**Tutorial de configuração do SIPAC 4.22.13**

Primeiramente, deve ter o java 6 instalado no computador, para que seja possível fazer a configuração do ambiente.

Utilizando o Eclipse Luna, é necessário alterar o compilador para qu esteja compatível com o ambiente. Para isto:

Acessar a opção “Window → Preference → Java → Compiler” e na seção “JDK Compliance”, opção “Compiler compliance level:” indicar “1.6”. Clique no botão “OK”. Aparecerá uma tela semelhante a da Figura 1.

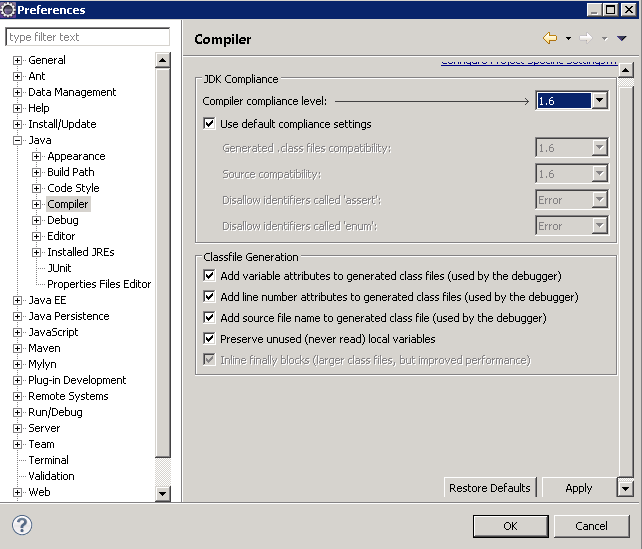


Figura 1 Fonte: wiki UFRN

Em seguida, alterar a configuração do ambiente de execução, através do caminho:

**“Window → Preference → Java → Installed JREs”**

A figura 2 demonstra a tela de configuração.

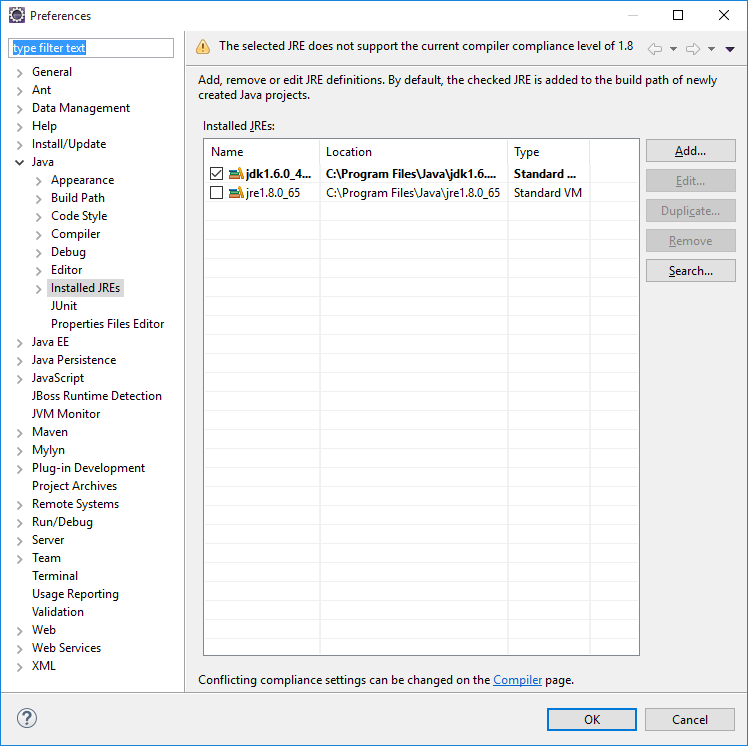


Figura 2 Alterar a versão do java

E indicar o jdk 6 ou o jre6, ficando a critério do usuário. No caso deste tutorial, foi utilizado o jdk1.6.0\_45.

O próximo passo é configurar a codificação de caracteres, que o padrão do ambiente é a ISO-8859-1. Para selecioná-la, deve-se seguir o caminho:

**“Window → Preference → Workspace”**

E em “Text file encoding”, selecionar o ISO-8859-1. A figura 3 exibe a configuração.

