqualified"     elementFormDefault="qualified"
targetNamespace="java:normal.client"         xmlns:n1="http://schemas.xmlsoap.org/wsdl/"
xmlns:s0="java:normal.client"                                xmlns:s1="http://example.org"
xmlns:s2="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:complexType name="LoanStruct">
        <xs:sequence>
          <xs:element minOccurs="0" name="Name" nillable="true" type="xs:string"/>
          <xs:element minOccurs="0" name="SSN" nillable="true" type="xs:string"/>
          <xs:element minOccurs="0" name="Rate" nillable="false" type="xs:double"/>
          <xs:element minOccurs="0" name="Amount" nillable="false" type="xs:long"/>
          <xs:element minOccurs="0" name="NumOfYear" nillable="false" type="xs:int"/>
          <xs:element minOccurs="0" name="Notes" nillable="true" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
  </n1:types>
  <n2:message xmlns:n2="http://schemas.xmlsoap.org/wsdl/" name="processLoanApp">
    <n2:part name="loanRequest" type="s0:LoanStruct"/>
  </n2:message>
  <n3:message xmlns:n3="http://schemas.xmlsoap.org/wsdl/" name="processLoanAppResponse">
```

```
    <n3:part name="return" type="s0:LoanStruct"/>
  </n3:message>
  <n4:portType xmlns:n4="http://schemas.xmlsoap.org/wsdl/" name="myPortType">
    <n4:operation name="processLoanApp" parameterOrder="loanRequest">
      <n4:input message="s1:processLoanApp"/>
      <n4:output message="s1:processLoanAppResponse"/>
    </n4:operation>
  </n4:portType>
  <n5:binding                                  xmlns:n5="http://schemas.xmlsoap.org/wsdl/"
name="ManagerApprovalServiceSoapBinding" type="s1:myPortType">
    <s2:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <n5:operation name="processLoanApp">
      <s2:operation soapAction="http://example.orgprocessLoanApp" style="rpc"/>
      <n5:input>
        <s2:body               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://example.org" use="encoded"/>
      </n5:input>
      <n5:output>
        <s2:body               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://example.org" use="encoded"/>
      </n5:output>
    </n5:operation>
  </n5:binding>
  <n6:service xmlns:n6="http://schemas.xmlsoap.org/wsdl/" name="ManagerApprovalService">
    <n6:port binding="s1:ManagerApprovalServiceSoapBinding" name="helloPort">
      <s2:address location="http://localhost:7021/mjws_basic_ejb/ManagerSimpleBean"/>
    </n6:port>
  </n6:service>
</definitions>
```

Figura 22 – Conteúdo do arquivo ManagerApprovalService.wsdl

Para incluirmos os WSDL, escolhemos o tipo de recurso que queremos inserir. A lista de recursos disponíveis está no dropdown, em Resources ➔ Create Resource, conforme mostrado na Figura 23. Escolheremos a opção WSDL.

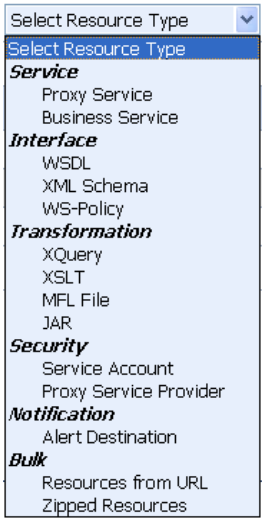

Figura 23 – Lista de recursos suportados pelo barramento

Em seguida, preenchemos o formulário com o nome do recurso, o qual chamaremos de normalLoan, e o caminho do arquivo WSDL que descreve a interface do serviço que processa empréstimos com taxa de juros maior que 5% (Figura 24).

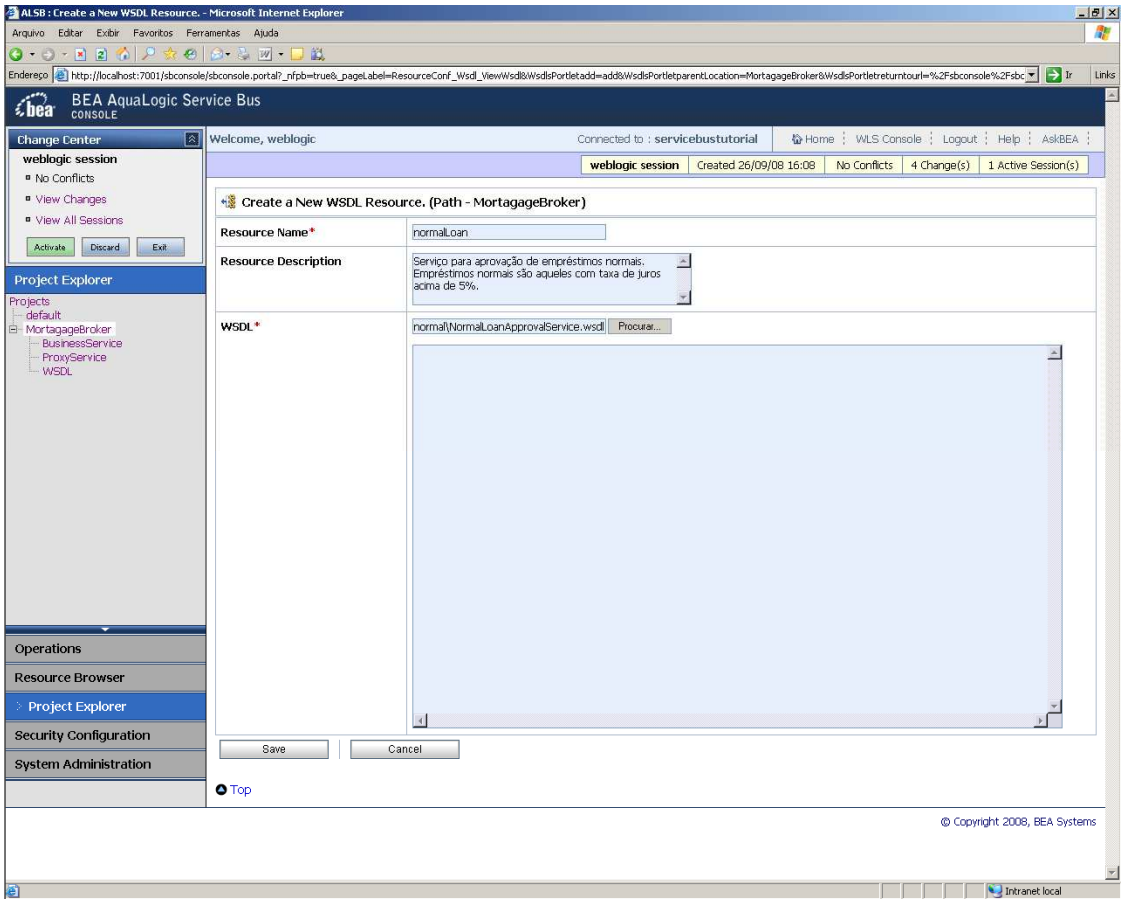

Figura 24 – Tela para registro de interface de serviços

Ao aceitarmos os dados, o barramento registra a interface do serviço normalLoan e o acrescenta ao nosso projeto, conforme é mostrado na tela Figura 25.

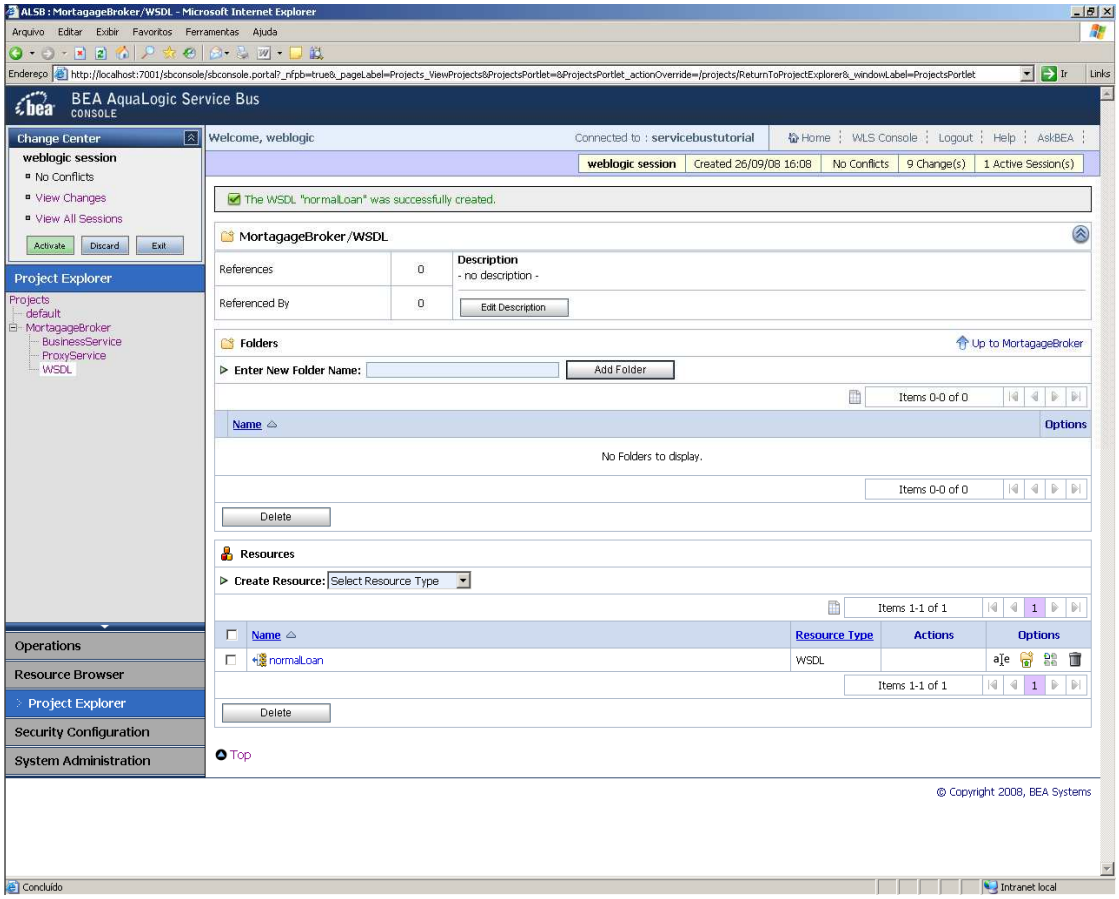

Figura 25 – Projeto com serviço cadastrado

Ao clicarmos no recurso WSDL, podemos visualizar os detalhes da interface (Figura 26).

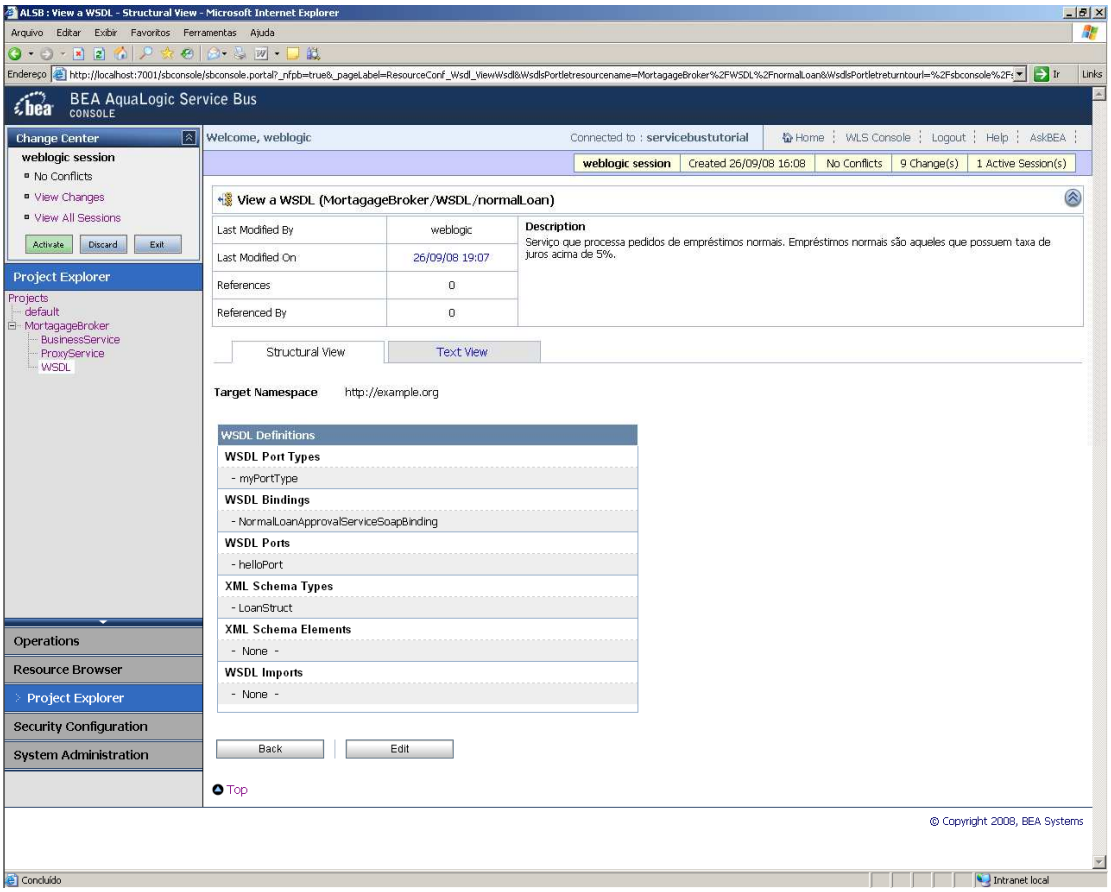

Figura 26 – Serviço normalLoan

Repetiremos os passos acima para registrar o serviço ManagerApproval, o qual será o responsável por aprovar empréstimos com taxas de juros muito baixas, abaixo de 5%. Ao final do processo, teremos os dois serviços registrados no nosso projeto (Figura 27).

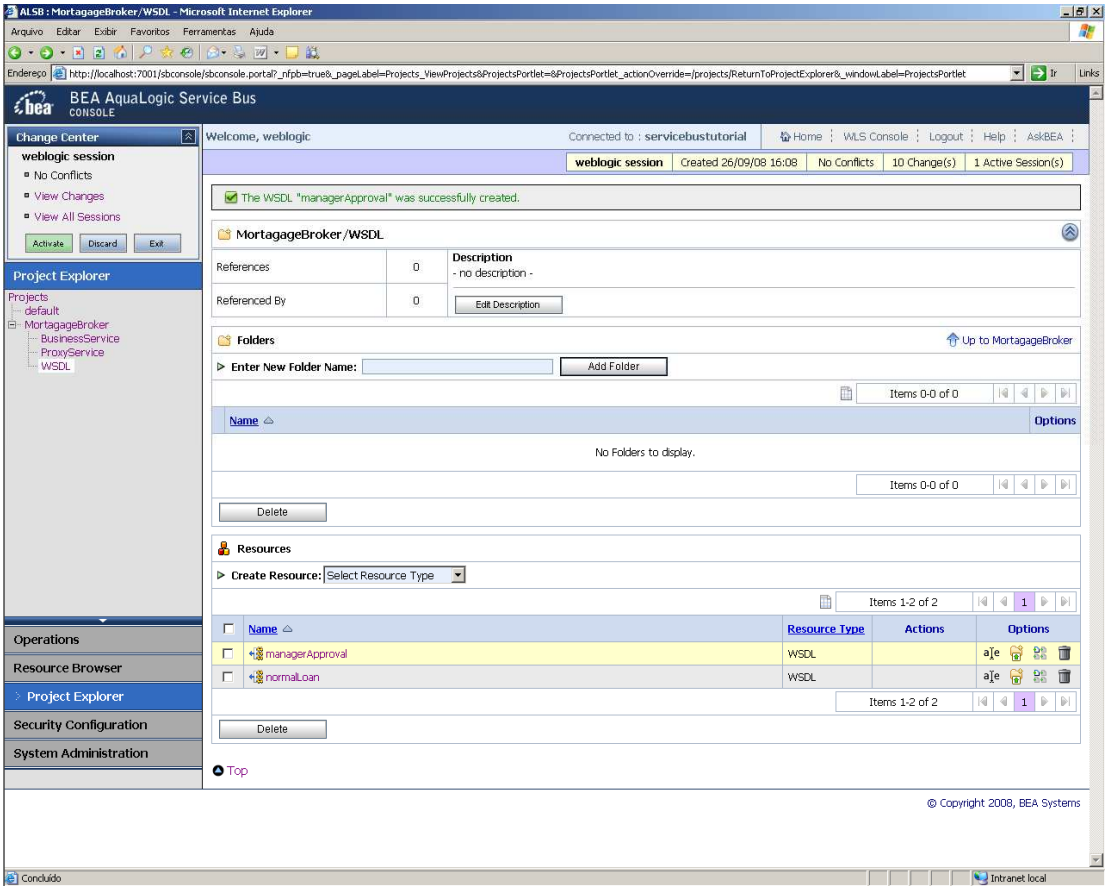

Figura 27 – Projeto com serviço cadastrado

Para finalizar a disponibilização das interfaces dos serviços no barramento, iremos ativar essa sessão de mudança do barramento, clicando no botão Activate, no Change Center. É interessante notar nesse ponto a funcionalidade de registro de alterações feitas pelo administrador do barramento.

Ao clicarmos no botão Activate, somos direcionados para a tela apresentada na Figura 28, a qual disponibiliza um formulário para descrição da semântica das alterações realizadas nesta sessão.

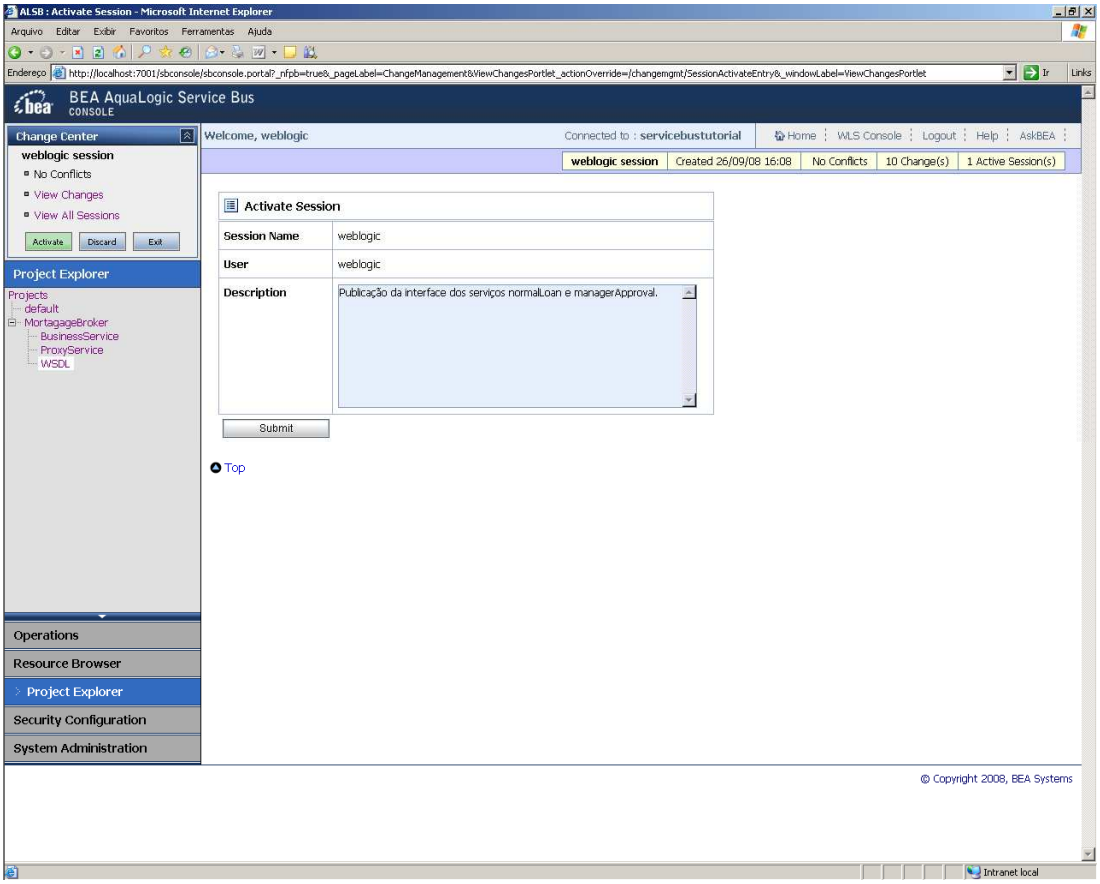

Figura 28 – Ativando sessão no barramento

Uma vez submetido o formulário, a seção é ativada e registrada, conforme mostra a Figura 29.

