



Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de software

Manuais de Uso de Ferramentas Utilizadas no TCC1

Brasília, DF
2016



Lista de ilustrações

Figura 1 – Log de inicialização do servidor ESB.	10
Figura 2 – Menu para troca de senha padrão.	10
Figura 3 – Estrutura do conector.	12
Figura 4 – Template da operação <i>getRanking</i>	13
Figura 5 – Template da operação <i>init</i>	13
Figura 6 – Descrição do componente "enturma-requests".	14
Figura 7 – Descrição do conector "enturma".	14
Figura 8 – Adição do conector "enturma"ao ESB.	15
Figura 9 – Como criar um serviço de proxy no WSO2 ESB.	16
Figura 10 – Mudar modo de visualização do arquivo de configuração do serviço.	16
Figura 11 – EnTurma - Visualizar Ranking	17
Figura 12 – EnTurma - Resposta da Requisição para Visualizar Ranking	19
Figura 13 – WSO2 - Logs de requisições no ESB.	19

Lista de tabelas

Lista de abreviaturas e siglas

SOA	<i>Service-Oriented Architecture</i>
SOAP	<i>Simple Object Access Protocol</i>
REST	<i>Representational State Transfer</i>
ESB	<i>Enterprise Service Bus</i>
WSDL	<i>Web Services Description Language</i>
JSON	<i>JavaScript Object Notation</i>
WEB	Palavra inglesa que significa "rede"
URL	<i>Uniform Resource Locator</i>
HTTP	<i>Hyper Text Transfer Protocol</i>
HTTPS	<i>Hyper Text Transfer Protocol Secure</i>

Sumário

1	INSTALAÇÃO DO WSO2 ESB	9
1.1	Introdução	9
1.2	Pré-requisitos	9
1.3	Instalação	9
1.3.1	Trocando senha de acesso padrão	10
2	CRIAÇÃO E USO DE CONECTORES	11
2.1	Criação de conectores	11
2.1.1	Estrutura do conector	12
2.1.1.1	Template para acesso à operações do serviço	13
2.1.1.2	Descrição do componente do conector	14
2.1.1.3	Descrição do conector	14
2.2	Adição do conector ao ESB	15
2.3	Criação do serviço no ESB	15
2.4	Uso do serviço ESB em uma aplicação	17
	REFERÊNCIAS	21
	APÊNDICES	23
	APÊNDICE A – CONFIGURAÇÃO DO SERVIÇO DE PROXY CUSTOMIZADO NO ESB	25

1 Instalação do WSO2 ESB

"Este capítulo tem como objetivo apresentar o referencial teórico que embasa a pesquisa deste trabalho, incluindo conceitos abordados"

1.1 Introdução

Este manual de instalação é referente à ferramenta WSO2 ESB - WSO2 Enterprise Service Bus. Esta é uma das várias ferramentas que implementam o ESB, um mecanismo utilizado dentro da arquitetura SOA que possibilita a comunicação entre diversas aplicações utilizando diferentes formatos e protocolos de mensagens. Utilizando o ESB, uma aplicação se comunica apenas com esta ferramenta e esta, por sua vez, é responsável por transformar e entregar as mensagens do serviço destinatário correto (([TUTORIAL...](#))).

1.2 Pré-requisitos

Para a instalação do WSO2 ESB, são necessários:

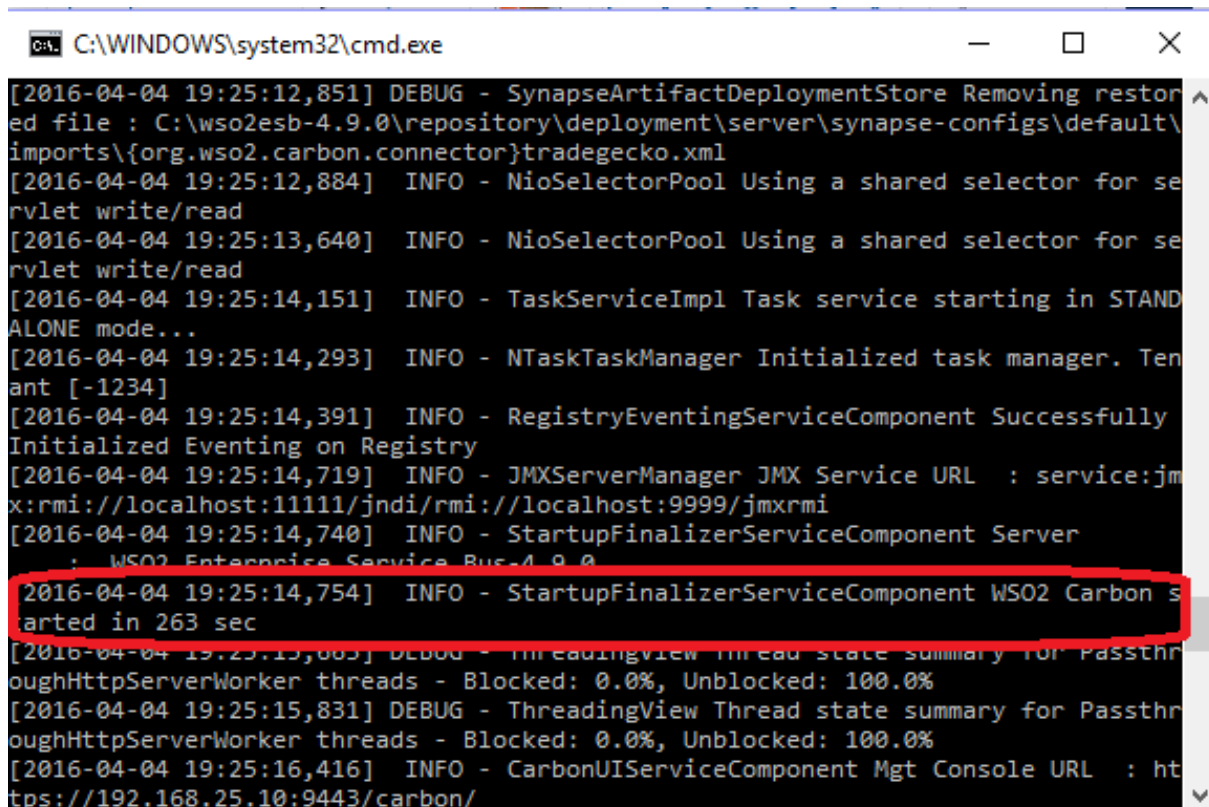
- JDK 1.6.* ou versão mais recente.
- Variável de ambiente JAVA_HOME configurada para <JDK_HOME>.

1.3 Instalação

Os passos para a instalação do WSO2 ESB (versão 4.9.0) estão descritos abaixo:

1. Fazer o download do arquivo **wso2esb-4.9.0.zip**.
2. Extrair o arquivo **wso2esb-4.9.0.zip**.
3. No local onde o arquivo **wso2esb-4.9.0.zip** foi extraído (referenciado como <WSO2>), acessar a pasta <WSO2>\bin.
4. Iniciar o servidor ESB executando:
 - Em ambiente Linux: **wso2server.sh**
 - Em ambiente Windows: **wso2server.bat**