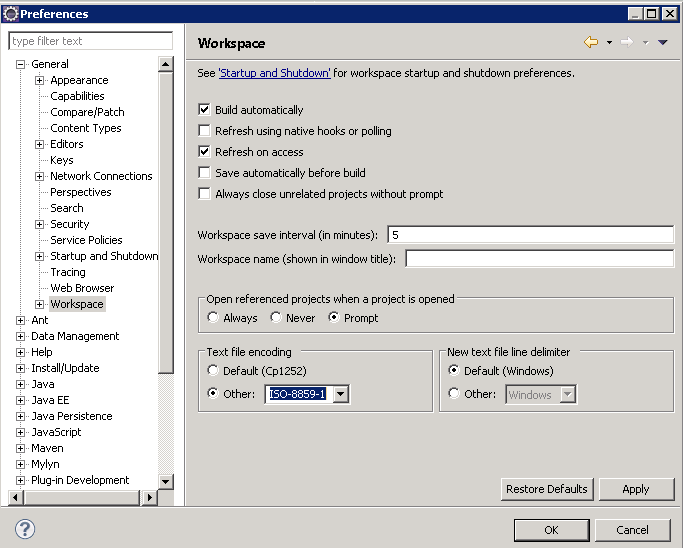


Figura 3 Configuração da codificação

O passo seguinte é a instalação do plugin para SVN, preferencialmente o subeclipse 1.10.13. Para isto, é preciso acessar a opção:

**“Help → Eclipse Market Place”**

E pesquisar pelo subeclipse, em seguida, instalá-lo.

Após instalar o plugin, deve-se clicar em “*open perspective*” e em seguida em “*SVN Repository Exploring*”. Na tela que aparecer, deve-se informar a URL que o eclipse irá conectar para baixar os arquivos do projeto.

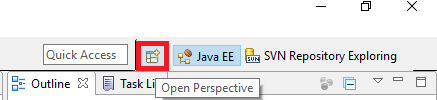


Figura 4: Open Perspective

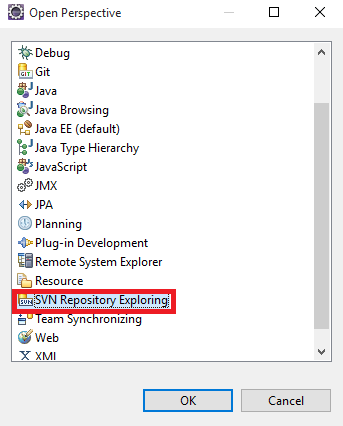


Figura 5 SVN Repository Exploring

Para ter acesso aos arquivos, é necessário configurar o acesso. Para isto, deve-se clicar com o botão direito do “mouse” na aba “SVN Repositories”, selecione “New → Repository Location…”, conforme exibido pela Figura 6.

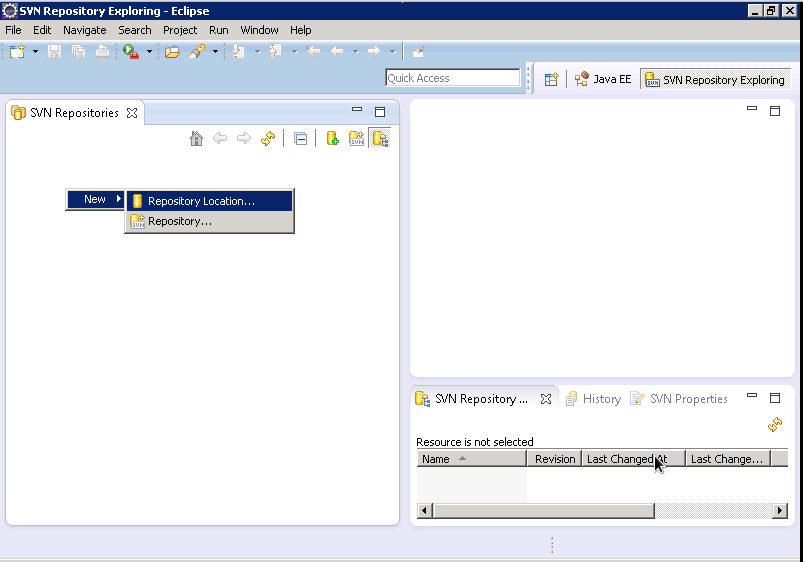


Figura 6 New Repository Location

O passo seguinte é para informar a url, usuário e senha de conexão ao repositório da UFRN. O link é <https://version.info.ufrn.br/cooperacao>, o usuário é “**ufpa**” e a senha é “**@ufpa#ufrn**” (sem as aspas).

A figura 7 demonstra um exemplo do wiki da ufrn.