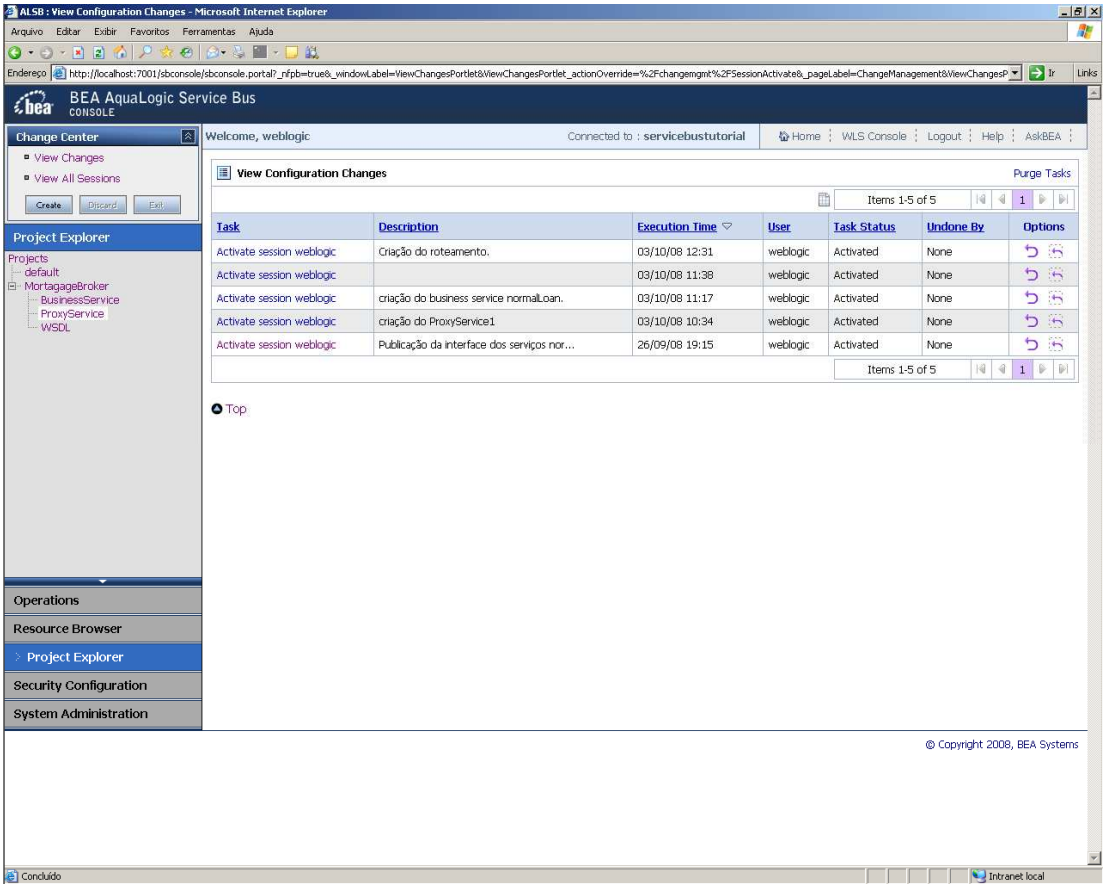

Figura 29 – Registro da sessão

Agora, é possível conhecer quem realizou alterações no barramento e, principalmente, podemos desfazer as alterações (clicando na seta rosa em Options) ou rastrear as alterações feitas em uma determinada sessão. Para rastrear as alterações feitas na sessão, basta clicarmos o nome da tarefa. No nosso caso, a tarefa recebeu o nome de "Activate session weblogic". Ao clicarmos nesse registro de sessão, podemos visualizar os detalhes da tarefa, conforme mostra a Figura 30, que mostra as pastas e tipos de recursos criados durante essa sessão.

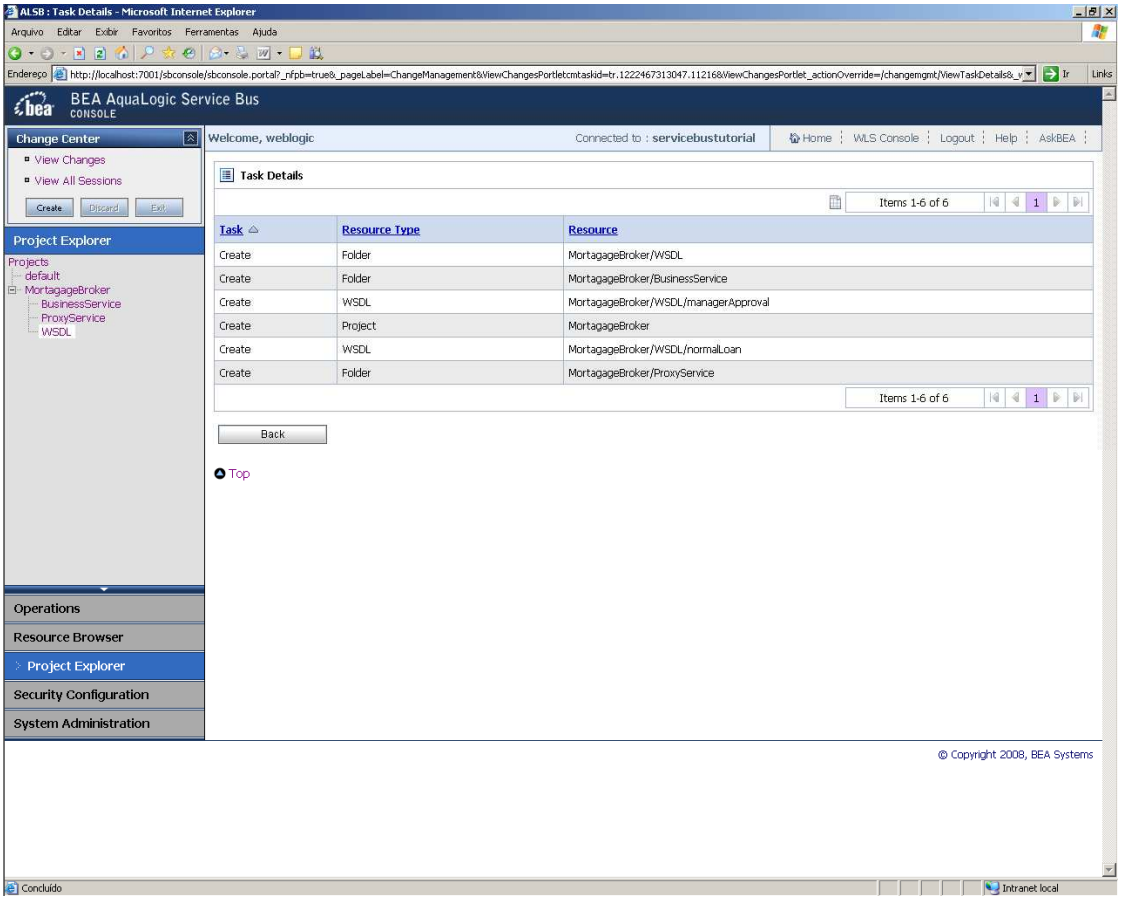

Figura 30 – Detalhes da sessão

Neste exato momento, conseguimos, parcialmente, preparar o barramento para disponibilizar dois serviços web. O barramento está parcialmente preparado, pois uma particularidade do ALSB é que, ao publicar os WSDL como fizemos, ainda não estamos disponibilizando os serviços. Até o momento, somente criando um projeto e colocamos dois recursos que são dois arquivos WSDL. Para que o serviço seja disponibilizado realmente no barramento, teremos que criar um recurso chamado BusinessService e associar esse recurso ao WSDL que adicionamos ao projeto. Esse passo é efetuado na seção A.1.2.3.3.

Porém, antes, vamos começar a definir o nosso Proxy.

A.1.2.3.2 Criação de um serviço de proxy

Nosso exemplo vai além da disponibilização de serviços web no barramento, vamos demonstrar como criar o roteamento de mensagem entre esses serviços utilizando um serviço de Proxy. O serviço de Proxy será o responsável por rotear as mensagens para os serviços web apropriados, com base no conteúdo da mensagem.

O serviço de Proxy é um tipo de serviço web específico do barramento e, para criá-lo, clicamos na opção ProxyService nas opções de recursos do ESB, conforme foi visto na Figura 23. O ProxyService será chamado de LoanGateway1. A Figura 31 apresenta a tela de criação de um ProxyService.

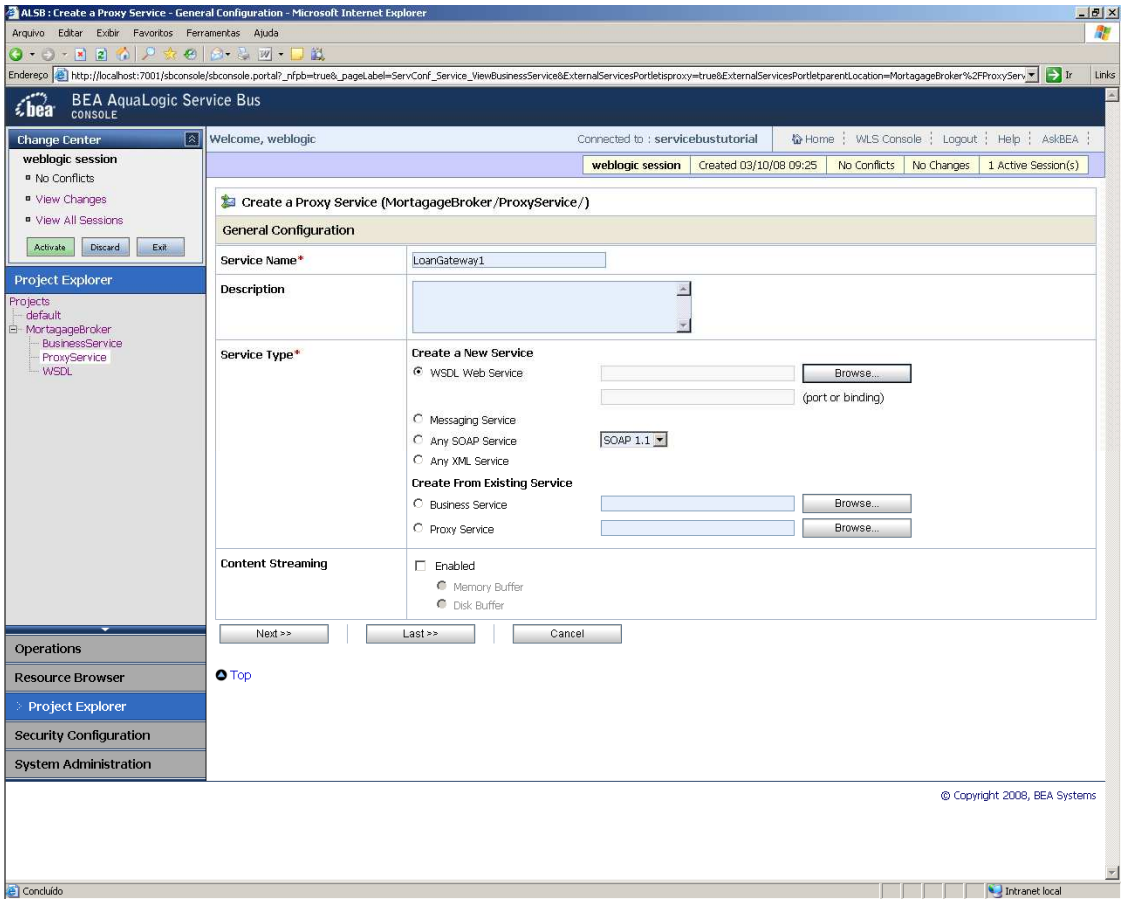

Figura 31 – Cadastramento do ProxyService

Esse serviço deverá ter um WSDL que o descreva, como todo o serviço. Uma opção interessante do ALSB é permitir criar esse WSDL a partir do zero ou, para facilitar, utilizando um WSDL já existente. Pelo nosso exemplo, o LoanGateway1 irá se comunicar diretamente com o serviço normalLoan (ou seja, qualquer mensagem que o Proxy receber será enviada para o serviço normalLoan. Definiremos as exceções posteriormente). Vamos criar o WSDL do LoanGateway1 baseado no WSDL normalLoan, herdando suas definições. Para tal, clicaremos na opção Browse, em Creating a New Service ➔ WSDL Web Service e escolheremos o WSDL do serviço normalLoan. Será exibida a tela apresentada na Figura 32.

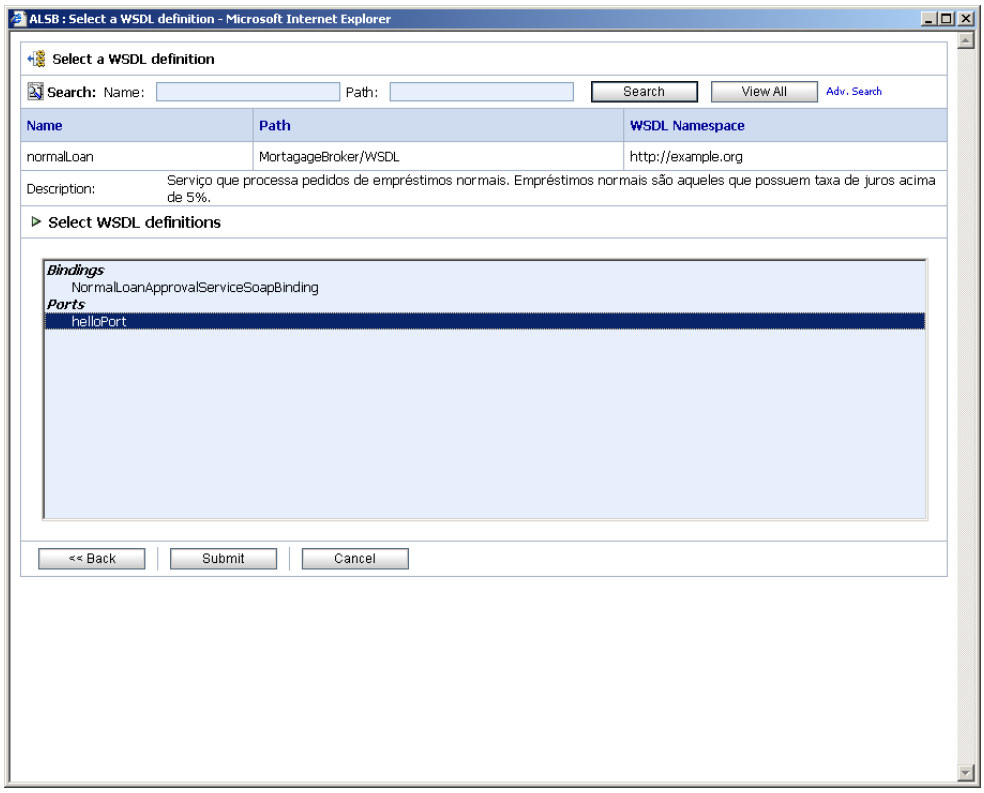

Figura 32 – Criando um WSDL baseado em definições existentes

A tela exibe as definições do WSDL. Escolheremos a porta (Port) helloPort, ou seja, estamos criando um WSDL para o GatewayProxy1 que terá as mesmas definições de porta do helloPort do WSDL normalLoan. Em seguida, seremos direcionados para continuar a configuração do serviço GatewayProxy1. Inicialmente, definiremos as configurações de transporte das mensagens. Escolheremos que esse serviço irá se comunicar através do protocolo HTTP e será acessado através do endpoint chamado /loan/gateway1, ou seja, essa será a URI para que aplicações enviem mensagens para esse serviço. Neste ponto, é possível também configurar algum cabeçalho específico que o serviço possa ler. Contudo, não configuraremos cabeçalhos por enquanto. A Figura 33 exibe essas configurações.

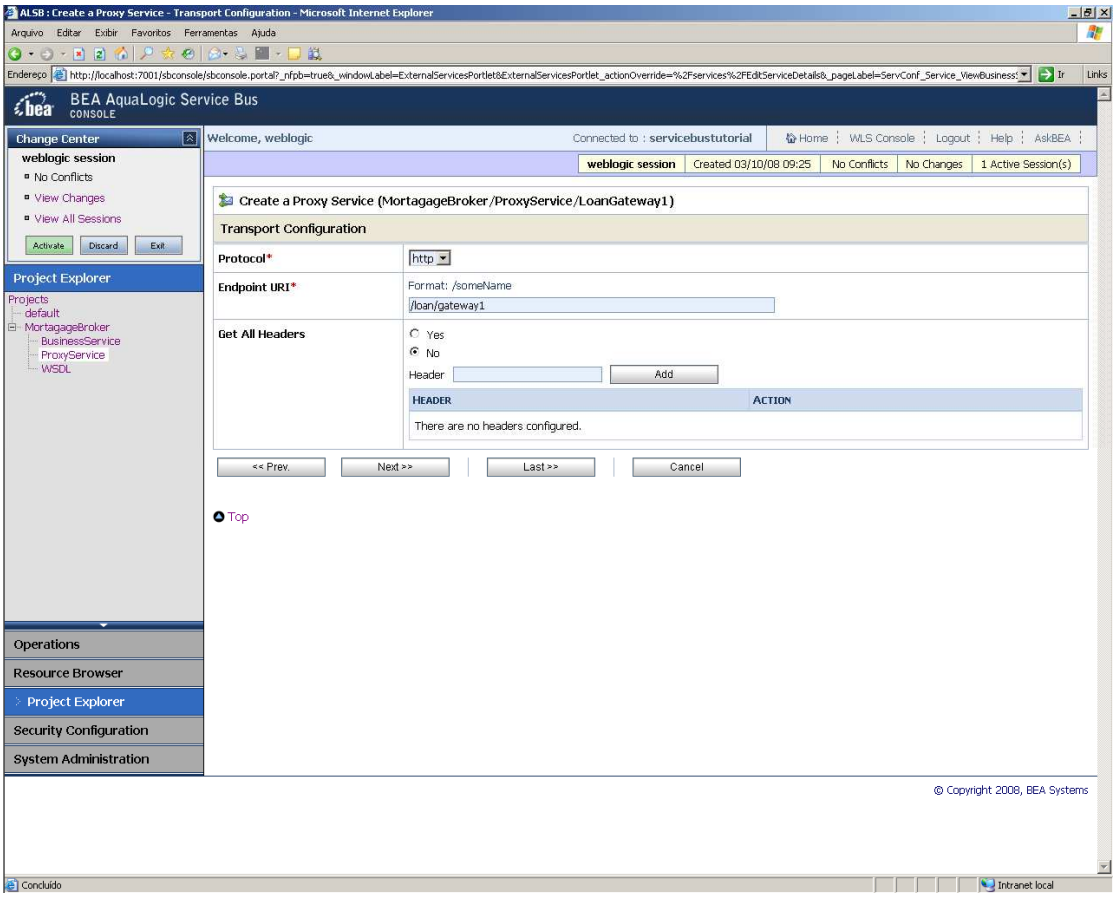

Figura 33 – Definindo configurações de transporte

Continuando, seremos direcionados para que possamos definir as configurações de transporte HTTP. É possível, por exemplo, definir cabeçalhos de autenticação para o serviço, impedindo que ele seja chamado por aplicações desconhecida (Figura 34). Contudo, para simplificar, não definiremos questões de autenticação neste exemplo.

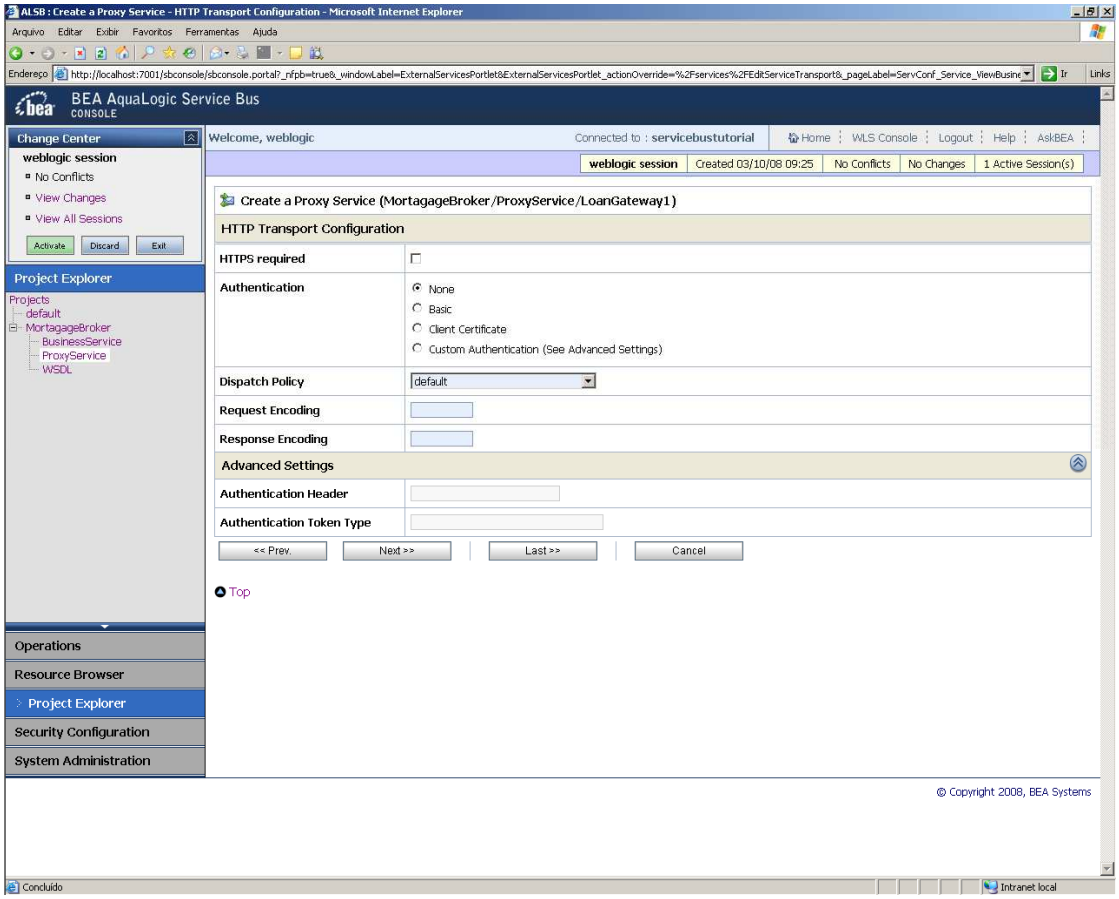

Figura 34 – Definindo configurações de transporte HTTP

Por fim, teremos que definir as configurações de seleção de operação do serviço. Este termo no ESB (configuração de seleção de operação) corresponde ao atributo *binding* da especificação WSDL, ou seja, são as definições de como será feita a chamada da operação na mensagem que será transmitida em protocolo HTTP. No nosso exemplo, utilizaremos a opção mais comum, que é transmitir a chamada da operação como corpo de mensagem SOAP, conforme mostra a Figura 35.

Vale atentar que o serviço de Proxy no barramento ALSB é, por *default*, um web service chamado via RPC. Assim, o corpo da mensagem armazena qual a operação será executada. Uma operação em um WSDL de serviço web desenvolvido em Java corresponde a um método público na classe. Se existem muitos métodos públicos acessíveis para um cliente, cada método terá uma operação definida no WSDL. Para uma mensagem baseada em RPC, o corpo da mensagem SOAP indica a operação selecionada pelo cliente. O ALSB permite que o cabeçalho SOAP também possa especificar a operação. Contudo, por convenção, o corpo da mensagem SOAP é quem especifica essa operação.