O servidor estará completamente ativado quando a mensagem destacada abaixo aparecer no *log* da aplicação.



```
[2016-04-04 19:25:12,851] DEBUG - SynapseArtifactDeploymentStore Removing restored file : C:\wso2esb-4.9.0\repository\deployment\server\synapse-configs\default\imports\{org.wso2.carbon.connector}tradegecko.xml
[2016-04-04 19:25:12,884] INFO - NioSelectorPool Using a shared selector for servlet write/read
[2016-04-04 19:25:13,640] INFO - NioSelectorPool Using a shared selector for servlet write/read
[2016-04-04 19:25:14,151] INFO - TaskServiceImpl Task service starting in STANDALONE mode...
[2016-04-04 19:25:14,293] INFO - NTaskTaskManager Initialized task manager. Tenant [-1234]
[2016-04-04 19:25:14,391] INFO - RegistryEventingServiceComponent Successfully Initialized Eventing on Registry
[2016-04-04 19:25:14,719] INFO - JMXServerManager JMX Service URL : service:jmx:rmi://localhost:11111/jndi/rmi://localhost:9999/jmxrmi
[2016-04-04 19:25:14,740] INFO - StartupFinalizerServiceComponent Server : WSO2 Enterprise Service Bus-4.9.0
[2016-04-04 19:25:14,754] INFO - StartupFinalizerServiceComponent WSO2 Carbon started in 263 sec
[2016-04-04 19:25:15,003] DEBUG - ThreadingView Thread state summary for PassthroughHttpServerWorker threads - Blocked: 0.0%, Unblocked: 100.0%
[2016-04-04 19:25:15,831] DEBUG - ThreadingView Thread state summary for PassthroughHttpServerWorker threads - Blocked: 0.0%, Unblocked: 100.0%
[2016-04-04 19:25:16,416] INFO - CarbonUIServiceComponent Mgt Console URL : https://192.168.25.10:9443/carbon/
```

Figura 1: Log de inicialização do servidor ESB.

1.3.1 Trocando senha de acesso padrão

Para a troca da senha padrão, basta acessar a aplicação ESB usando login e senha padrão e, em seguida, acessar o menu conforme exibido na figura abaixo:

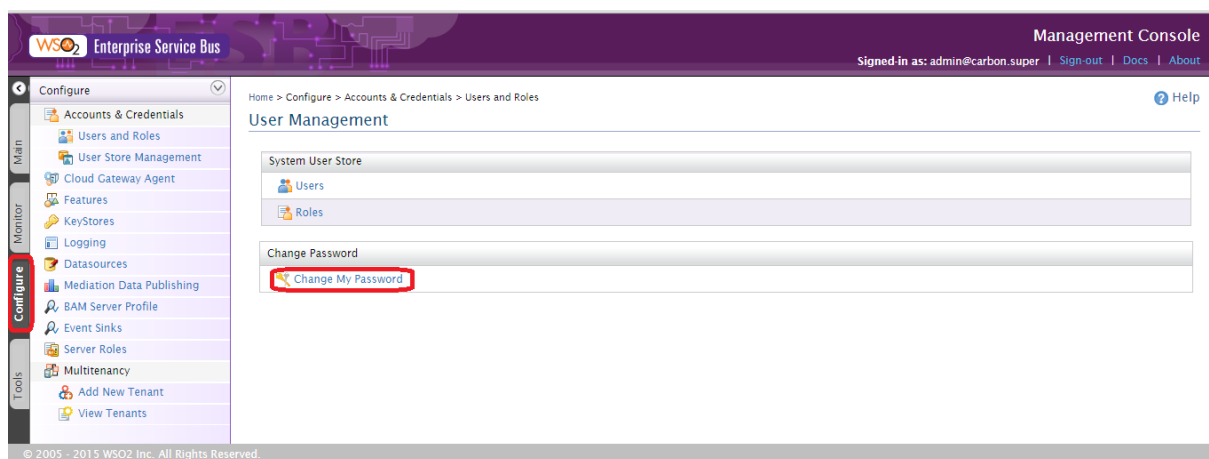


Figura 2: Menu para troca de senha padrão.

2 Criação e Uso de Conectores

Neste capítulo veremos como criar um conector, gerenciá-lo e utilizá-lo através da criação de um serviço de proxy customizado.

2.1 Criação de conectores

A criação de conectores é feita para encapsular a API de um serviço externo, disponibilizando operações que podem ser acessadas por aplicações de forma simplificada. Os conectores são definidos por templates, que contém todas as operações disponíveis bem como propriedades (ou valores) necessárias para a execução da operação do serviço solicitada (([WORKING...](#))).