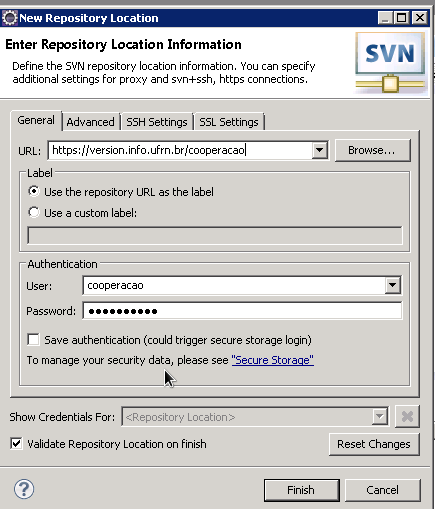


Figura 7 Fonte: Wiki UFRN

Com a configuração realizada, deve-se fazer o checkout das seguintes pastas:

* **ServicosIntegrados 1.4.12;**
* **Arquitetura 2.7.20;**
* **SharedResources 1.6.9;**
* **EntidadesComuns 1.4.15;**
* **Libs 1.5.5;**
* **SIPAC 4.22.13;**

Após o término do download, é necessário verificar se todas as pastas estão com o java 6 no build path. Caso não esteja, será necessário inseri-lo.

O passo seguinte é a geração dos .jar’s que auxiliarão a execução do projeto. Deve-se criar os seguintes arquivos:

* **arq-2.7.20;**
* **dto-1.4.12;**
* **comum-1.4.15**

Para gerar estes três arquivos, é preciso alterar os seguintes arquivos, chamados de build.properties:

* Da pasta Arquitetura;
* Da pasta EntidadesComuns;
* Da pasta ServicosIntegrados;

As figuras 8, 9 e 10 demonstram exemplos de configuração.

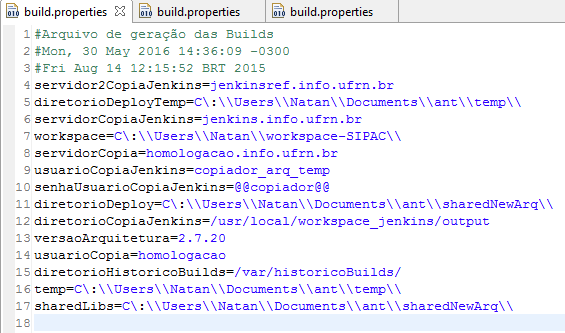


Figura 8 build.properties Arquitetura

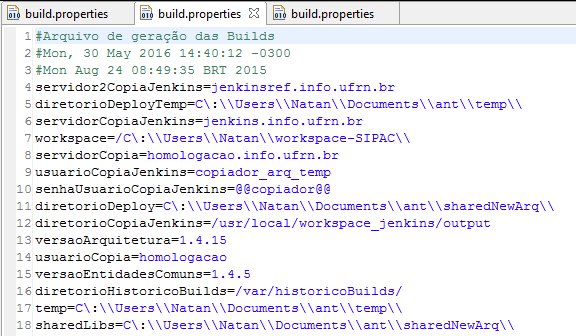


Figura 9 build.properties EntidadesComuns

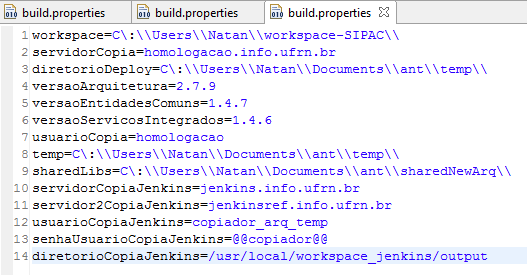


Figura 10 build.properties ServicosIntegrados

As pastas temp e sharedNewArq, foram criadas no computador onde foi executado a geração dos builds. O endereço foi repassado para os build.properties, conforme as três imagens anteriores exibem.

Para gerar os arquivos .jar, é necessário acessar o eclipse e clicar em “window → show view → other” e pesquisar por “ant”.

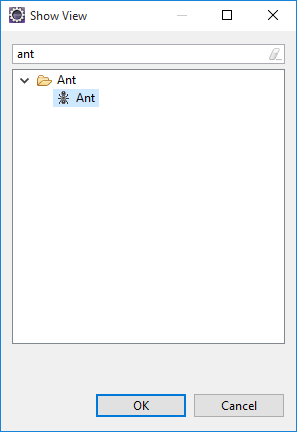


Figura 11 Ant

Aparecerá na parte de baixo do eclipse o ícone do ant, demonstrando que ele foi iniciado. Agora, para gerar os builds, é necessário clicar em “add buildfiles” e adicionar os build.properties que foram editados, das pastas Arquitetura, EntidadesComuns e ServicosIntegrados.