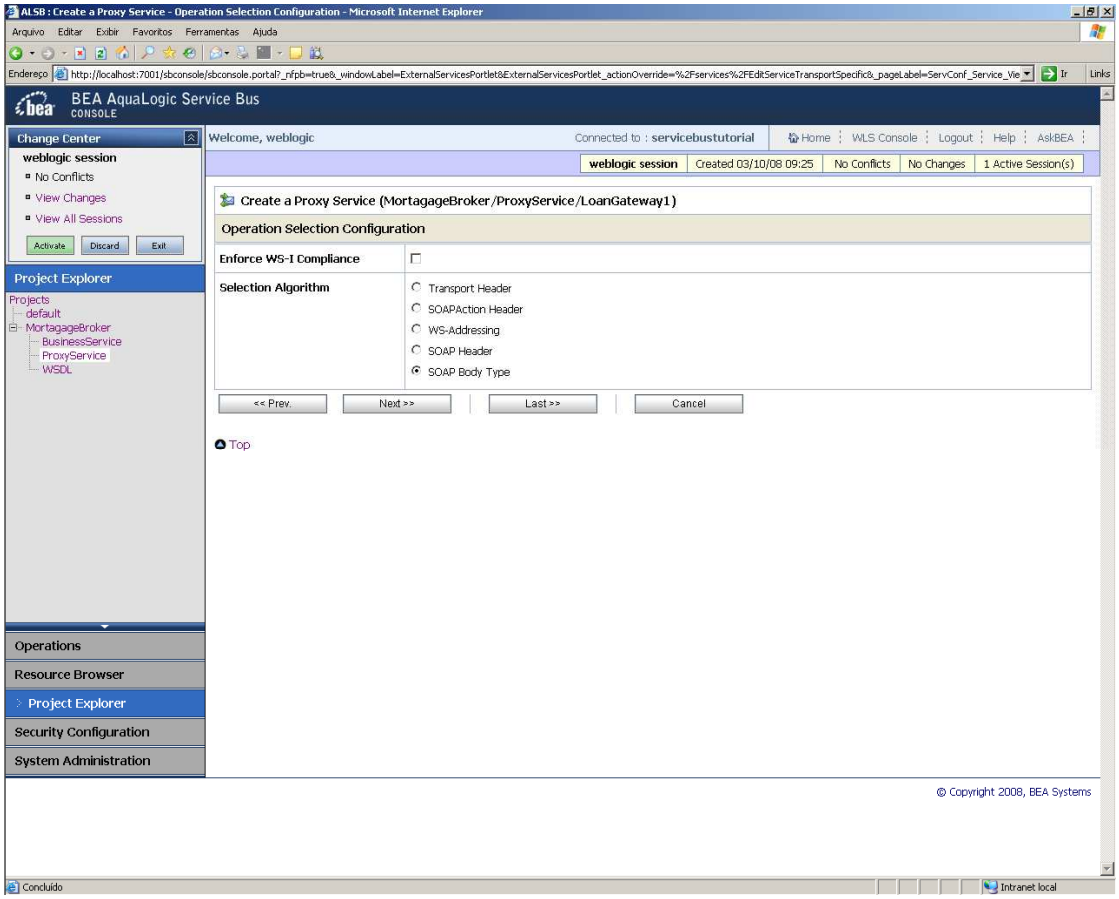

Figura 35 – Definindo configurações de seleção de operação do serviço

Após finalizarmos a operação, o serviço de Proxy estará disponível na nossa lista de ProxyServices, como mostra a Figura 36.

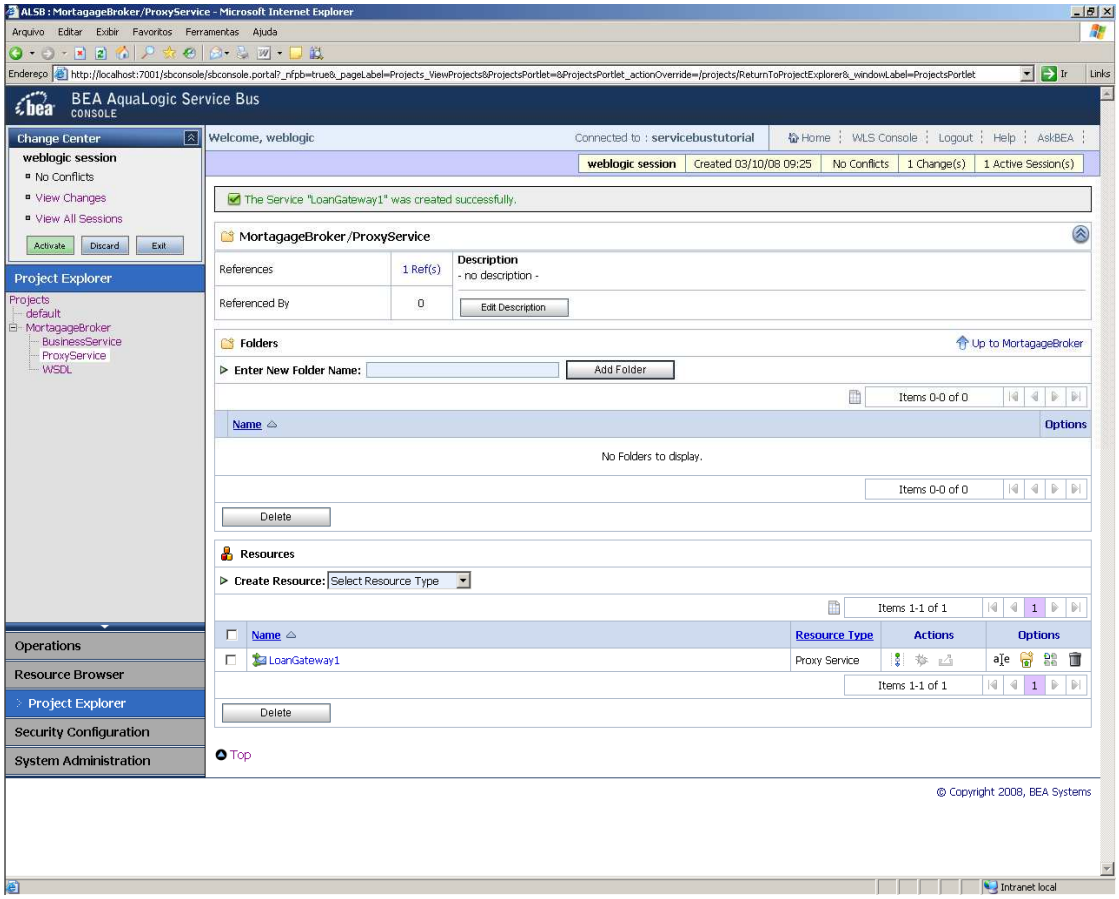

Figura 36 – Serviço de Proxy criado

Caso cliquemos no nome do serviço, podemos ver suas definições e serão exibidas novas configurações. Uma delas, interessante para SOA, é o monitoramento do serviço, descrito com mais detalhes no 0.

Neste ponto, temos um ProxyService chamado GatewayProxy1 que encaminhará todas as mensagens que receber para o serviço normalLoan. Posteriormente, adicionaremos uma exceção a esse Proxy: caso a mensagem contenha uma taxa de juros menor do que 5%, a mensagem será encaminhada para um serviço diferente. Esse passo está descrito na seção A.1.2.3.4.

A.1.2.3.3 Definição dos serviços no ALSB

Nesta seção, iremos fazer com que o barramento disponibilize os serviços descritos nos dois arquivos WSDL que adicionamos ao projeto. Para tal, criaremos o recurso BusinessService. Será criado um businessService para cada serviço, ou seja, um para o normalLoan e um para o managerApproval. Mostraremos aqui somente a criação deste primeiro serviço, uma vez que o processo será idêntico para o segundo.

Para começarmos, clicaremos no recurso BusinessService, nomearemos o serviço commom normalLoan (o mesmo nome do arquivo WSDL, o que é somente uma coincidência, não uma obrigatoriedade) e escolheremos basear esse serviço nas definição do WSDL normalLoan, conforme fizemos na seção anterior com o ProxyService. A Figura 37 apresenta a tela inicial de cadastramento do nosso serviço de negócio normalLoan.

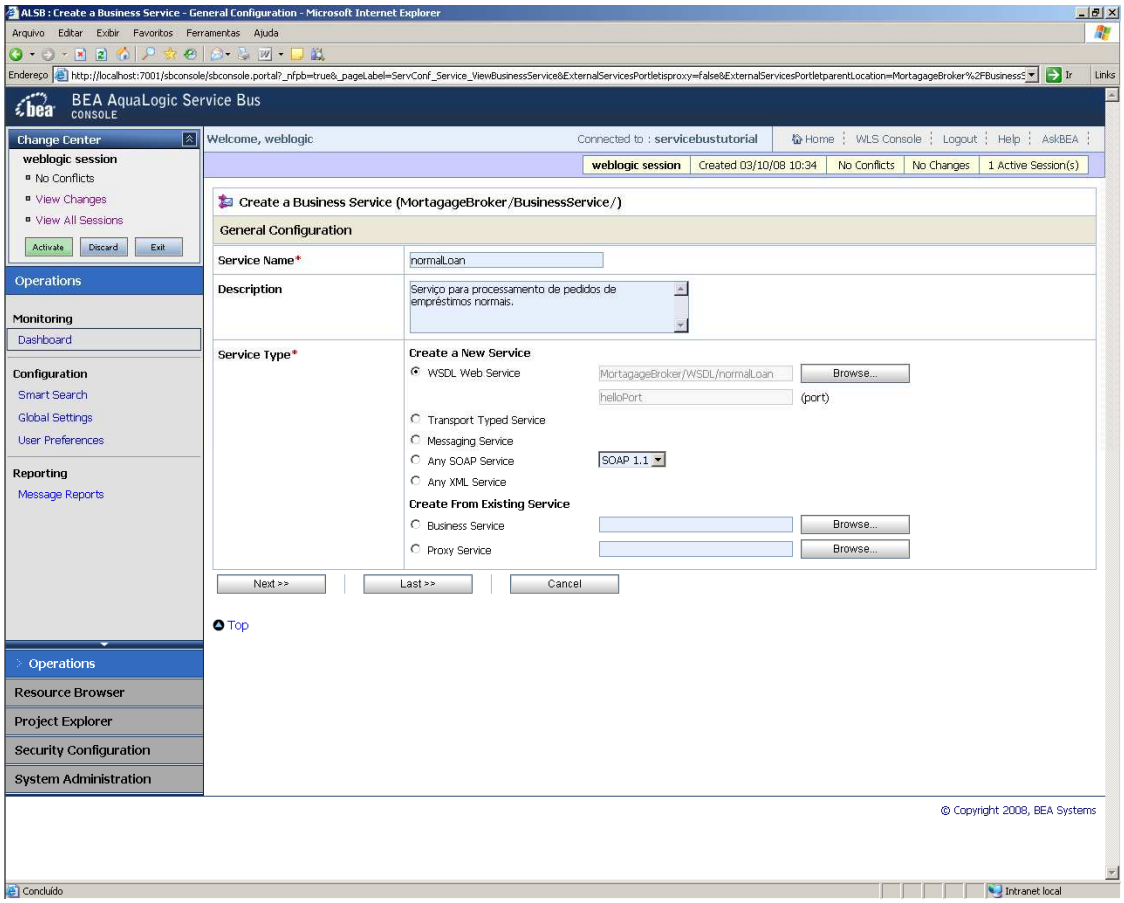

Figura 37 – Criando serviço de negócio normalLoan

Vale perceber que o requisito de se criar um BusinessService para os WSDL que foram inseridos no projeto torna a tarefa de publicação de serviços um tanto redundante, uma vez que repetiremos o que está no WSDL para o BusinessService. Porém, mesmo sendo parcialmente uma tarefa redundande, isso é necessário, pois o barramento permite definirmos configurações que não são possíveis em WSDL, uma vez que são específicas do barramento. Por exemplo, ao clicarmos em Next, somos direcionados para as configurações de transporte de mensagens do serviço e podemos definir, por exemplo, o algoritmo de balanceamento de carga que será usado para esse serviço. As opções são: none, round-robin, random ou random-weighted. Esse balanceamento é importante quando o serviço disponibiliza múltiplos endpoints (ou seja, cópias do mesmo serviço que podem estar no mesmo servidor ou até em outros servidores). Quando são definidos múltiplos endpoints e um algoritmo de balanceamento de carga, se um endpoint estiver sobrecarregado ou não estiver disponível em tempo de execução, a mensagem poderá ser enviada para o próximo serviço na lista de endpoints. No nosso exemplo, contudo, existe apenas um serviço. Então independente do tipo de balanceamento que escolhermos, ou mesmo se escolhermos nenhum, pois não afetará o comportamento do serviço. Na Figura 38, estas configurações estão exibidas. Repare no endereço http://localhost:7021/njws_basic_ejb/NormalSimpleBean que será o endpoint de acesso ao serviço que estamos publicando.

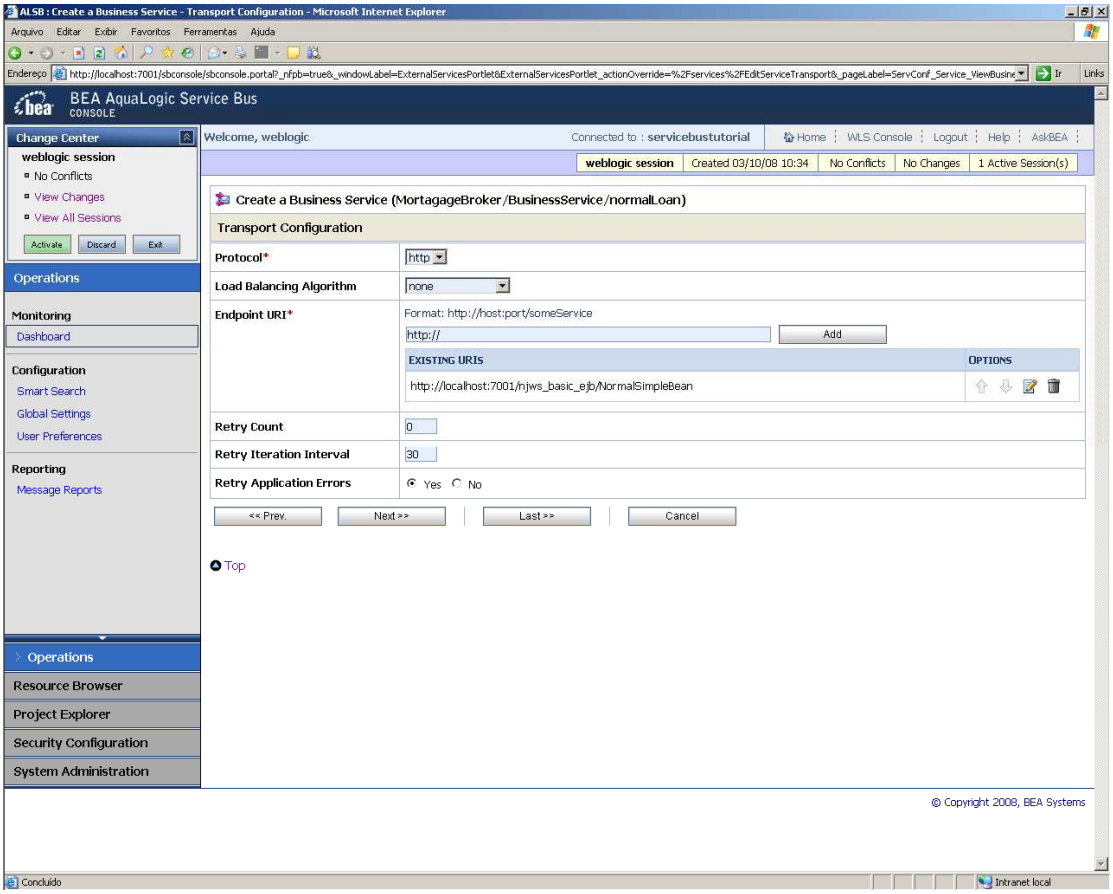

Figura 38 – Especificando as configurações de transporte de mensagens do serviço de negócio normalLoan

A partir desse ponto, os passos para criação do BusinessService são idênticos aos mostrados na seção A.1.2.3.2. Ao repetirmos esses passos para os dois serviços, estaremos disponibilizando o serviço no barramento. Na Figura 39, encontra-se a tela com os dois serviços de negócio criados. O serviço aprovador de créditos com taxa de juros menor que 5% foi chamado de ManagerLoanReview.

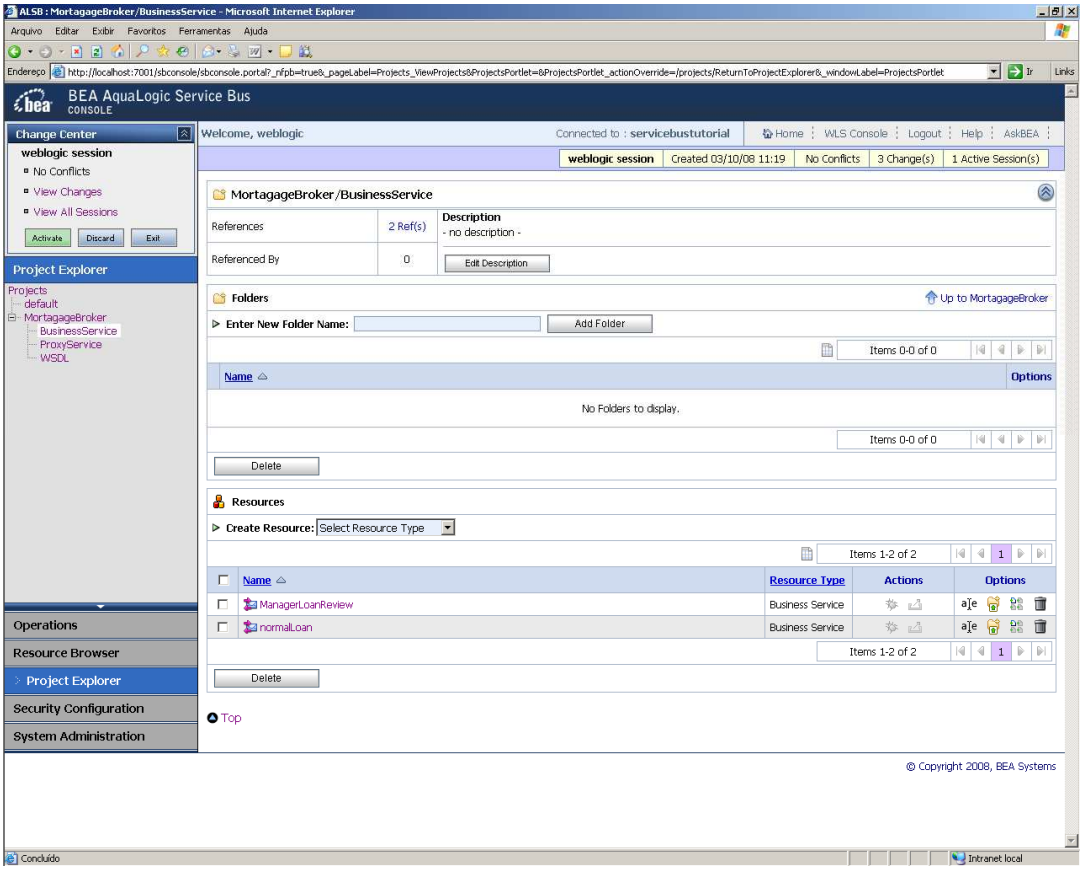

Figura 39 – Serviços de negócio criados

A.1.2.3.4 Configuração do roteamento de mensagem entre os serviços

Nesta seção, vamos definir o fluxo de mensagem entre os serviços. Isso é feito na opção Message Flows do ALSB, que define a implementação de serviços de Proxy. Os fluxos de mensagem podem ter zero ou mais pares de pipelines: pipelines de requisição e resposta para ProxyServices (ou para operações de serviços) e pipelines de manipulação de erros que podem ser definidos para outros estágios, pipelines ou ProxyServices. Os pipelines podem incluir um ou mais estágios, que, por sua vez, podem incluir ações. Um estágio é um elemento de um pipeline e também um container para ações definidas em um pipeline. Ações são elementos de um estágio de um pipeline que definem a manipulação de mensagens conforme elas passam pelo serviço de Proxy em tempo de execução.

Para editar o fluxo de mensagens do nosso serviço de Proxy, clicaremos na coluna Actions que aparece a frente do nome do nosso serviço, **LoanGateway1**. Seremos direcionados para a tela de edição gráfica do fluxo de mensagens, conforme é mostrado na figura Figura 40.

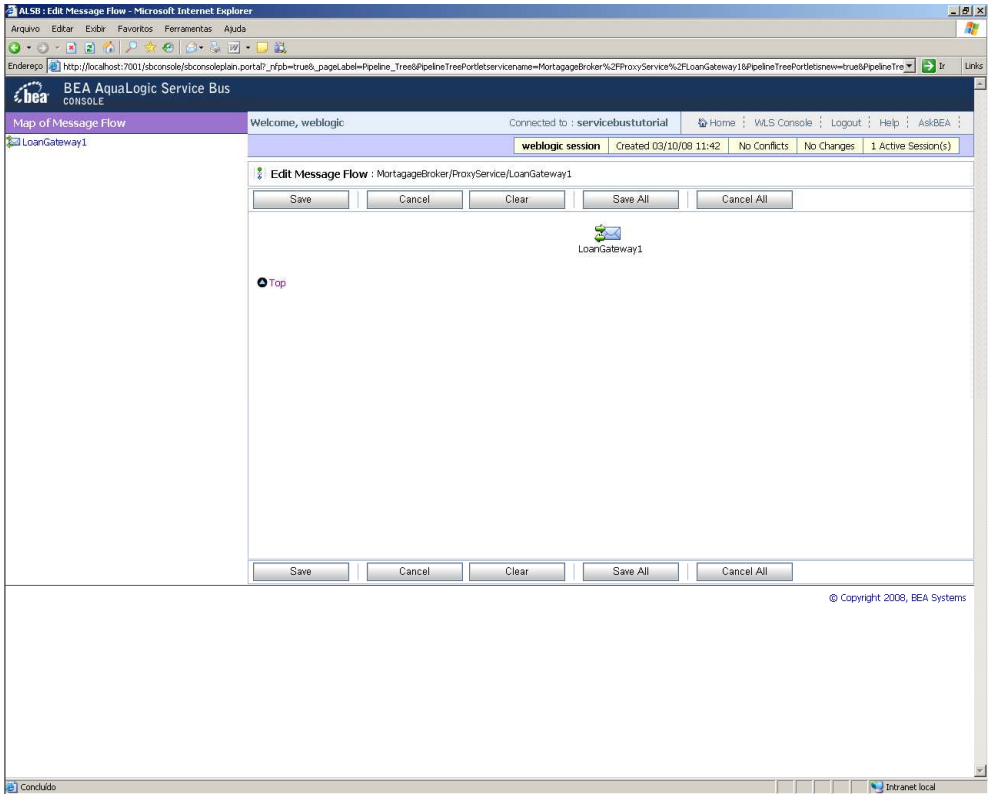

Figura 40 – Tela de edição de fluxo de mensagens

Criaremos uma rota de mensagem. Para tal, clicaremos no ícone que representa o ProxyService e aparecerão as opções exibidas na Figura 41. Clicaremos em "Add Route" para criar um estágio pro nosso serviço. Ao clicarmos nessa opção, será criado o elemento RouteNode1, conforme está exibido na Figura 42.