A ferramenta WSO2 ESB possui diversos conectores que permitem o uso de serviços tais como Twitter, Facebook, eBay, Foursquare, Github e Gmail. Também é possível criar conectores customizados e utilizá-los no WSO2 ESB.

Para a criação do conector deste manual, foi utilizada a aplicação EnTurma, desenvolvida por alunos da Universidade de Brasília para o projeto final das disciplinas Gerenciamento de Projetos e Portifólios e Métodos de Desenvolvimento de Software durante o primeiro semestre do ano letivo de 2015. A aplicação pode ser acessada através do link [<http://www.projetoenturma.com.br/>](http://www.projetoenturma.com.br/). O objetivo é exibir de forma mais intuitiva dados referentes à educação básica brasileira. A aplicação possui três funcionalidades:

- Visualização de Ranking: de acordo com o ano e turma escolhidos, é exibida uma lista ordenada de acordo com índices de evasão, rendimento escolar, distorção de idade com relação à recomendada para a turma escolhida e dados da prova IDEB (se foi aplicada no ano escolhido).
- Relatórios de turmas: são exibidos os mesmos dados para a visualização do ranking, mas apenas para uma turma escolhida de determinada localidade. É possível determinar a partir de qual ano os dados devem ser exibidos, mostrando a evolução dos dados.
- Comparar turmas: dadas duas turmas, é feita uma comparação. Assim é possível verificar se existem turmas com melhores desempenhos e proporcionar a pesquisa de melhores metodologias de ensino e promover a melhoria da educação brasileira.

Neste manual será criado um conector customizado para esta aplicação. Também serão exibidos como gerenciá-lo no ESB e o uso deste conector em uma segunda aplicação.

2.1.1 Estrutura do conector

Para a criação do conector é necessário o acesso à API do serviço para o qual este será criado. A estrutura do conector varia de acordo com o tipo de API utilizada: REST APIs permitem a criação de conectores apenas com arquivos de configuração em formato XML, enquanto JAVA APIs necessitam que os arquivos de configuração sejam escritos em formato XML e utilizam (ou indicam) as operações escritas em Java.

A aplicação para o qual será criado o conector disponibiliza o acesso de seus serviços via REST API. Assim sendo, a estrutura do conector será:

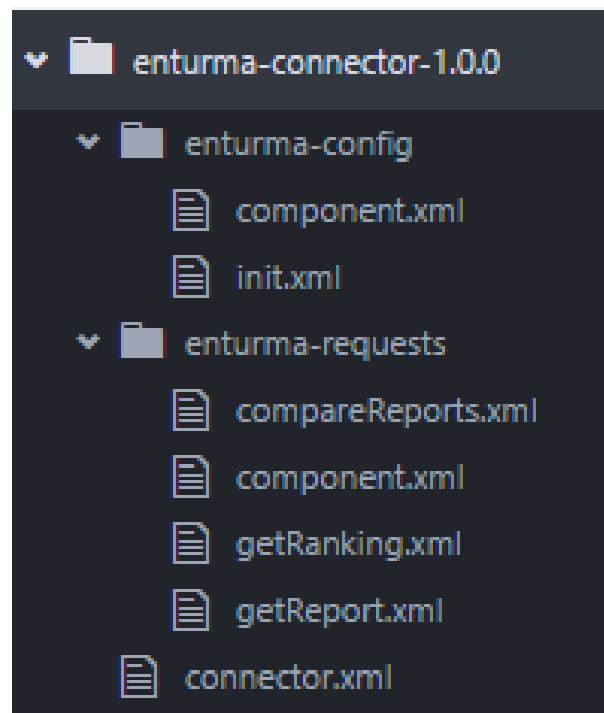


Figura 3: Estrutura do conector.