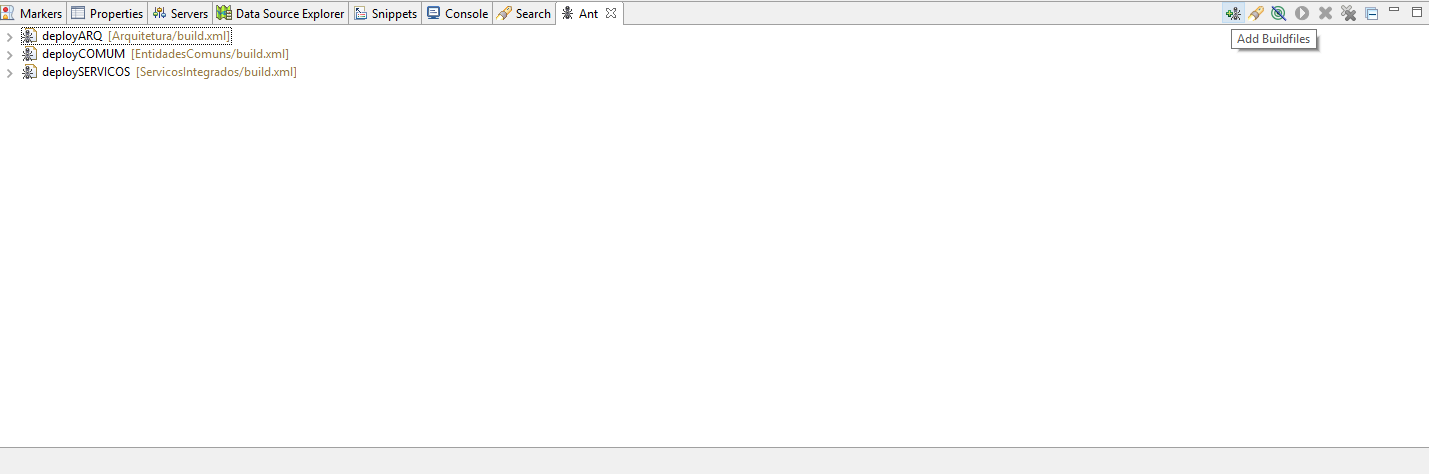


Figura 12 add Buildfiles

Após adicioná-los, é possível criar os .jar’s para a execução do projeto.

Para a geração dos arquivos, primeiramente deve-se clicar em “DeployARQ **→** deployAll [default]” para gerar o jar da arquitetura. Ao clicar, ele vai demonstrar uma janela como a da Figura 13, onde deve ser informado o parâmetro **2.7.20**.

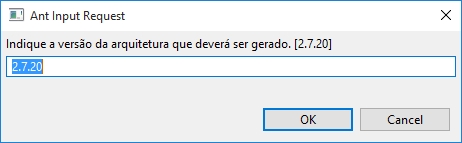


Figura 13 arq-2.7.20

O próximo a ser gerado é o dos serviços. Note que deve ser clicado no ícone “deployDTO” e fornecer o valor **1.4.12**, conforme a Figura 14.

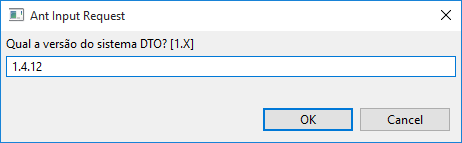


Figura 14 dto-1.4.12

Por fim, deve-se criar o do comum, clicando no ícone “deployCOMUM → deployAll[default]” e informar o valor **1.4.15**, conforme a figura 15.

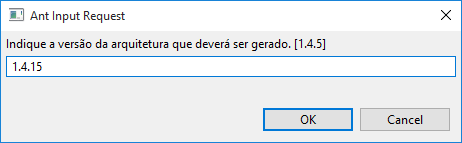


Figura 15 comum-1.4.15

Quando são gerados os .jar’s dto e comum, o eclipse demonstra uma mensagem em vermelho, mas geralmente os arquivos serão criados e estarão na pasta sharedNewArq.

Com os três arquivos, é necessário verificar todas as pastas do projeto, exceto as pastas “LIBS”, “Arquitetura” e “ServicosIntegrados”, procurando se existe o diretório “dependencias” dentro delas. Se não existir, será necessário criá-la.

Após as devidas pastas possuírem o subdiretório “dependencias”, deve-se adicionar os .jar’s da seguinte maneira:

* EntidadesComuns/dependencias: Adicionar os .jar’s **arq-2.7.20** e **dto-1.4.12**;
* SharedResources/dependencias: Adicionar os .jar’s **arq-2.7.20**, **dto-1.4.12** e **comum-1.4.15**;
* SIPAC/dependencias: Adicionar os .jar’s **arq-2.7.20**, **dto-1.4.12** e **comum-1.4.15**;

Estes arquivos que foram gerados não serão adicionados ao build path, pois no teste realizado não foi necessário para o funcionamento do programa, porém os arquivos estavam dentro das referidas pastas. Provavelmente não são necessários, mas somente com mais testes que isto pode ser esclarecido.