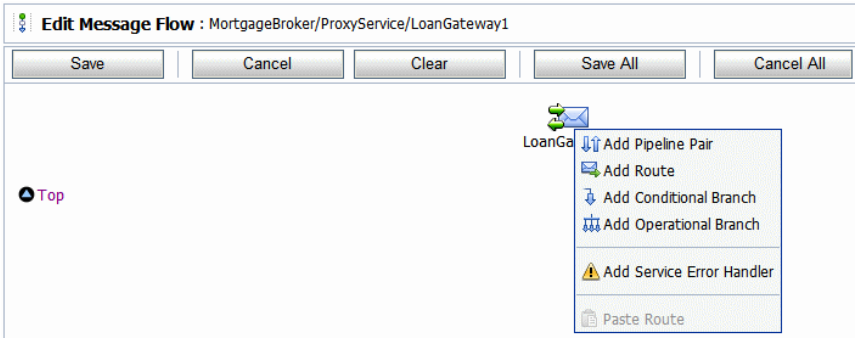

Figura 41 – Opções de criação de rotas de mensagens

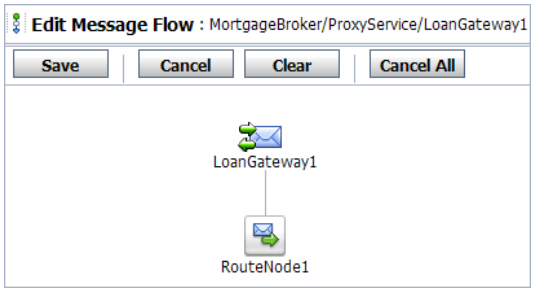

Figura 42 – Rota RouteNode1 criada

Clicaremos no RouteNote1 que criamos e escolheremos a opção Edit Note para criarmos a ação desse estágio. Seremos direcionados para a tela apresentada na Figura 43, a qual contém somente um único objeto com link de nome "Add an Action", conforme mostrado na Figura 43.

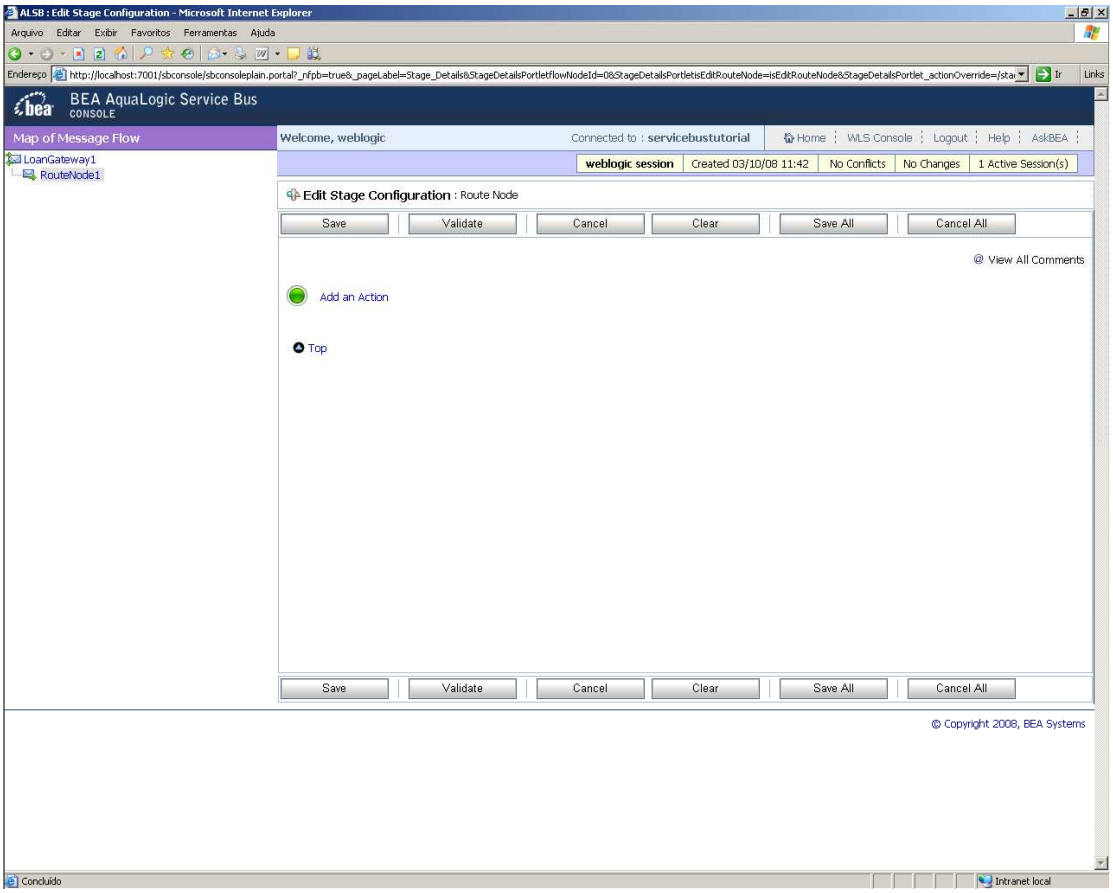

Figura 43 – Tela de edição de estágios

Clicaremos no link "Add an Action" e escolheremos a opção Communication ➔ Routing Table, conforme exibido Figura 44.

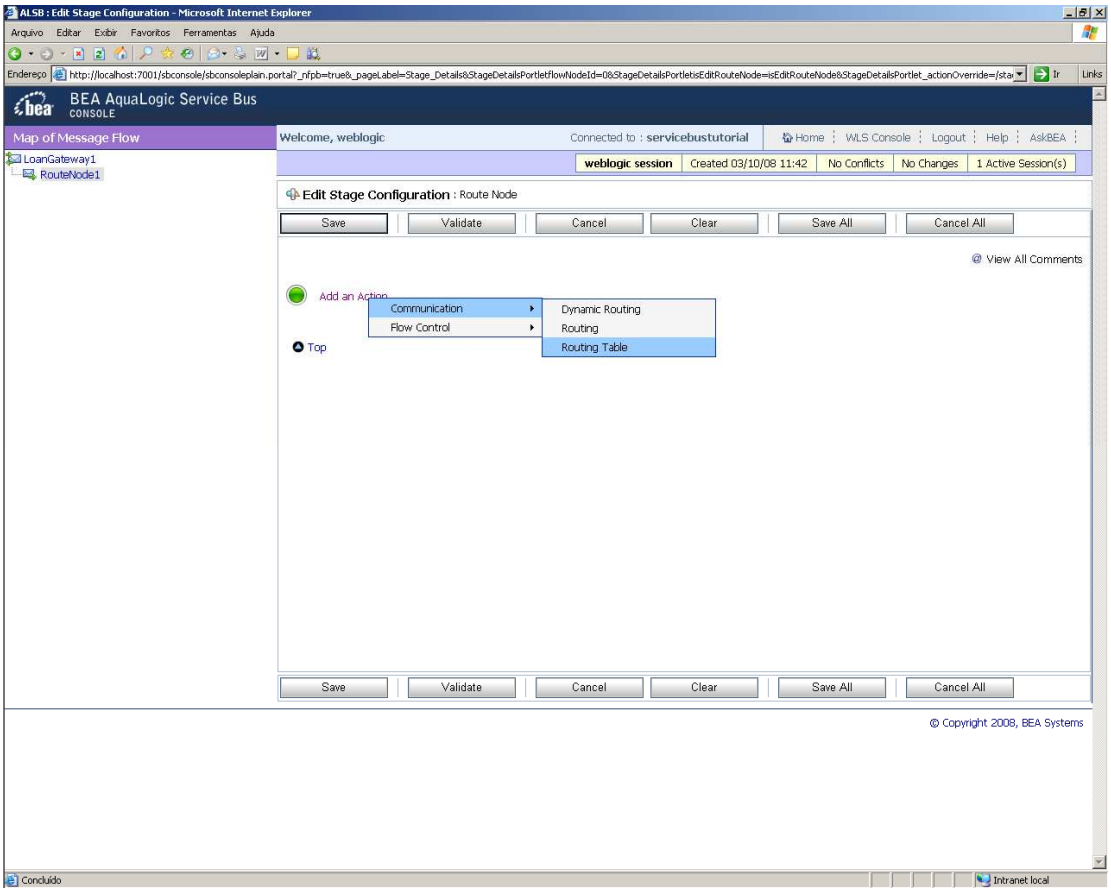

Figura 44 – Opções do link Add an Action

Seremos direcionados para a página de configuração da tabela de roteamento. Essa tabela define quais as regras de roteamento tanto para mensagens de requisição quanto para mensagens de resposta (Figura 45).

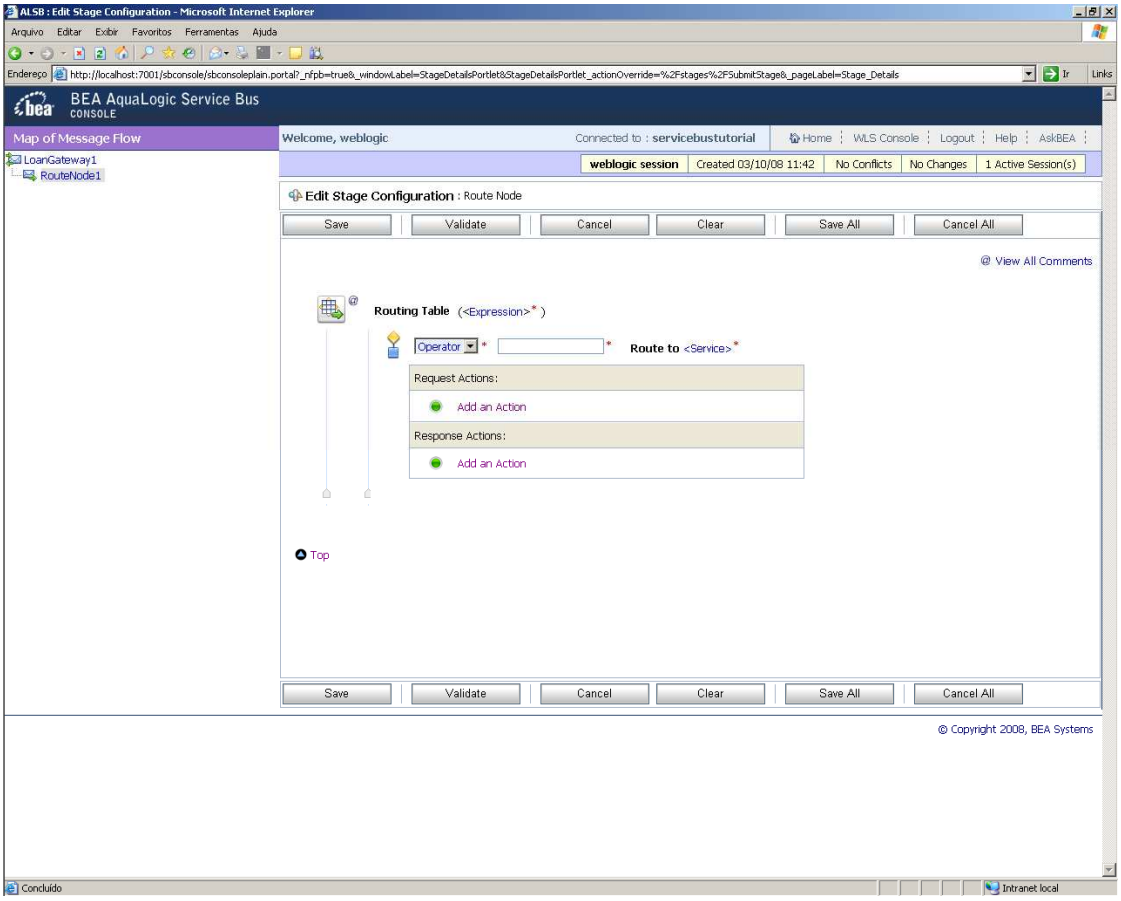

Figura 45 – Tabela de roteamento

Precisamos configurar a tabela de roteamento para rotear mensagens baseado no valor do elemento taxa de juros da mensagem de requisição. Para realizar essa operação, nós podemos configurar esse roteamento baseado no conteúdo da mensagem criando uma expressão XQuery, através do XQuery Expression Editor. Para isso, clicaremos no link <Expression> da tela apresentada na Figura 45 e seremos direcionados para o editor de expressões, conforme apresentado na Figura 46.

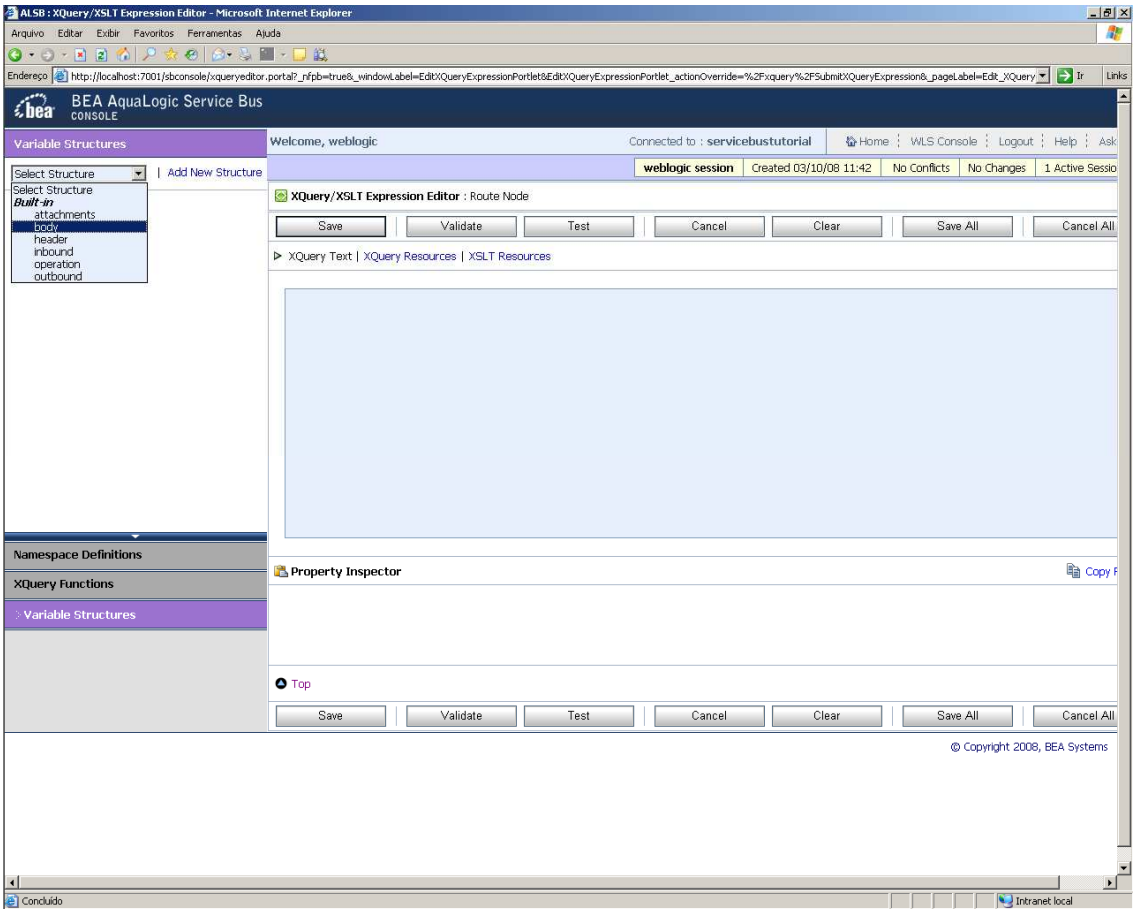

Figura 46 – Editor de expressões XQuery

No editor de expressões, iremos escolher qual parte da estrutura da mensagem SOAP iremos consultar (parte direita da Figura 46). Escolheremos a estrutura Body, que representa o corpo da mensagem SOAP e é onde está a informação da taxa de juros na mensagem (Figura 47). Ao escolheremos essa opção, serão exibidas as mensagens definidas no WSDL. Temos uma mensagem de requisição (loanRequest) e uma mensagem de resposta (loanResponse). Ambas contêm seus atributos da mensagem e, entre eles, o Rate, que é a taxa de juros. Ao arrastarmos o objeto Rate para a área de texto, a expressão que representa esse elemento será adicionada à área. Se clicarmos no botão Validade, podemos verificar se essa expressão está correta e evitamos erros em tempo de execução.

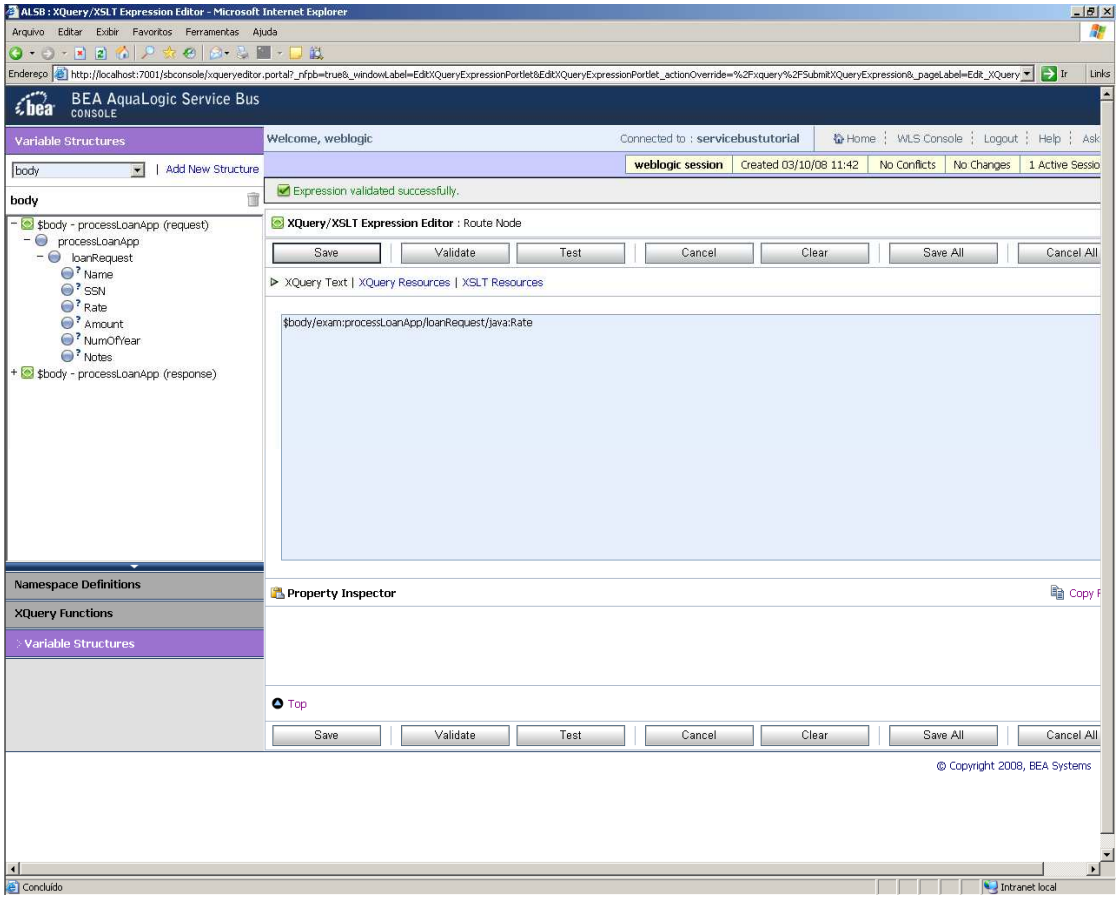

Figura 47 – Estrutura da mensagem loanRequest

Ao clicarmos no botão Save, voltaremos à tela anterior para definir o roteamento que queremos. Então vamos definir que todas as mensagens com taxa de juros abaixo de 5% serão roteadas para o serviço managerLoanReview. Assim, escolheremos a operação "<" e escreveremos o numeral 5 no campo correto, conforme mostra a Figura 48.

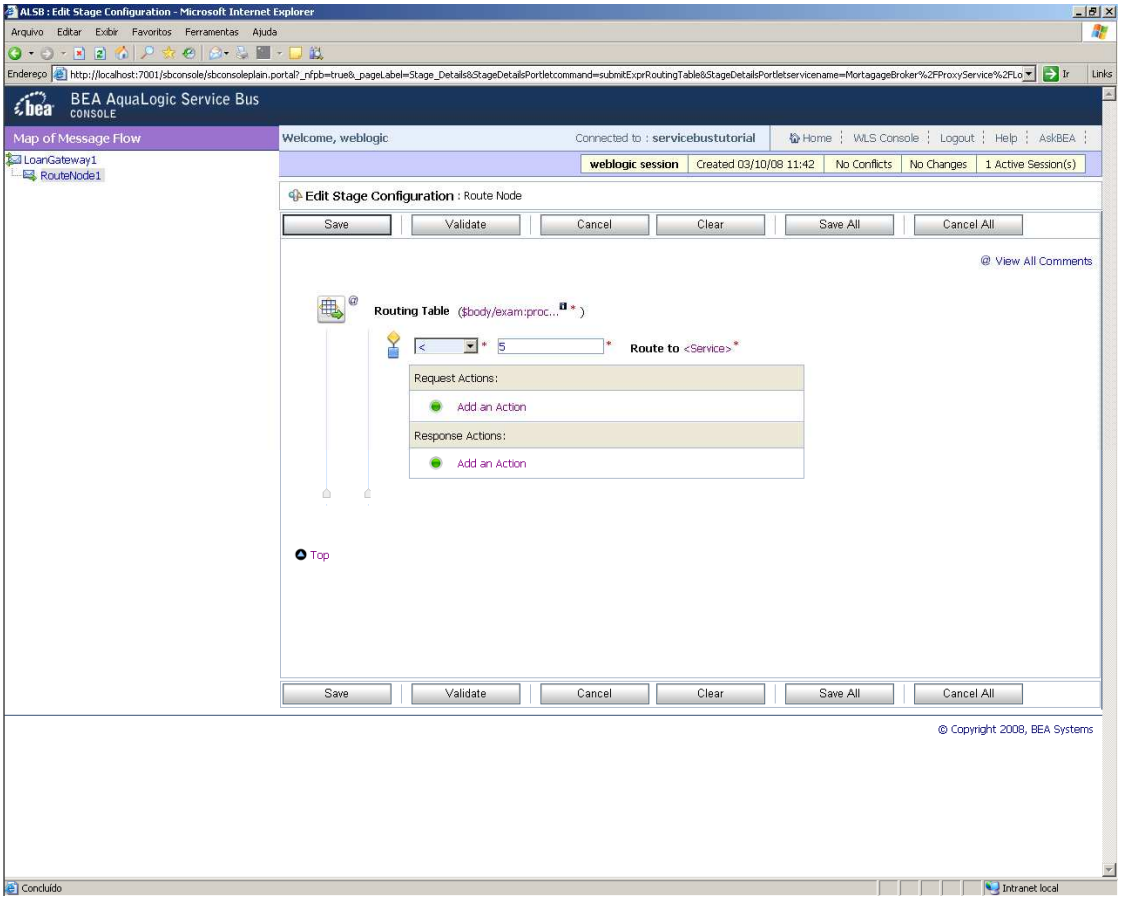

Figura 48 – Definindo roteamento

Para escolhermos o serviço, clicaremos no link "<Service>" (Figura 48) e a lista de serviços de negócio disponíveis aparecerá. Escolheremos o serviço correto e a operação desse serviço que será chamada. No nosso caso, o serviço ManagerLoanReview só possui uma operação, chamada de processLoanApp (Figura 49).

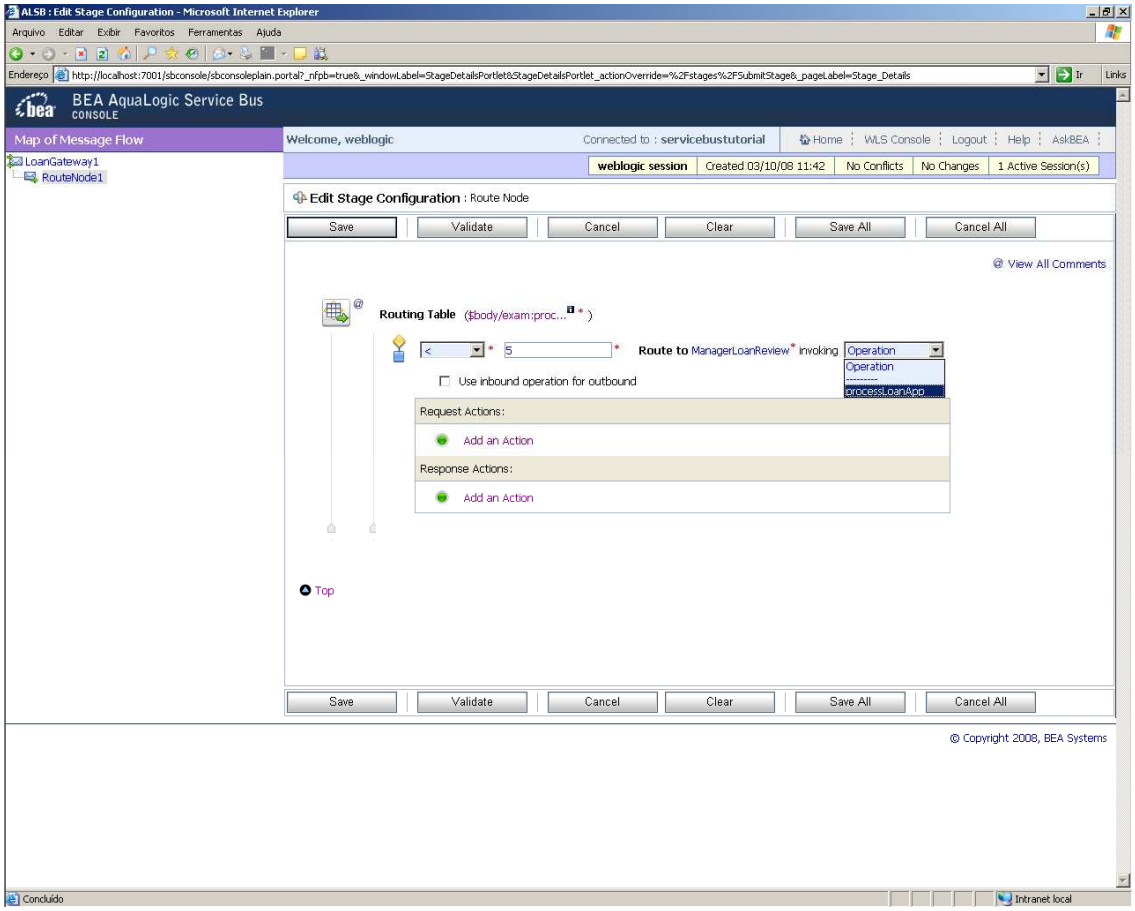

Figura 49 – Definindo o serviço que será chamado pelo roteamento

Ao clicarmos em Save, aceitamos esse roteamento e será finalizado o processo. Nosso Proxy já está criado e, no momento, toda mensagem que chega com taxa de juros menor que 5% será roteada para o serviço ManagerLoanReview. Caso a mensagem contenha uma taxa de juros maior que 5%, será roteada para o serviço no qual o ServiceProxy foi baseado, normalLoan.

A.1.2.4 Teste de aplicações disponíveis no ALSB

O barramento ALSB permite testar as aplicações disponíveis. Neste documento, testaremos a aplicação que criamos para roteamento de mensagem entre os serviços de empréstimo.

Para testar a aplicação, clicaremos no ícone Launch Test Console. Esse ícone pode ser encontrado na tela do Resource Browser, na frente do nome do serviço, como mostra a Figura 50.

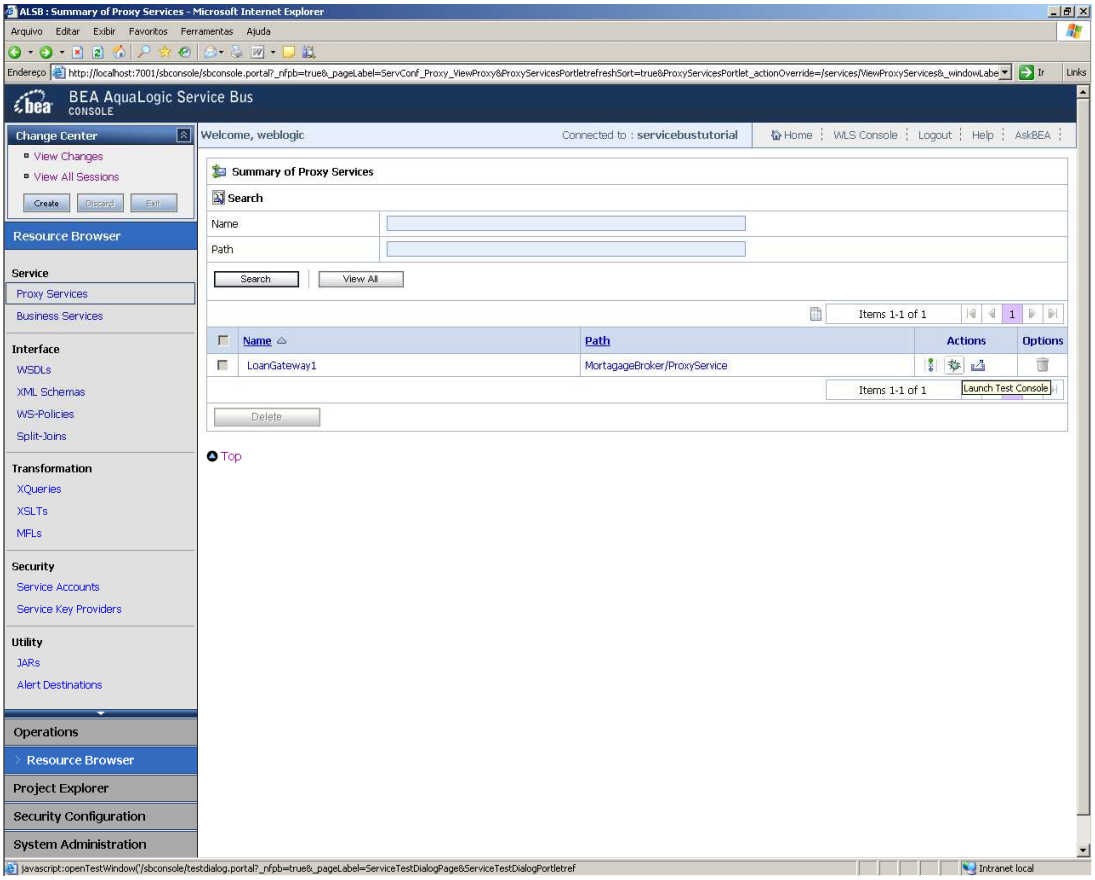

Figura 50 – Iniciando teste da aplicação

Ao clicarmos neste ícone, seremos direcionados para a página de teste da aplicação. A página nos exibe um cógido XML criado automaticamente para teste da mensagem (Figura 51).

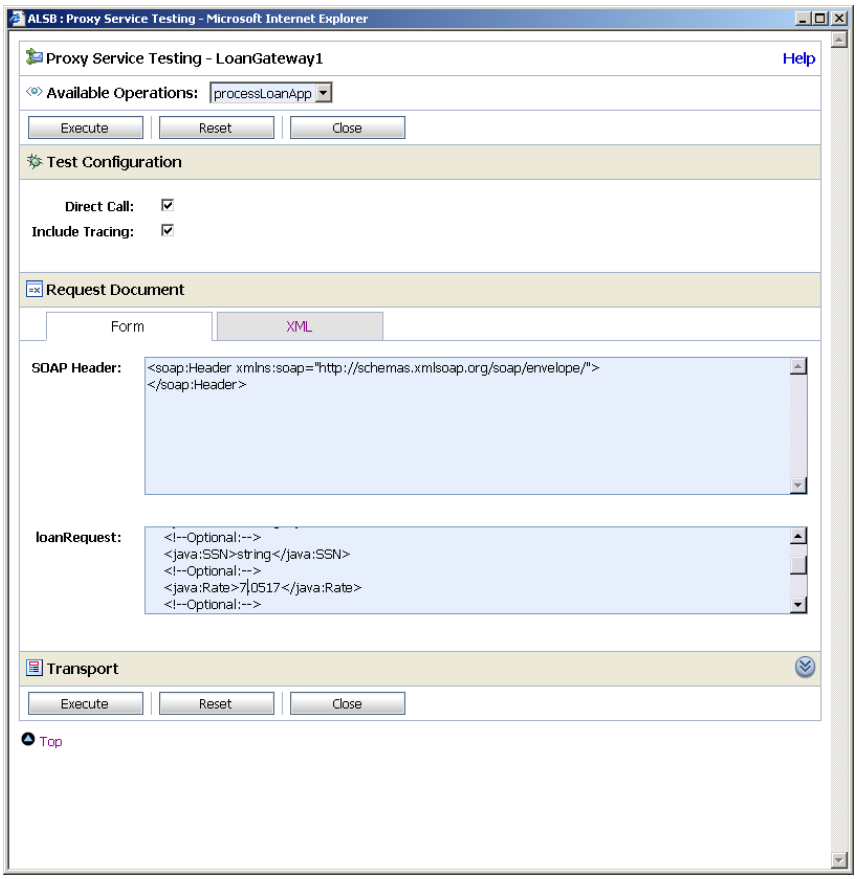

Figura 51 – Tela de teste

No exemplo que foi gerado (Figura 51), a taxa de juros é de 7,0517. Podemos alterar o corpo ou o cabeçalho da mensagem. Porém, os deixaremos intactos. Executando a aplicação com essa mensagem, espera-se que o gateway repasse a mensagem para o serviço normalLoan, uma vez que a taxa de juros é maior que 5. Ao clicarmos em Execute, somos direcionados para a tela apresentada na Figura 52, com a mensagem de resposta.

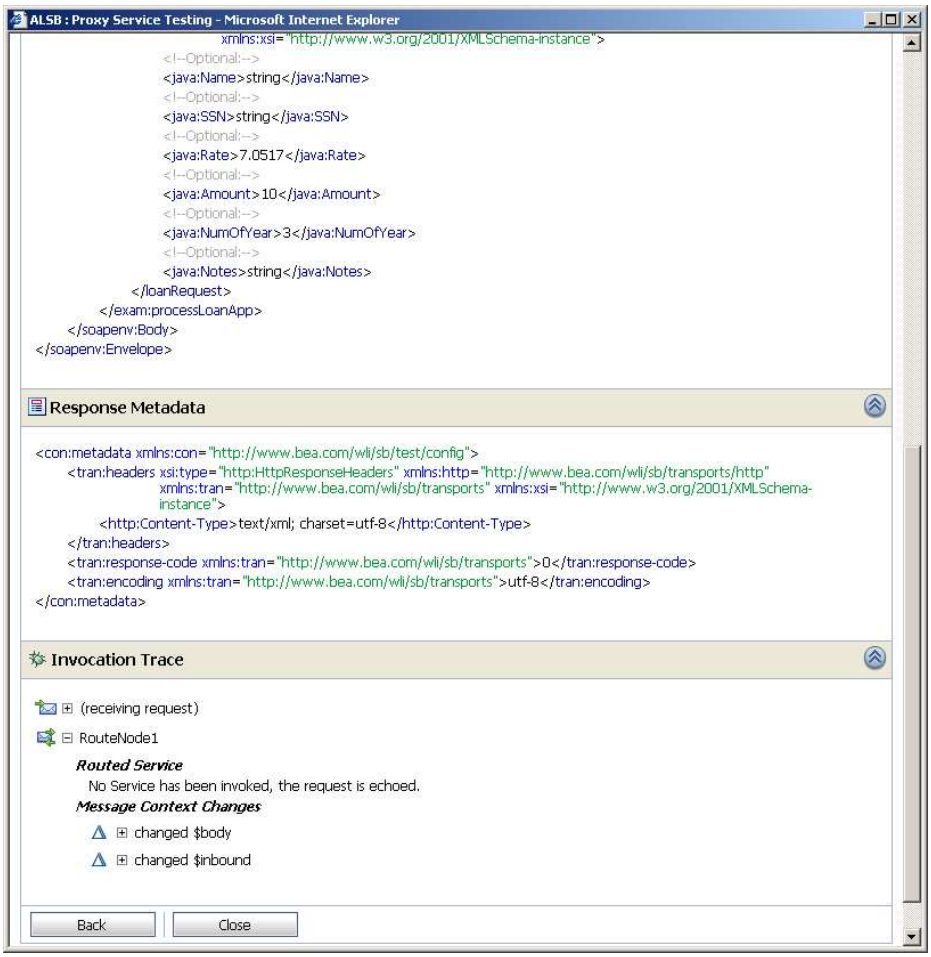

Figura 52 – Resposta com taxa de juros maior que 5%

A tela exibe a mensagem de requisição, a mensagem de resposta e os metadados da resposta recebida. O mais importante para o nosso exemplo, contudo, é a última seção desta tela, a qual contém a trilha de invocação. Podemos visualizar que houve um recebimento de mensagem (receiving request) e, após, foi acionado o elemento RouteNode1, que criamos na seção anterior. O serviço de roteamento, contudo, não foi acionado ("No service has been invoked, the request is echoed"). Uma vez que o roteamento que criamos não foi acionado, a mensagem foi enviada, então, para o serviço normalLoan, que é o caminho padrão para as mensagens.

Iremos, agora, repetir esse teste, porém trocando a taxa de juros da mensagem para o valor 2. Neste caso, esperamos que o roteamento seja acionado para que o pedido de empréstimo seja aprovado. A tela de resposta deste teste é apresentada na Figura 53. Neste caso, após o recebimento da mensagem de requisição, o serviço de roteamento RouteNode1 roteou a mensagem para o serviço ManagerLoanReview. De fato, ao verificarmos o campo Notes da mensagem de resposta (veja o primeiro trecho de XML da figura acima), consta o texto de aprovação do serviço ManagerLoanReview.

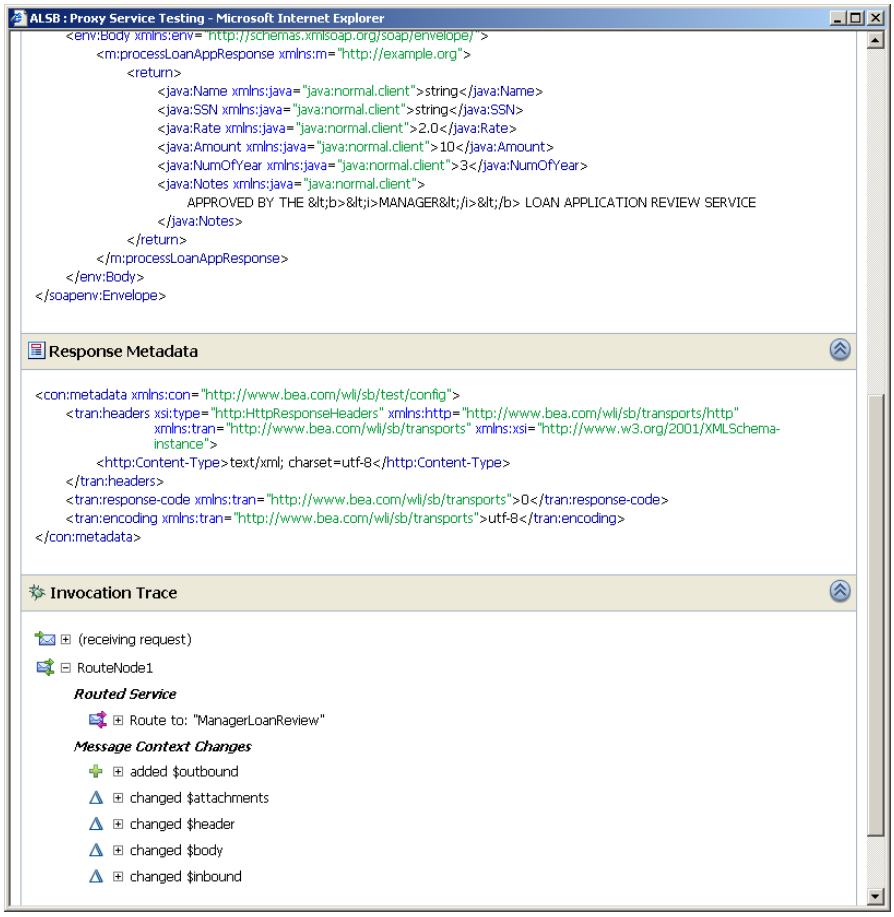

Figura 53 – Resposta com taxa de juros menor que 5%

Anexo 2 - Monitoramento de serviços no ESB

O barramento de serviços da AquaLogic permite o monitoramento dos serviços publicados ou definidos no barramento. Este monitoramento se dá através da criação de regras de alertas no barramento. Além dessas regras, é possível monitorar o estado de outros servidores que fazem parte da arquitetura de barramentos (caso o barramento faça parte de um *cluster*).

O monitoramento se dá para cada serviço disponível no barramento. Assim, para cada serviço que se queira monitorar, é necessário seguir alguns passos para ativar o monitoramento. Por padrão, o monitoramento está desligado para cada serviço publicado. Vale lembrar que, quanto mais monitoramentos forem definidos no barramento, maior pode ser o *overhead* na trasmissão de mensagens entre barramento e terceiros.

Para ativar o monitoramento, basta selecionar um serviço qualquer disponível no barramento e clicarmos em Monitoring ➔ Enabled. É possível alterarmos o intervalo de agregação. Esse intervalo corresponde ao intervalo em que as estatísticas do serviço serão agregadas no barramento (Figura 54).