Agora é necessário inserir as dependências dos projetos. Para informar as dependências das pastas “**EntidadesComuns**”, “**ServicosIntegrados**”, “**SharedResources**” e “**SIPAC**”. Para isto, é necessário clicar com o botão direito do mouse em uma destas pastas e escolher a opção “Properties”, conforme a Figura 16.

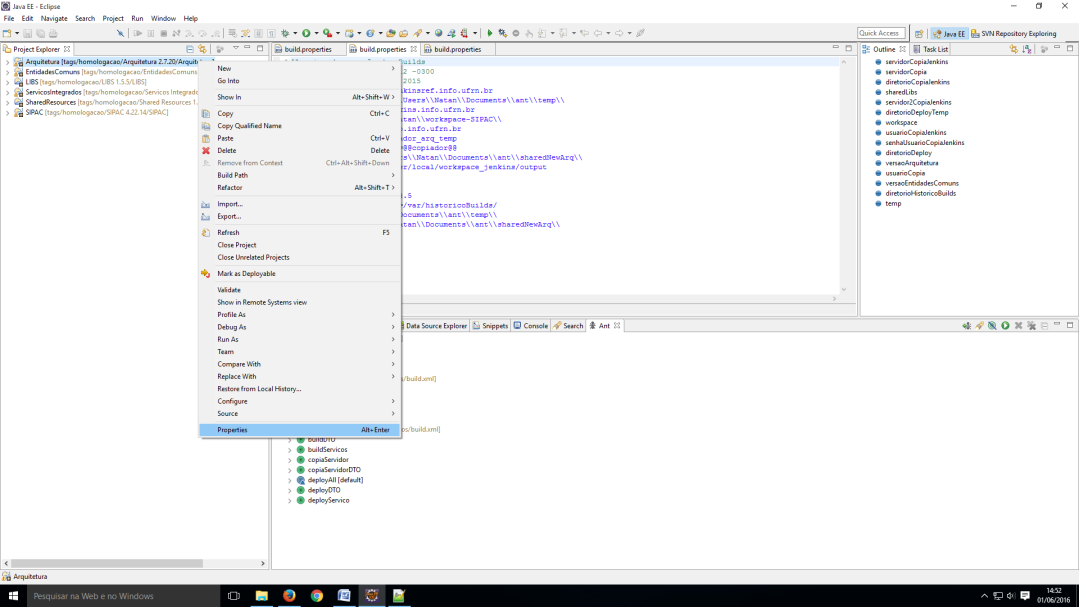


Figura 16 Properties

Depois, escolher a opção “Java Build Path”, depois a aba “Projects” e no botão “Add” para adicionar os projetos que a referida pasta depende. O procedimento é necessário para todas as pastas que foram informadas anteriormente. A figura 17 demonstra um exemplo da forma de inserir os projetos.

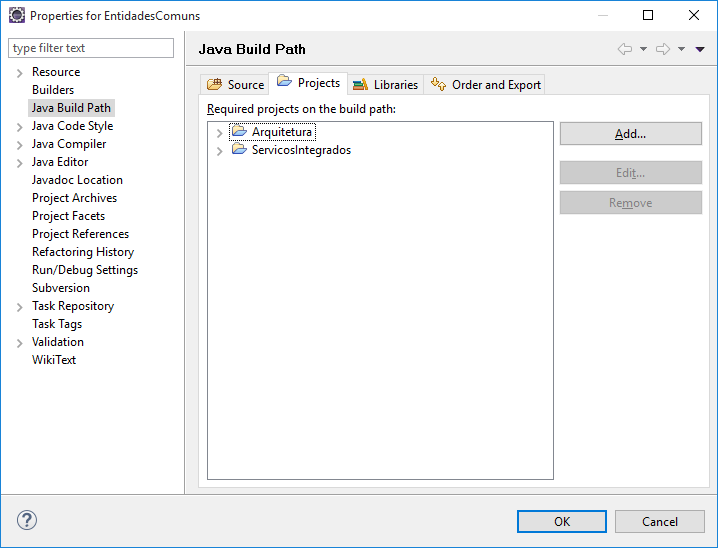


Figura 17 Projects

As dependências são:

* EntidadesComuns **depende** de Arquitetura e ServicosIntegrados;
* ServicosIntegrados **depende** de Arquitetura;
* SharedResources **depende** de Arquitetura, EntidadesComuns e ServicosIntegrados;
* SIPAC **depende** de Arquitetura, EntidadesComuns e ServicosIntegrados;

O próximo passo é configurar o Jboss 5.1, que deve ser baixado do seguinte link da UFRN: <http://migre.me/sLLui>. Ele deve ser descompactado no computador, em uma pasta a critério do usuário.

No eclipse, deve-se clicar na aba “servers” e depois em “No servers are available. Click this link to create a new server…”, conforme a figura 18.

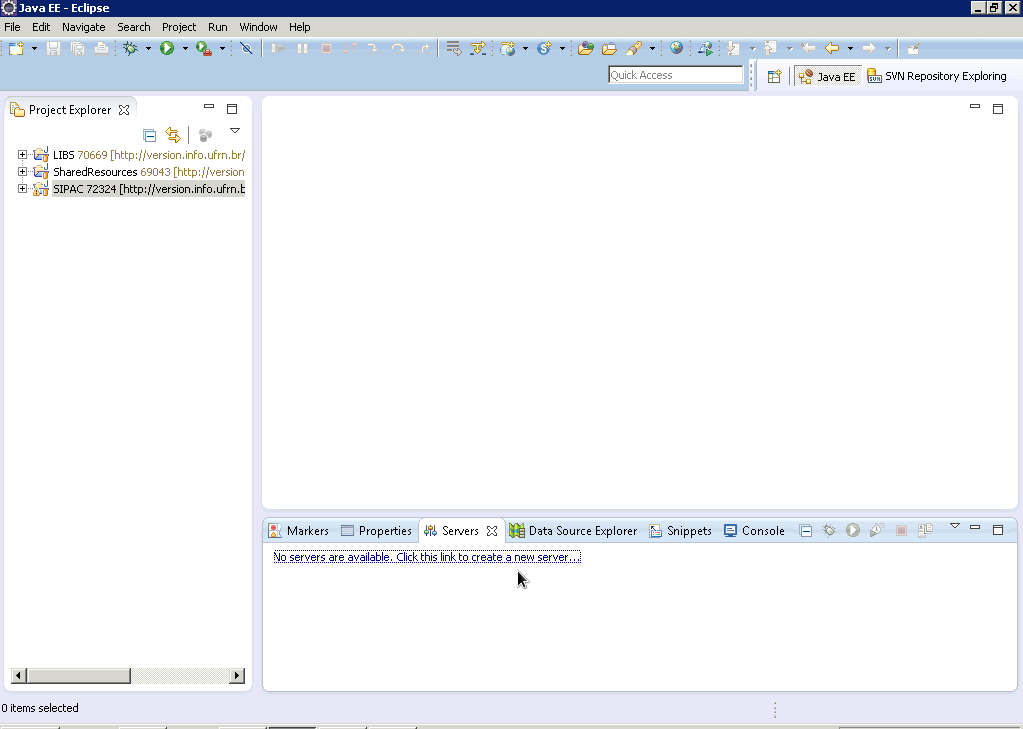


Figura 18 Fonte: Wiki UFRN

Na tela que for exibida, escolher o JBoss 5.1. Caso não apareça nenhum JBoss, é necessário instalá-lo através do “market place” do eclipse, num procedimento similar ao do svn.

A figura 19 exibe a escolha do JBoss 5.1.

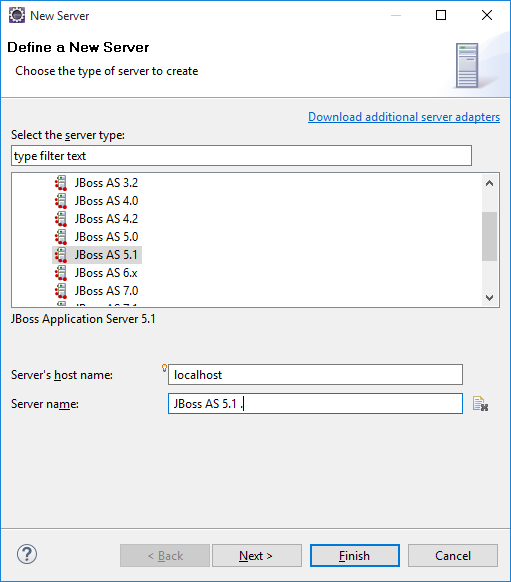


Figura 19 JBoss 5.1

Em seguida, clicar em “Next”, a tela a ser exibida é conforme a Figura 20. Em seguida, clicar em “Next” novamente.