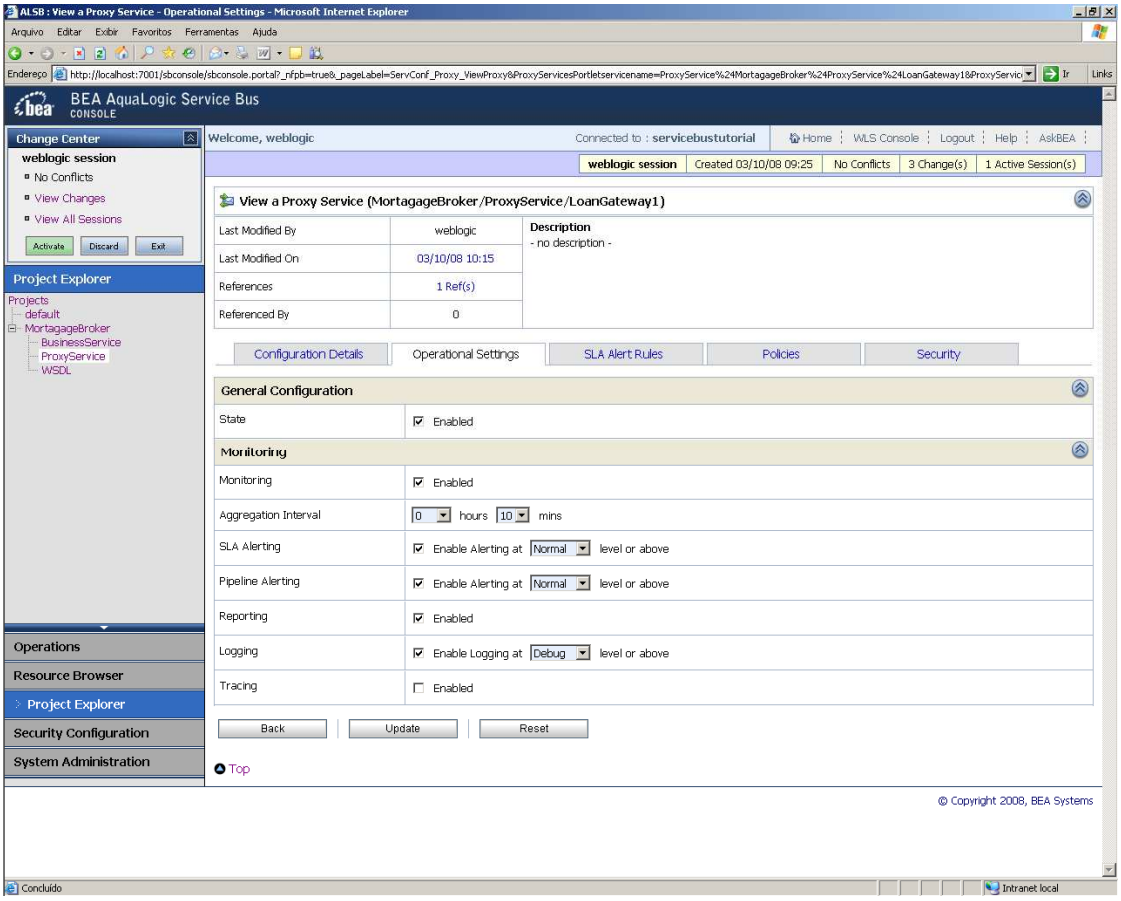

Figura 54 – Ativando o monitoramento do serviço

A criação das regras de alerta é feita na aba SLA Alert Rules, opção New Rule. As regras são criadas para projetos específicos criados no barramento. Cada regra de alerta é associada a um serviço existente no projeto, conforme mostra a Figura 55. Nesta figura, exemplificamos a criação de regra para monitorar o tempo de resposta do elemento RouteNode1, criado na seção A.1.2.3.4 deste relatório. Caso o elemento tenha um tempo médio de resposta maior que 100 milisegundos, será enviada uma mensagem de alerta para o barramento.

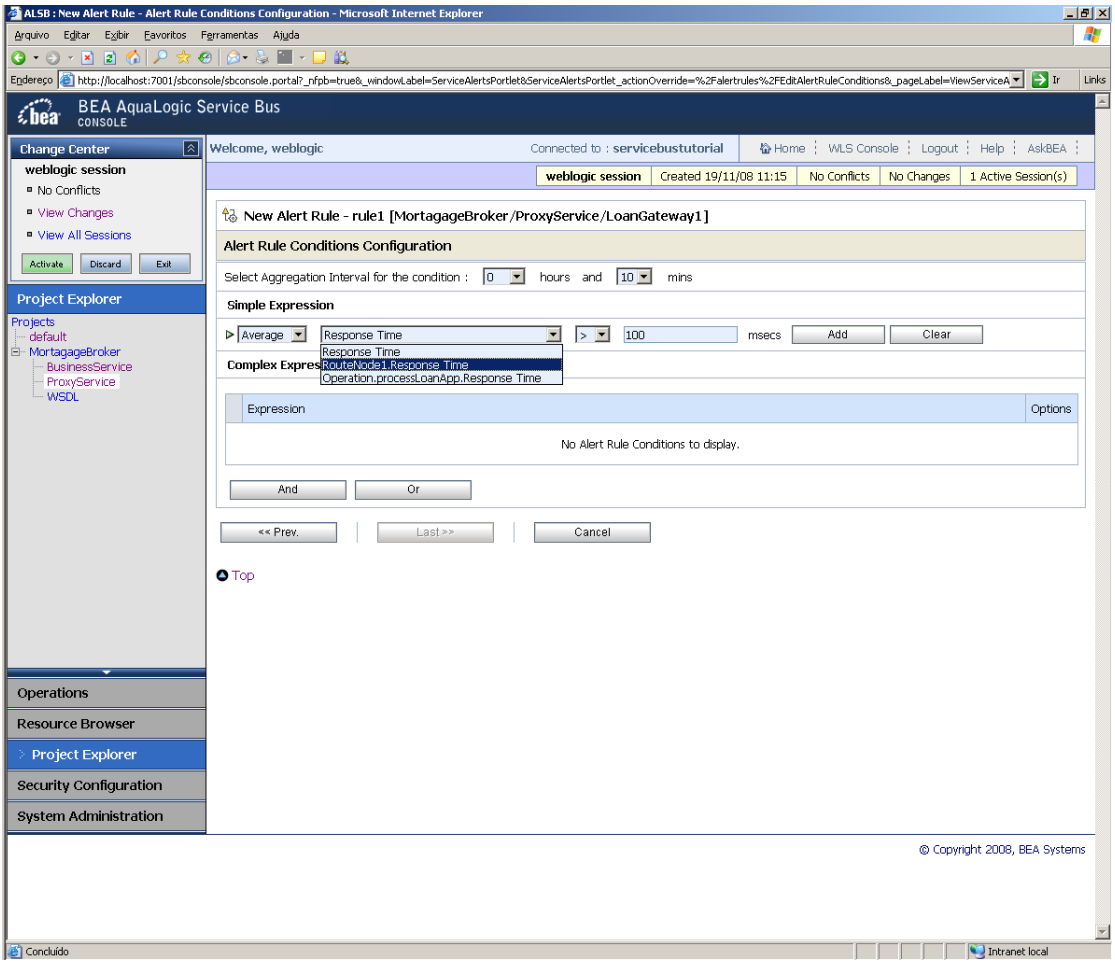

Figura 55 – Criando regra de alerta SLA

O barramento coleta as mensagens de alerta recebidas e as exibe em uma página chamada de *dashboard* (Figura 56). Essa tela permite uma visão panorâmica dos erros existentes nos serviços expostos no barramento e demais servidores existentes na arquitetura.

No painel "Service Summary" é possível visualizar a porcentagem de mensagens de alerta por tipo de alerta (*warnings*, *normal*, *minor*, *major*, etc) e os serviços que estão lançando a maioria dos alertas no intervalo de agregação mais recente.

Os detalhamentos dessas mensagens de alerta podem ser visualizados no quadro "Alert Summary", onde estão, em ordem cronológica, as mensagens de alerta recebidas. Ao clicar na regra de alerta que lançou a mensagem é possível visualizar qual o motivo do recebimento da mensagem de alerta no barramento, pelo análise do alerta definido pelo serviço (vide Figura 55).

Por fim, o quadro "Server Summary" exibe as informações sobre o estado dos servidores existentes no *cluster*. Na Figura 56, o gráfico de pizza está todo verde indicando que 100% dos servidores estão em funcionamento (*alive*). Este é um caso simples, pois o barramento está executando em somente um servidor.

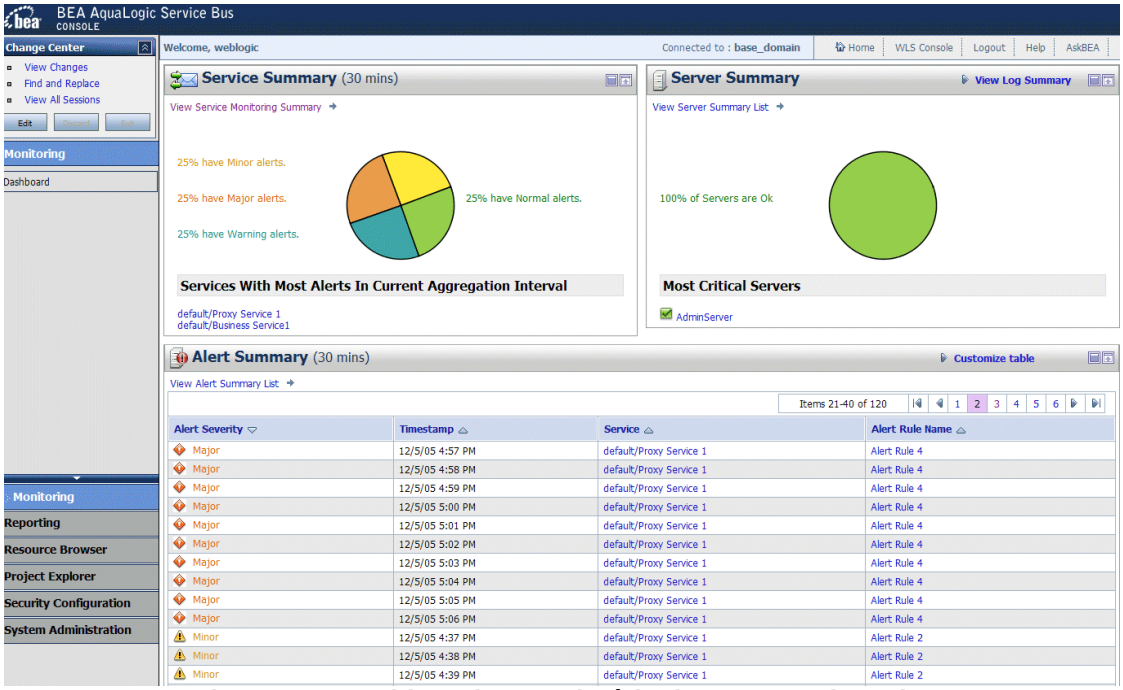

Figura 56 – Dashboard com relatório de erros e avisos do ESB

Anexo 3 - AquaLogic Service Registry

A.1.3 Instalação do AquaLogic Service Registry

O AquaLogic Service Registry (também conhecido como ALSR) está disponibilizado para *download* em [ALSR, 2008]. Este é um repositório UDDI [UDDI, 2004], ou seja, é responsável por armazenar as interfaces dos serviços disponíveis na corporação e fornecer meios para consultar as interfaces armazenadas.

O ALSR necessita do JDK (Java Development Kit), uma vez que parte da aplicação é web e é necessário compilar páginas JSP em tempo de execução, não sendo suficiente somente a JRE (Java Runtime Environment). A Figura 56 mostra a tela inicial de instalação da ferramenta.

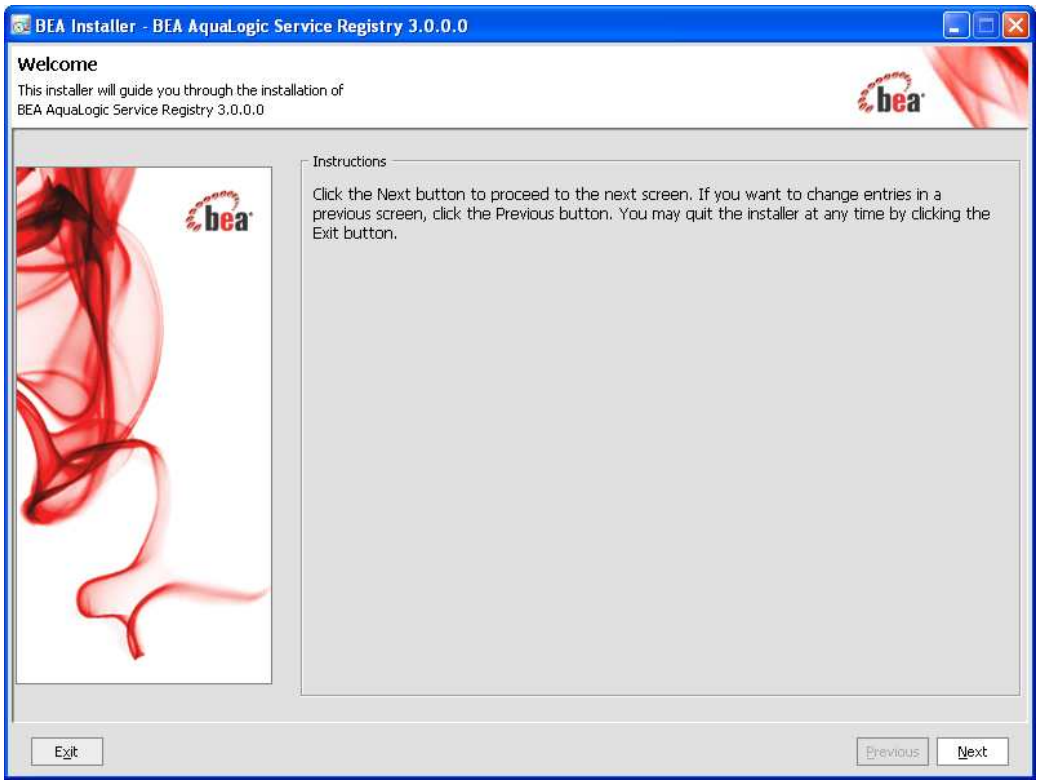

Figura 57 – Tela inicial de instalação do ALSR

As telas seguintes do assistente de instalação permitem selecionar o diretório onde será instalada a ferramenta, os atalhos que estarão disponíveis no sistema operacional, a senha do usuário administrador do repositório e outras opções comuns de várias ferramentas. Por serem opções comuns, não mostraremos essas telas.

O ALSR é uma aplicação web e, assim, necessita de um servidor de aplicação. No ato da instalação, é possível escolher o WLS como servidor de aplicação do ALSR ou um servidor HTTP/HTTPS embutido no ALSR é pré-configurado, como mostra a Figura 58.

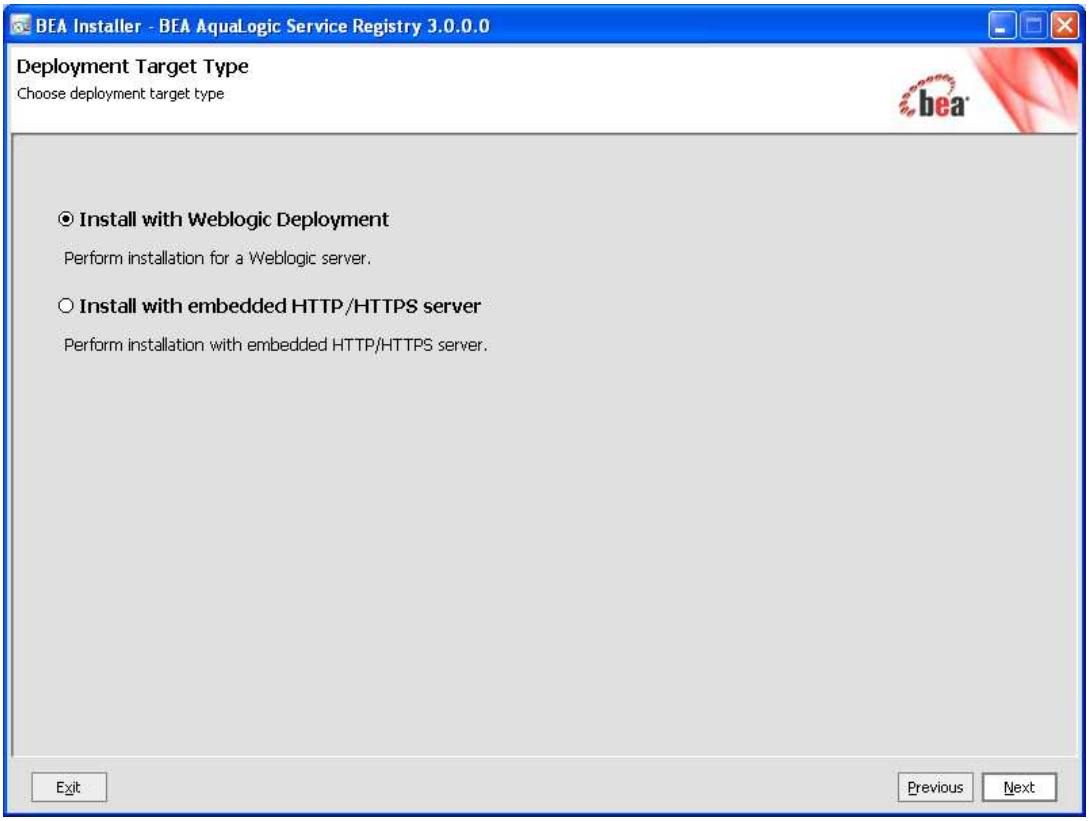

Figura 58 – Tela com opção de servidores de aplicação para o ALSR

O ALSR permite escolher cinco tipos de instalação, conforme mostra a Figura 59:

- *Standalone:* Essa é a instalação padrão, a qual permite instalar um repositório e criar uma nova base de dados.
- *Standalone registry with data integration:* Usado para migrar dados de um repositório de versão anterior para essa nova versão (3.0) do ALSR.
- *Publication registry:* Esse servidor é usado para publicação de registros para fins de teste e verificação dos dados antes de serem aprovados.
- *Discovery Registry:* Este tipo de servidor contém os dados aprovados previamente postados em um *Publication Registry*. Este servidor é controlado por usuários que possuem papel de aprovadores.
- *Intermediate registry:* Esse tipo de repositório faz o papel tanto de repositório de publicação quanto de repositório de pesquisa (*Discovery registry*). É utilizado quando o processo de aprovação dos dados dos serviços necessita ser realizado em mais de uma atividade de aprovação. Para tal, é necessário que o *Discovery Registry* esteja previamente instalado.

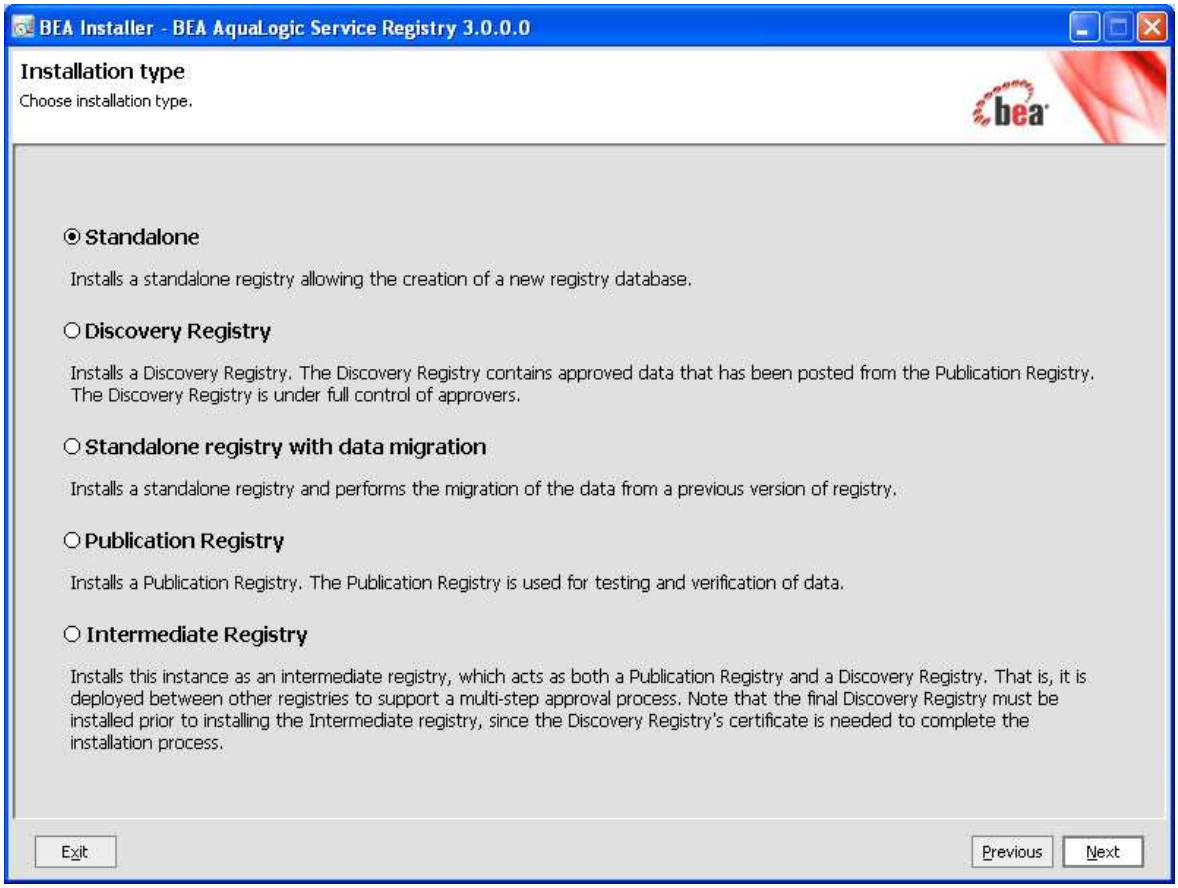

Figura 59 – Tela com tipos de instalação do ALSR

Estes tipos de repositório existem para compor processos de aprovação. Usando um único repositório (*standalone*), os dados dos serviços serão publicados neste servidor e prontos para serem acessados por aplicações ou usuários, sendo este o cenário mais simples. Um cenário mais complexo, e mais controlado do ponto de vista gerencial, é um processo de aprovação onde os dados dos serviços são publicados em um repositório (*Publication registry*) e, ao serem aprovados, são migrados para um repositório específico para interação com usuário e aplicações (*Discovery registry*), como ilustra a Figura 60.

Figura 60 – Processo de aprovação

O *Discovery registry* pode também receber dados aprovados de diferentes repositórios de publicação, como mostra a Figura 61.