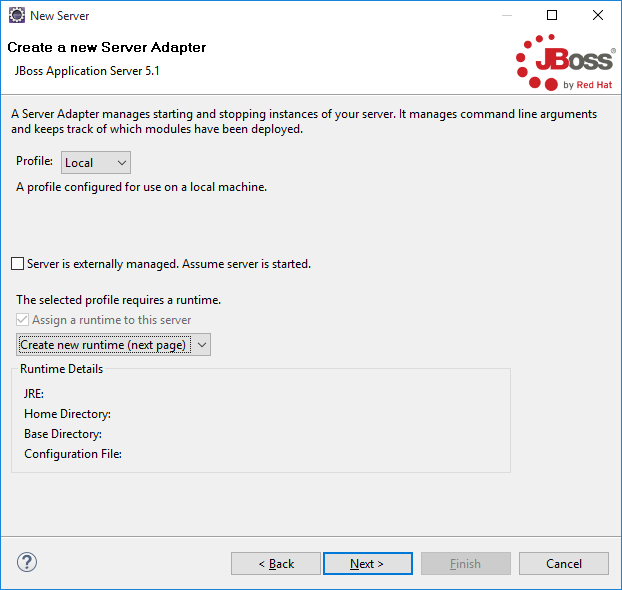


Figura 20 Create a New Server Adapter

Na próxima tela será necessário informar o local onde o JBoss foi descompactado, a versão do java que será utilizado e o nome para o servidor. Conforme o exibido pela Figura 21.

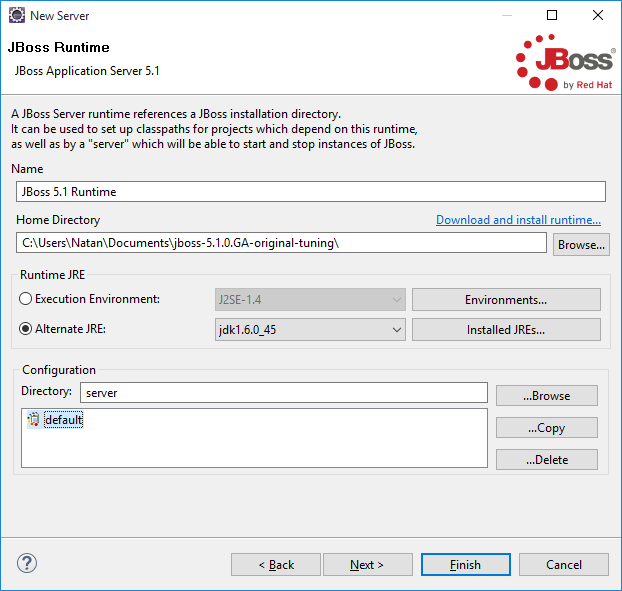


Figura 21 JBoss Runtime

A configuração deve ser similar ao da Figura 21. Terminado isso, clique em “Next” e depois em “Finish”, para terminar a configuração do JBoss no eclipse.

Antes de iniciar os testes, é necessário editar os arquivos do JBoss: profile.xml, postgres-ds.xml, run.conf e remote-services.xml, para adaptá-los aos testes.

O arquivo profile.xml fica na pasta que foi descompactada, no subdiretório jboss-5.1.0.GA-original-tuning\server\default\conf\bootstrap. Neste arquivo são inseridos os diretórios que o JBoss irá tentar executar a aplicação a configuração deve ser similar a:

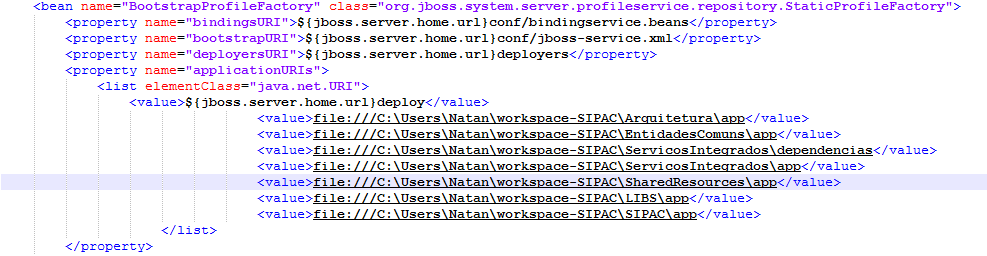


Figura 22 profile.xml

Após salvar o arquivo, deve-se alterar o postgresql-ds.xml, para que seja possível a conexão com o banco de dados. O arquivo fica no diretório jboss-5.1.0.GA-original-tuning\server\default\deploy.



Figura 23 datasource

Todos os campos entre o <xa-datasource> e </xa-daatsource> devem ser preenchidos com os parâmetros do banco, no caso os property name, conforme o exemplo da Figura 23.

Após salvar o arquivo, alterar o arquivo run.conf, que está no diretório jboss-5.1.0.GA-original-tuning\bin. Neste arquivo, basta descomentar a linha onde está o JAVA\_HOME e inserir o caminho onde o java 6 está instalado no computador. A figura 24 exibe um exemplo da edição.