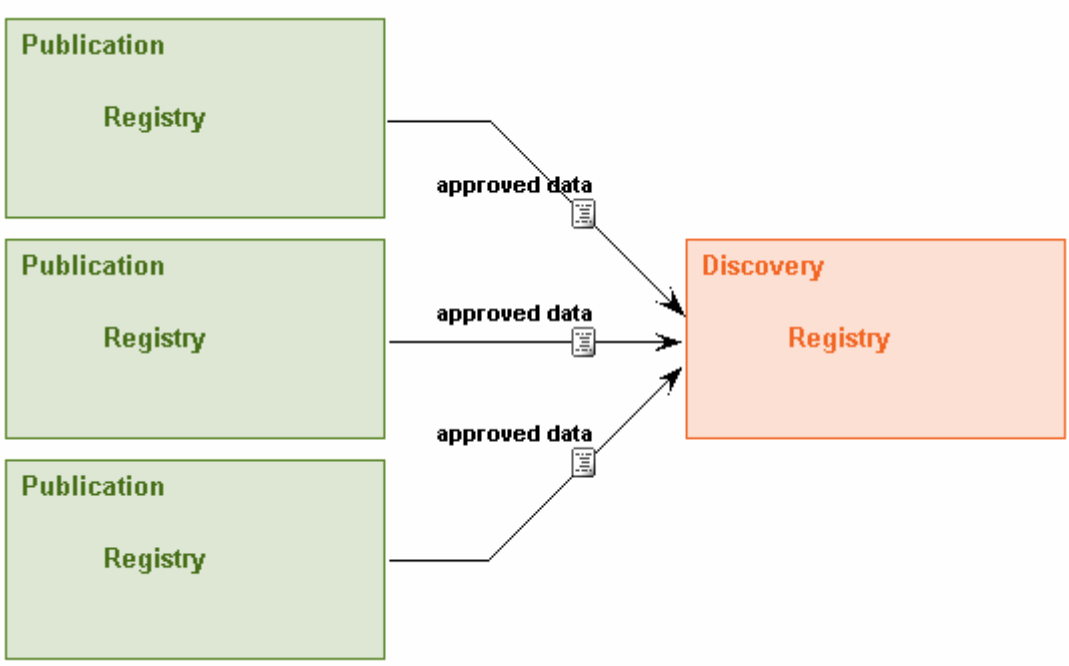

Figura 61 – Processo de aprovação com vários repositórios de publicação

Um cenário mais complexo é a aprovação em dois passos, o qual utiliza um repositório intermediário para receber os dados aprovados do repositório de publicação e aguarda uma segunda aprovação desses dados antes de serem disponibilizados no *Discovery registry*. Essa abordagem está ilustrada na Figura 62.

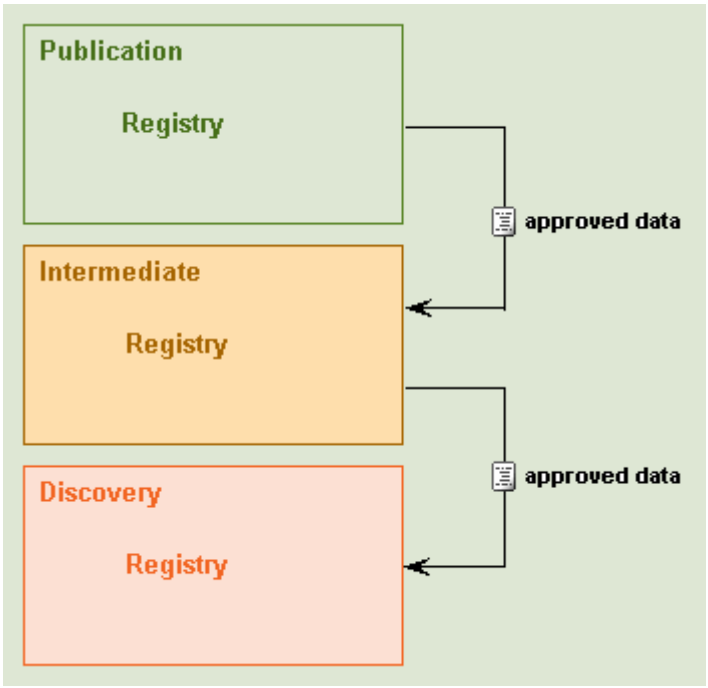

Figura 62 – Processo de aprovação em dois passos

Para cada passo de qualquer um dos processos escolhidos, o usuário que enviou os dados para serem publicados e/ou aprovados pode receber notificações via email sobre a mudança de estado dos dados, uma vez configurado o servidor SMTP na instalação do ALSR, como mostra a Figura 63.

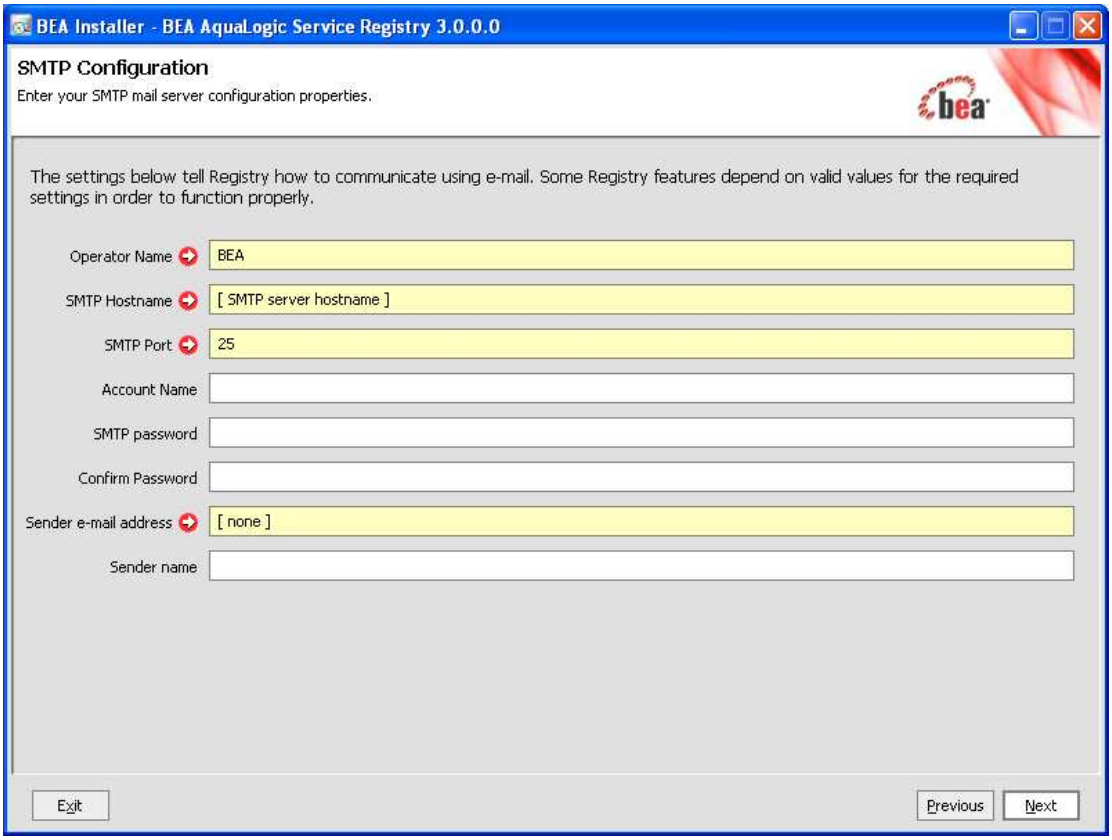

Figura 63 – Configuração do servidor de SMTP

Os dados postados ou aprovados serão armazenados em um banco de dados, o qual será escolhido durante a instalação, como mostrado na Figura 64. O ALSR criará um esquema para o banco de dados escolhido. No ambiente de teste vamos utilizar o banco de dados HSQL, por ser de fácil manipulação e não necessitar de instalação com direitos de administrador da máquina.

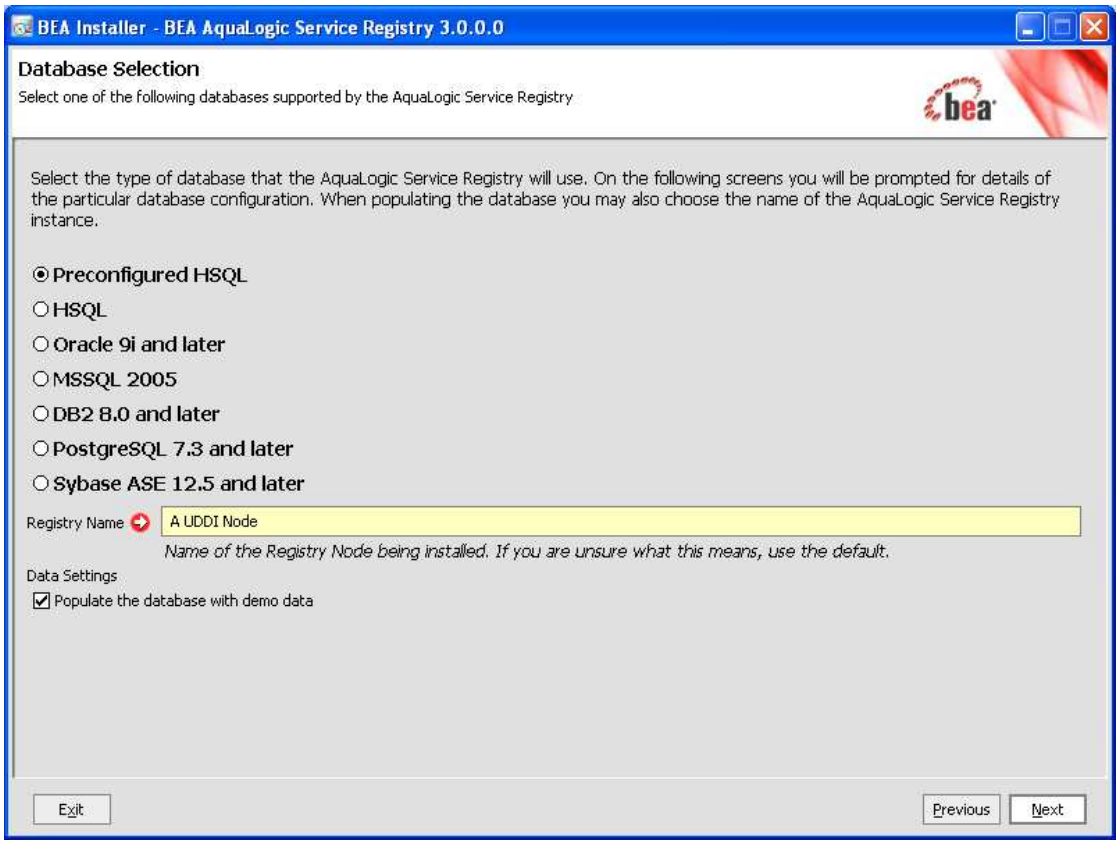

Figura 64 – Configuração do banco de dados que armazenará os registros

Por fim, é necessário configurar o provedor de autenticação dos usuários, como mostra a Figura 65. O provedor de autenticação pode ser o próprio banco de dados do repositório, um servidor LDAP ou um repositório externo que será acessado através de uma API.

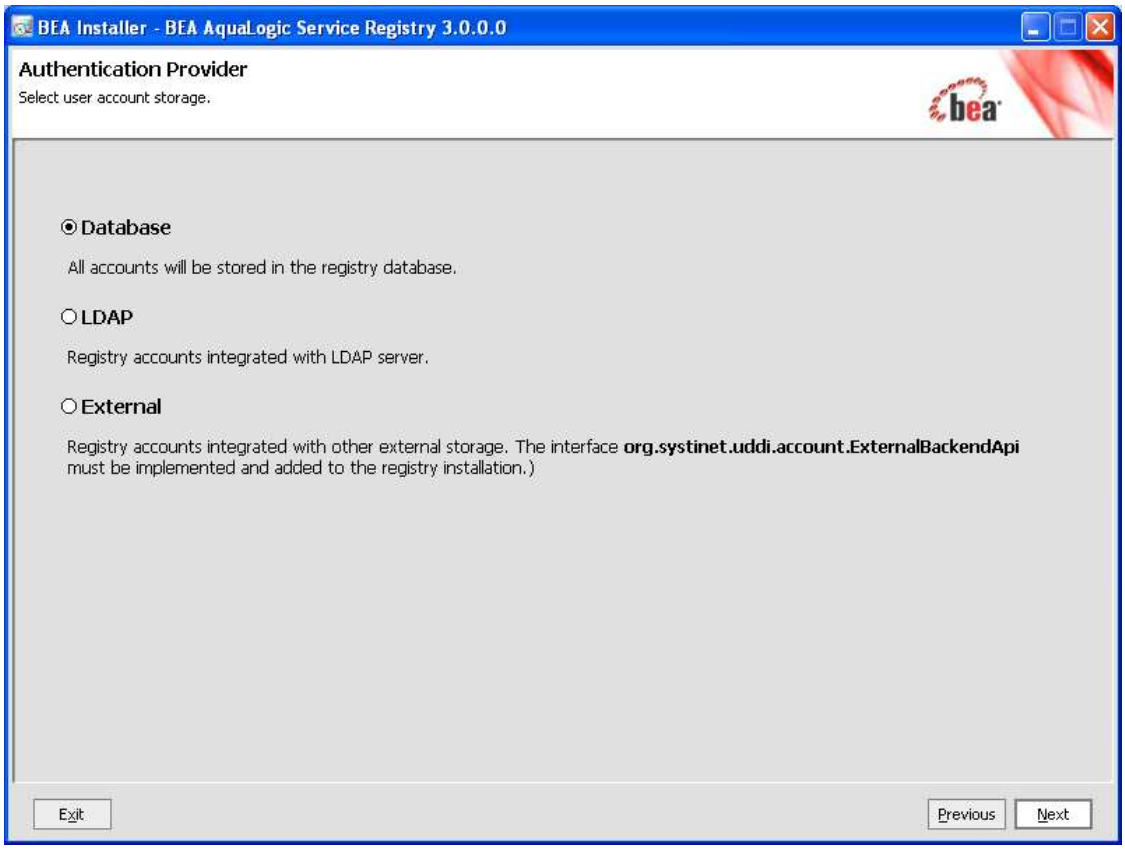

Figura 65 – Configuração do provedor de autenticação de usuários

A.1.4 Utilização do Registry

O ALSR possui duas interfaces para usuários distintos. Uma primeira interface, chamada de Business Service Console (Figura 66), é o portal para manipulação do conteúdo do repositório. Essa interface é utilizada por usuários cadastrados pelo administrador do repositório e não permite alterações das configurações do repositório. Entre as principais opções dessa interface, estão as opções para percorrer o catálogo de serviços e gerar relatórios, além da publicação de serviços web para o repositório. Essa publicação segue os esquemas discutidos na seção anterior, necessitando de aprovação do administrador do repositório caso existam mais de uma instância do ALSR definido.

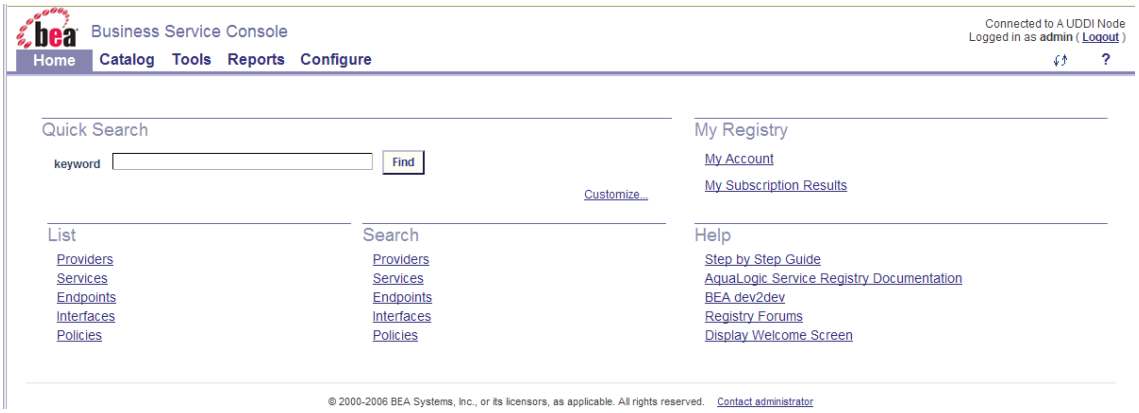

Figura 66 – Tela principal do Business Service Console

A segunda interface para manipulação do repositório é chamada de Registry Console (Figura 67) e é utilizada pelo usuário administrador para configurar o repositório ou aprovar definições de serviços enviadas para publicação, além de outras tarefas de administração.

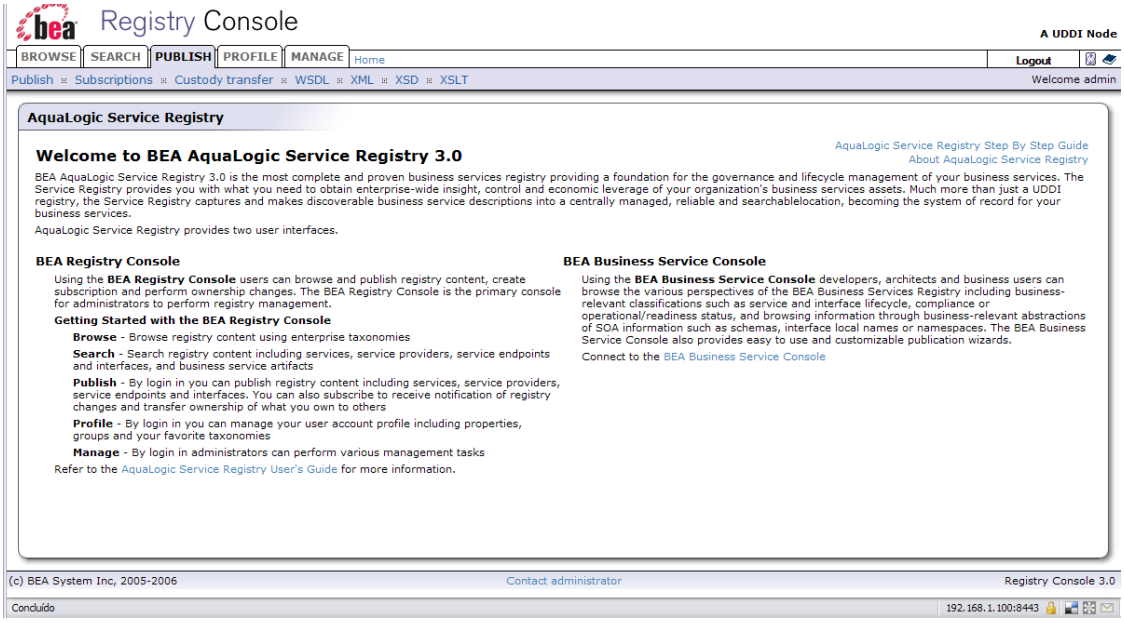

Figura 67 – Tela principal do Registry Console

O ALSR é um repositório que permite armazenar não somente a definição de serviços web, isto é, arquivos WSDL, mas também os artefatos necessários para a comunicação com esses serviços, como arquivos XSLT, tipos definidos em XML Schemas e outros arquivos em formato XML. Esses artefatos podem ser consultados através da geração de relatórios. A Figura 68 mostra os tipos de relatórios existentes na ferramenta. Entre eles estão relatórios de uso de elementos do registro, disponibilidade de serviços ou pontos de acesso (endpoints) específicos, estado de uma interface de serviço, etc. Esses relatórios são classificações disponibilizadas pelo ALSR, ou seja, o resultado visual dos relatórios é o mesmo, apresentando somente os serviços que se adequam à classificação escolhida.

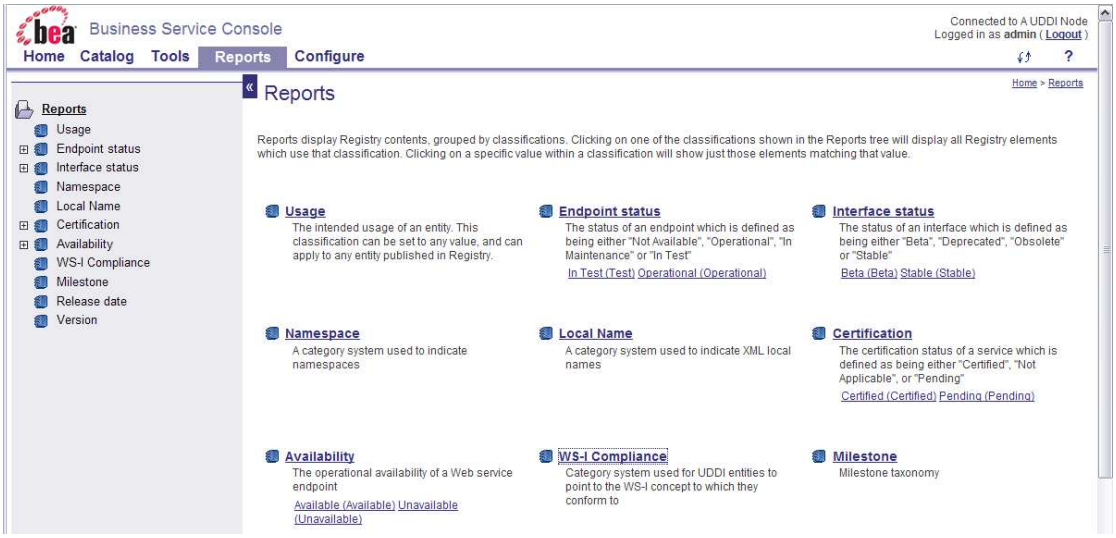

Figura 68 – Tipos de relatório disponíveis para o usuário do Business Service Console

A Figura 69 mostra o resultado de um dos relatórios, o relatório de uso. Neste relatório, os serviços são classificados pela coluna "usage", a qual informa a finalidade de cada serviço do repositório.