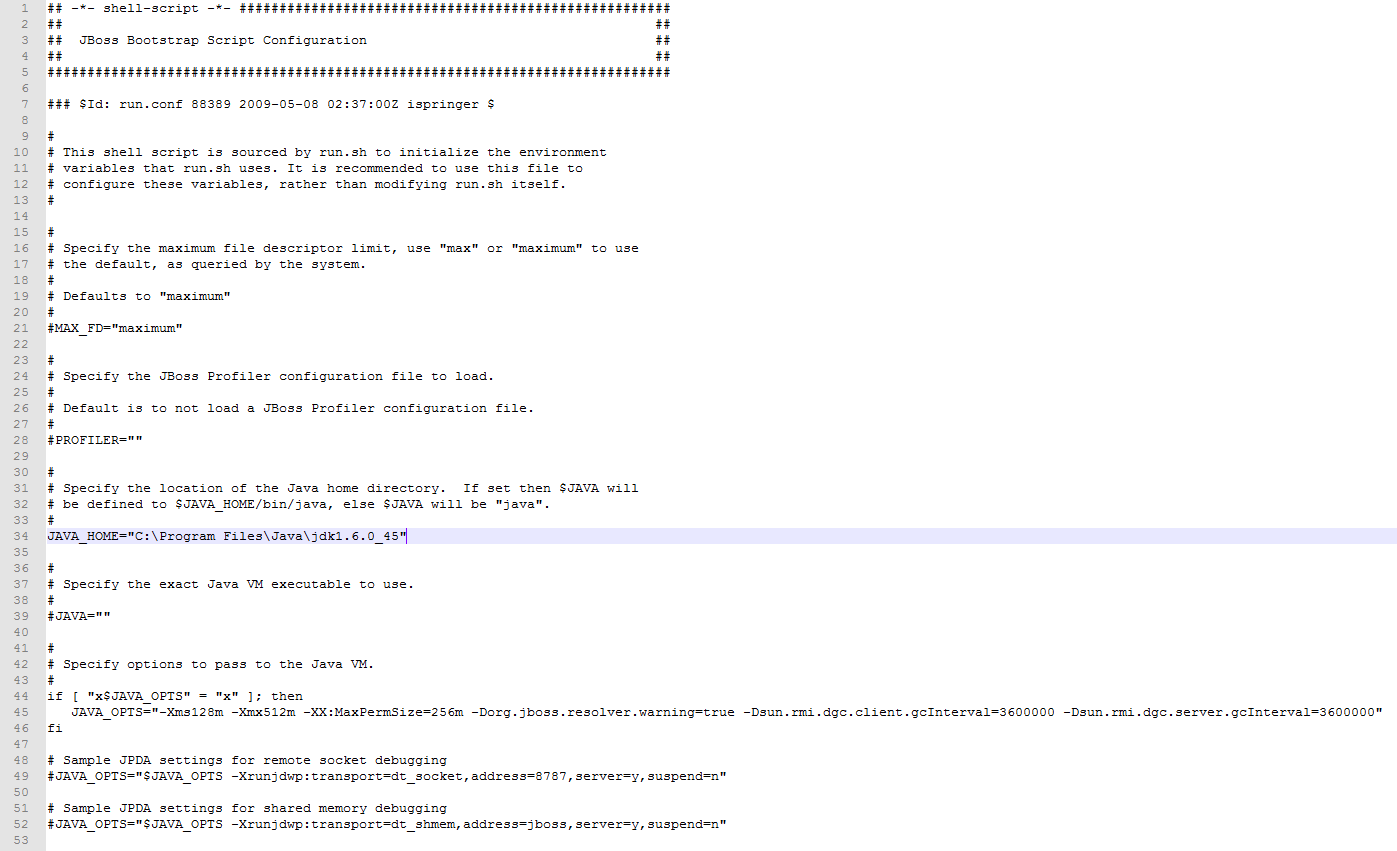


Figura 24 run.conf

Após fechar e salvar o arquivo, deve-se editar o remote-services.xml, deixando-o similar ao que está na versão anterior do SIPAC, pois nos testes que foram realizados, aconteciam alguns erros e o deploy não ocorria. Quando este arquivo foi alterado para ficar similar ao da versão anterior, o deploy ocorreu. O arquivo encontra-se no caminho C:\Users\Natan\workspace-SIPAC\ServicosIntegrados\app\servicos.ear\servicos.war\WEB-INF, e deve ser alterada a linha 34 do arquivo, com o seguinte conteúdo:

<jaxws:endpoint id="buscaServicosRemotosExporter" implementor="br.ufrn.integracao.servicos.BuscaServicosRemotosRemoteServiceImpl" address="/BuscaServicosRemotosService">

Depois deve-se salvar o arquivo e voltar ao eclipse para fazer o deploy da aplicação. Com o eclipse iniciado, clicar na aba “servers” e depois em “start”. Alguns erros podem acontecer durante a execução do deploy, mas provavelmente irá funcionar de maneira normal.