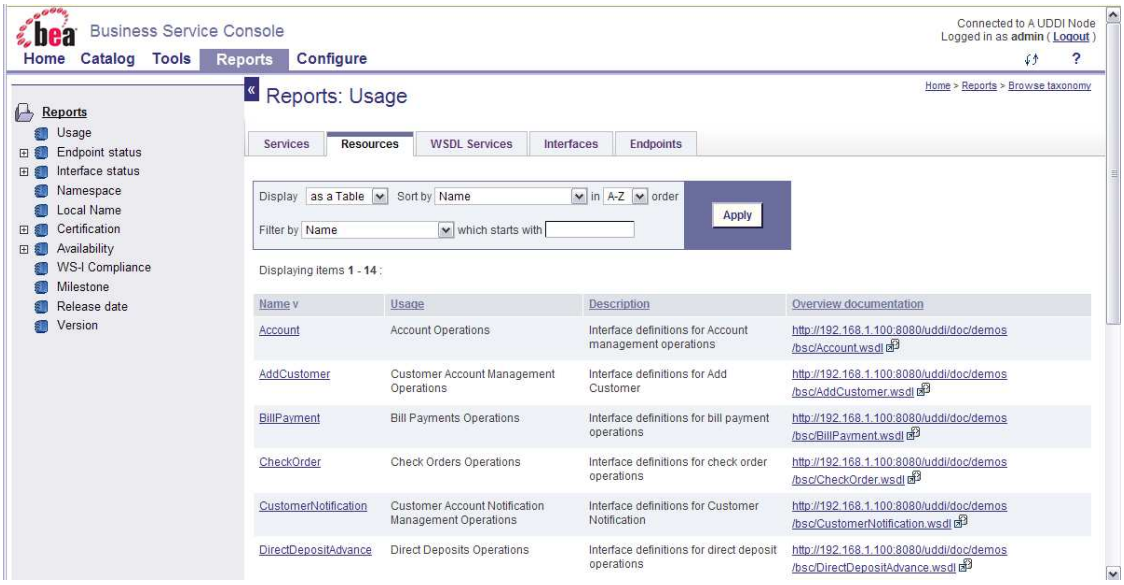

Figura 69 – Resultado de um tipo de relatório

O *Business Service Console* permite navegar nos artefatos do repositório através do catálogo de artefatos, como mostra a Figura 70. O catálogo contém elementos WSDL, pontos de acesso, tranformações XLST, tipos XML Schema, etc. Na tela abaixo é possível também visualizar a opção "Publish a new service" utilizada para enviar uma descrição de serviço para o catálogo. Opções similares são exibidas para cada elemento escolhido na barra lateral do catálogo.

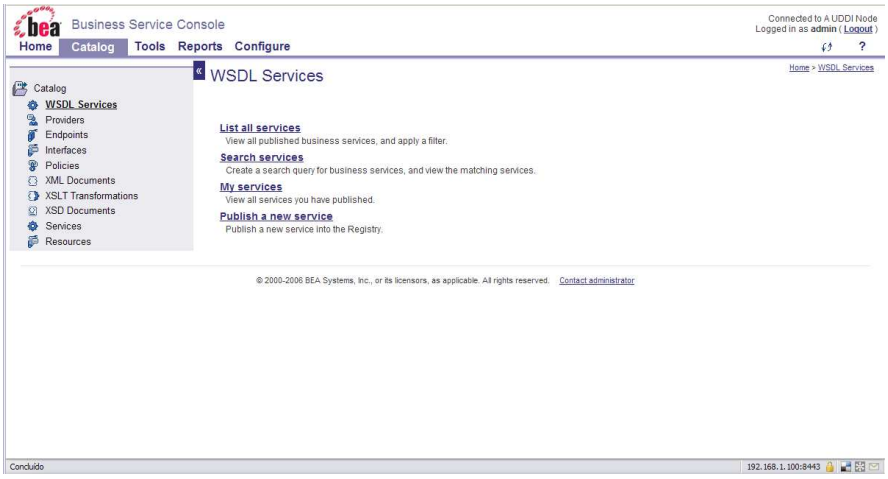

Figura 70 – Visão do catálogo

A Figura 71 mostra a visão do catálogo com a lista de definições WSDL dos serviços.

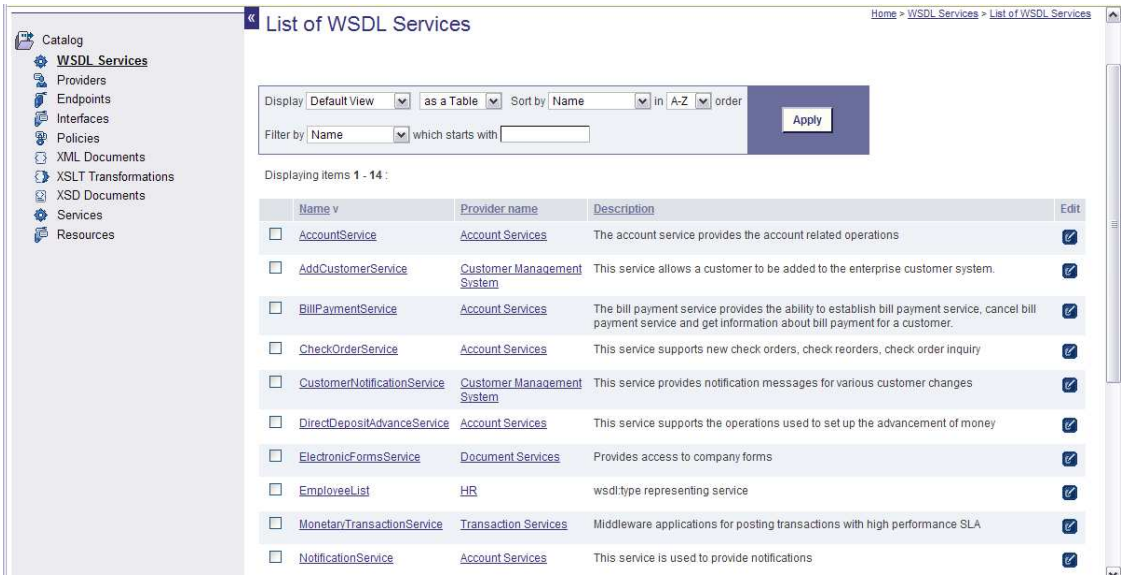

Figura 71 – Visão do catálogo com os WSDL publicados

No Registry Console, o administrador possui uma visão do catálogo de artefatos, porém uma visão mais detalhada visando à manutenção das informações, conforme mostra a Figura 72. A tela mostra informações do artefato, como os identificadores únicos (*service key* e *business key*), nome do serviço, datas de criação e modificação do registro etc.

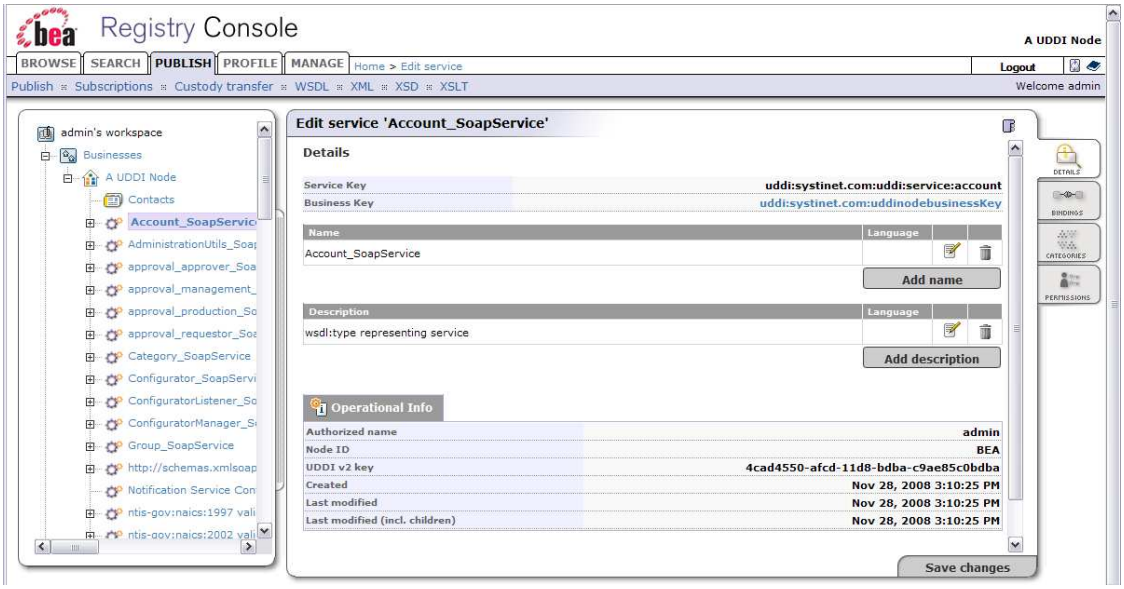

Figura 72 – Tela de gerência dos serviços publicados

O interessante de um repositório de serviços que siga a especificação UDDI é que o repositório permite também associar elementos técnicos, por exemplo, serviço com elementos de negócio, tais como registros de departamentos da empresa e funcionários responsáveis. A Figura 73 apresenta a tela exibindo o que a especificação UDDI chama de *business view*, ou seja, uma visão de registro com dados do negócio que estão associados a um certo número de serviços.

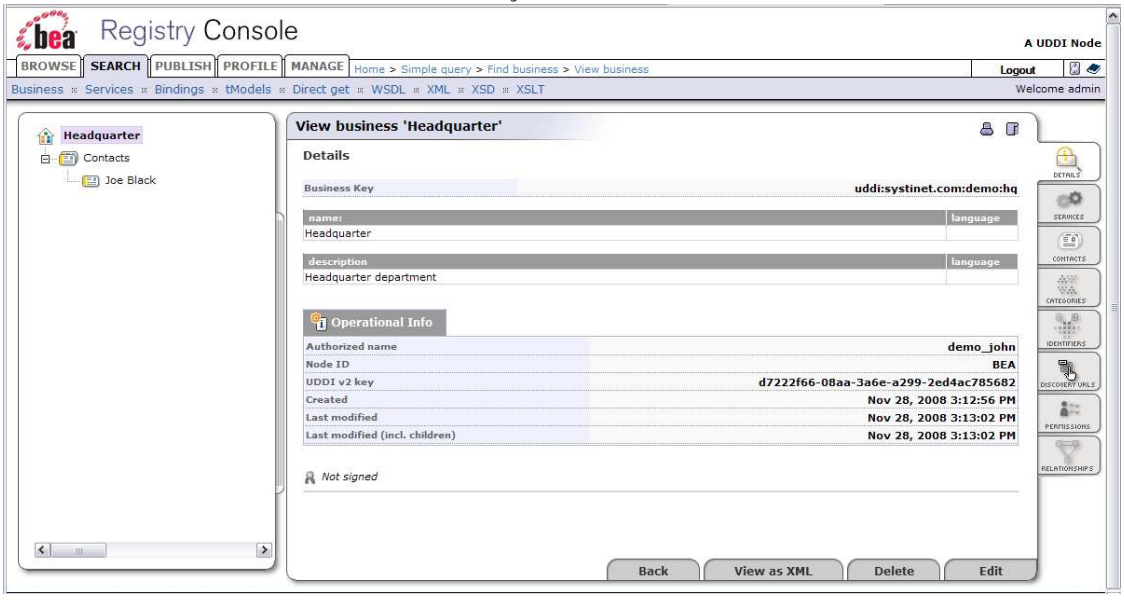

Figura 73 – Visão de negócio

As principais funcionalidades do *Registry Console* estão presentes na aba de gerência (*manage*), como mostra a Figura 74. Nesta aba estão opções para gerência de contas e grupos de usuários, permissões de acesso ao repositório, gerência da taxonomia aplicada aos serviços disponíveis, gerência das chaves primárias dos elementos dos serviços e estatísticas do repositório. Outra opção importante é a gerência de replicação do repositório, que pode ser usada por medida de segurança ou para balancear o acesso ao servidor, evitando a sobrecarga.

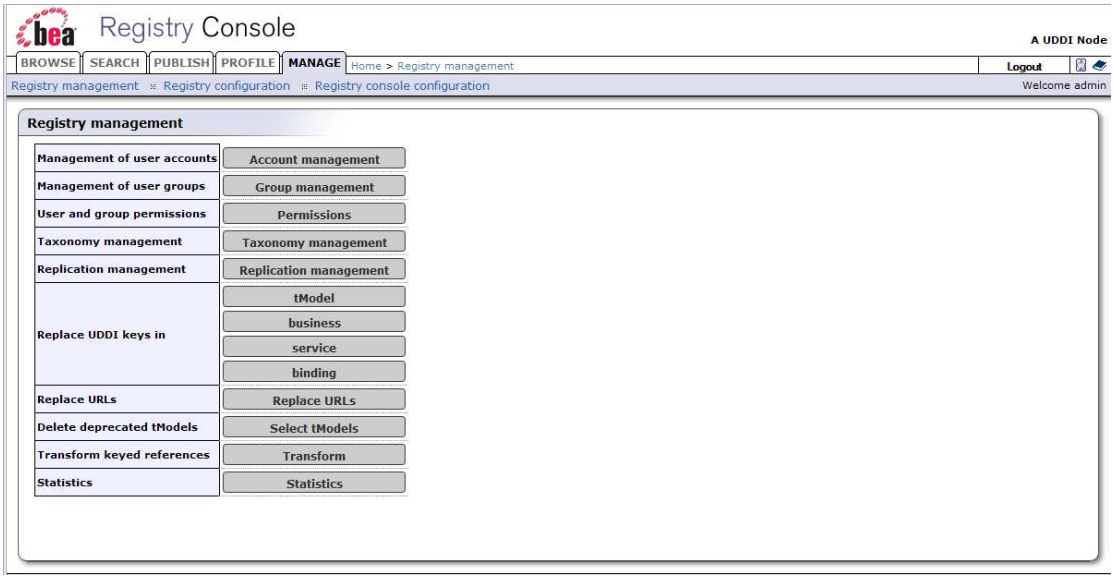

Figura 74 – Opções de gerência do repositório

Anexo 4 - Utilização do barramento com repositório UDDI

O barramento de serviços da BEA, AquaLogic Service Bus, pode se comunicar com um repositório UDDI para publicar ou recuperar serviços, em especial com o repositório ALSR.

A comunicação com o repositório UDDI se dá pela troca de mensagem com dois objetivos: publicar e importar serviços. Os serviços a serem publicados no repositório são os serviços de Proxy, uma vez que são os serviços criados no próprio barramento. Os serviços de negócio, que é o nome dado aos serviços que estão disponibilizados em outros servidores e serão acessados pelo barramento, podem ser importados para o barramento através do repositório UDDI. A Figura 75 resume a comunicação entre o barramento e o repositório UDDI.

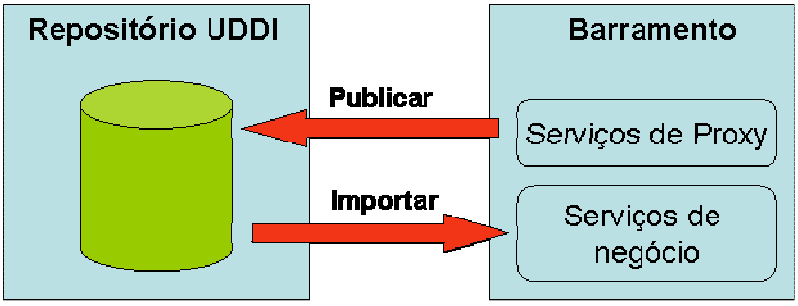

Figura 75 – Comunicação entre repositório UDDI e barramento de serviços

O ALSB permite, inclusive, a utilização de mais de um repositório UDDI. A configuração do repositório UDDI é feita no console do ALSB, no ambiente de administração do sistema (System Administration). A Figura 76 mostra a tela de administração do ALSB com as opções para repositórios UDDI para cadastro de repositórios, importação de serviços, publicação de serviços de Proxy e opções para verificar importações e publicações definidas como automáticas.

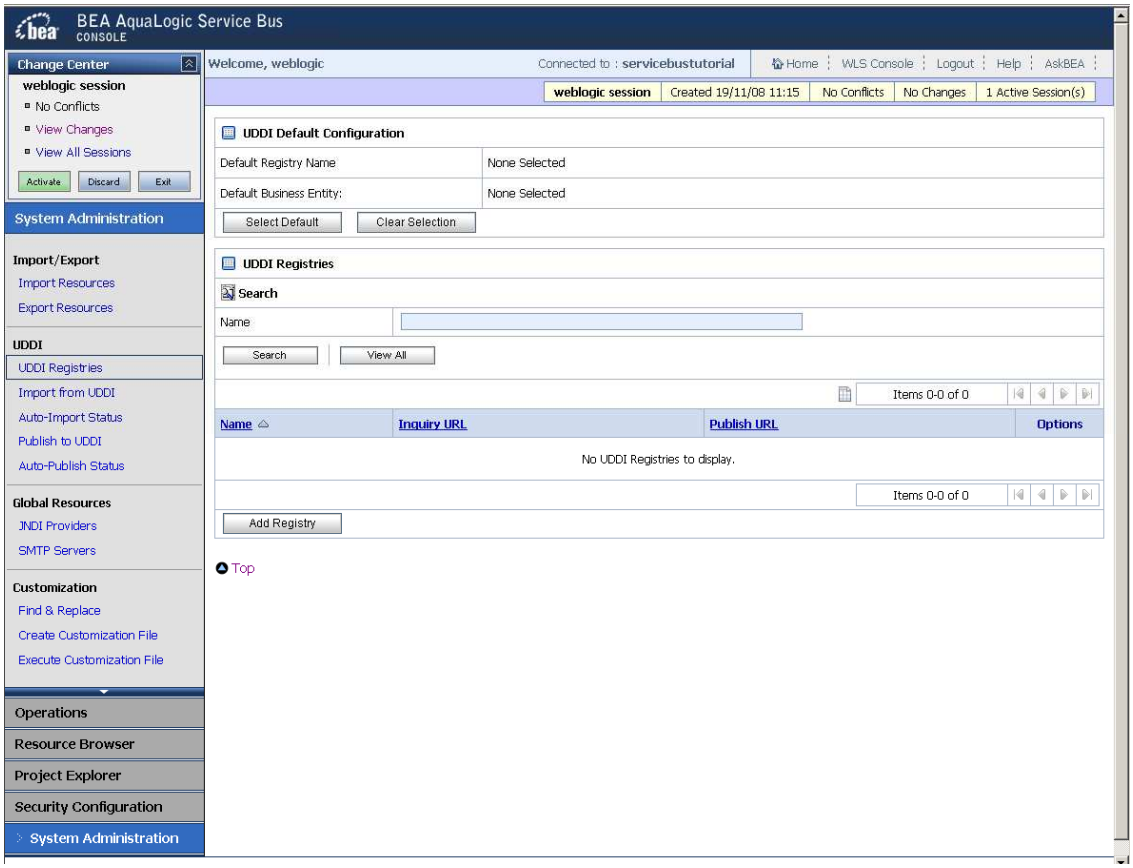

Figura 76 – Tela de admininistração de sistema do ALSB

A Figura 77 apresenta o formulário para cadastrar um repositório UDDI no ALSB. É preciso cadastrar as URIs para consulta, publicação, assinatura e segurança do repositório UDDI. Normalmente esses endereços estão num mesmo host, porém não é uma regra e, assim, é necessária a especificação de cada interface do repositório.

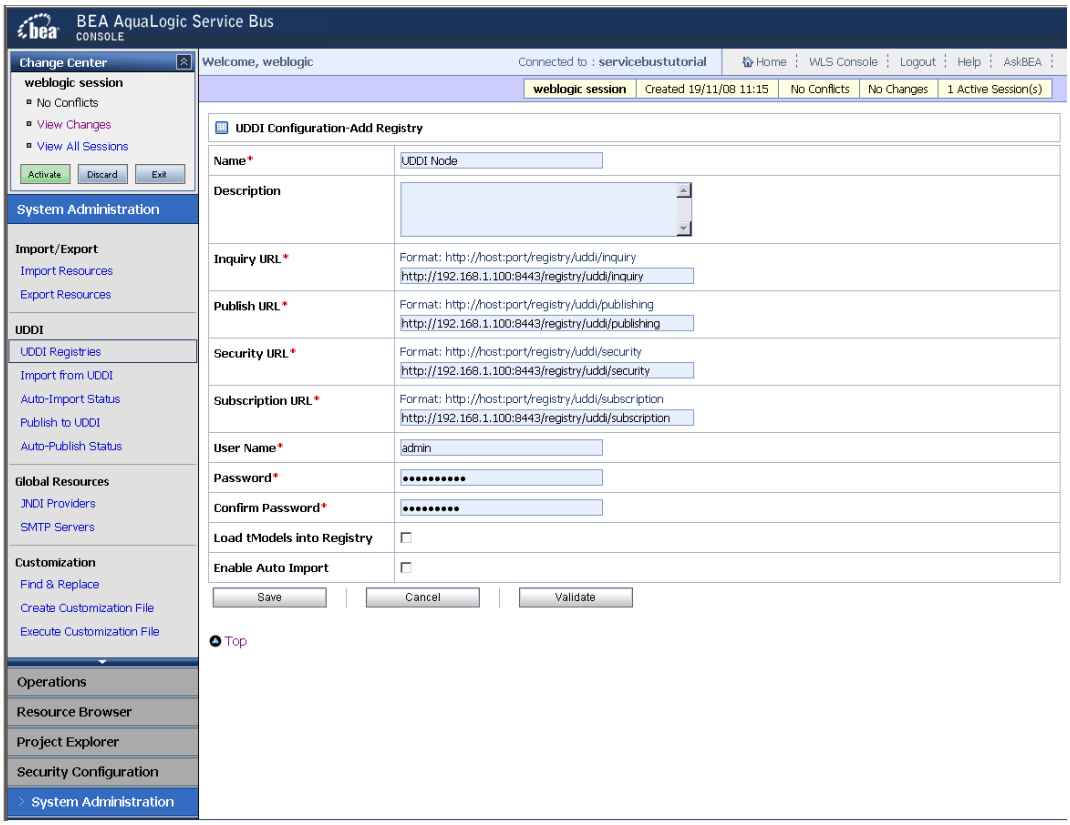

Figura 77 – Formulário para cadastro de repositório UDDI

Para informações mais detalhadas e técnicas sobre o uso de repositórios UDDI com o barramento da BEA, podem ser consultados os endereços disponíveis na base de manuais do fabricante: <http://edocs.bea.com/alsb/docs21/consolehelp/systemadmin.html#1067433> ou <http://edocs.bea.com/alsb/docs21/userguide/uddi.html>.