- As pastas devem conter os conjuntos de templates de operações oferecidas pelo serviço. A criação de várias pastas é recomendada para o agrupamento de operações do mesmo tipo.
- Cada pasta é considerada um componente do conector. O arquivo *component.xml* contém a declaração e descrição das operações disponíveis.
- Cada operação deve ser descrita em um arquivo *.xml*. As operações serão identificadas no arquivo *component.xml*, onde serão denominadas e associadas ao arquivo que as descrevem.
- É recomendado que as configurações (parâmetros) essenciais para o uso do serviço estejam descritas em uma pasta *<nome_conector>-config*.

- O arquivo *connector.xml* contém a descrição de todos os componentes do conector.

2.1.1.1 Template para acesso à operações do serviço

As operações disponibilizadas para uso do serviço são descritas através de templates. Quando a operação é chamada por uma aplicação, os parâmetros solicitados são substituídos pelos valores fornecidos e a operação é executada de acordo com a descrição do template.

Veja na figura abaixo o template da operação que permite a recuperação de dados sobre o ranking de dado ano e grade (parâmetros usados pelo template).

```
getRanking.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <template name="getRanking" xmlns="http://ws.apache.org/ns/synapse">
3    <parameter name="year" description="The initial year of the report"/>
4    <parameter name="grade" description="The grade the user want to see the report (1º ano to 9º ano)
5      - value from 1 to 9"/>
6    <sequence>
7      <property name="uri.var.year" expression="$func:year"/>
8      <property name="uri.var.grade" expression="$func:grade"/>
9      <call>
10       <endpoint>
11         <http method="get"
12           uri-template="{uri.var.baseUrl}/ranking/request_ranking.json?utf8=E2%9C%93&year={+uri.var.year}&
13             grade={+uri.var.grade}%C2%B0%20ano"/>
14         </endpoint>
15       </call>
16     </sequence>
17 </template>
```

Figura 4: Template da operação *getRanking*.

A operação *getRanking* é uma chamada do método GET da API REST no link especificado pelo *uri-template*. O retorno desta operação é o resultado desta chamada.

A URL base da aplicação é inicializada na operação *init* do serviço.

```
init.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <template name="init" xmlns="http://ws.apache.org/ns/synapse">
3    <parameter name="baseUrl" description="Url EnTurma Initial Page"/>
4    <sequence>
5      <property name="uri.var.baseUrl" expression="$func:baseUrl"/>
6    </sequence>
7  </template>
8
```

Figura 5: Template da operação *init*.

2.1.1.2 Descrição do componente do conector

Uma vez que as operações estão definidas, o componente pode ser descrito. Para cada pasta - conjunto de operações - deve ser criado um arquivo onde o componente será descrito. O conector criado para este manual possui dois componentes: "enturma-config" e "enturma-requests". Ambos possuem um arquivo chamado *component.xml*, onde as operações de tal componente são descritas. Na figura abaixo pode-se ver a descrição do componente "enturma-requests".

```
component.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <component name="enturma-requests" type="synapse/template">
3    <subComponents>
4      <component name="getReport">
5        <file>getReport.xml</file>
6        <description>Get report details from Prova Brasil tests results.</description>
7      </component>
8      <component name="getRanking">
9        <file>getRanking.xml</file>
10       <description>Get ranking details from Prova Brasil tests results.</description>
11     </component>
12     <component name="compareReports">
13       <file>compareReports.xml</file>
14       <description>Compare details from Prova Brasil tests results from two different classes of the same grade.</description>
15     </component>
16   </subComponents>
17 </component>
```

Figura 6: Descrição do componente "enturma-requests".

2.1.1.3 Descrição do conector

A definição do nome no ESB, o pacote e os componentes do conector são descritos no arquivo *connector.xml*. A descrição dos componentes são recomendadas.

```
connector.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <connector>
3    <component name="enturma" package="org.wso2.carbon.connector" >
4      <dependency component="enturma-config"/>
5      <dependency component="enturma-requests"/>
6      <description>enturma connector libraries</description>
7    </component>
8  </connector>
9
```

Figura 7: Descrição do conector "enturma".

Para finalizar, o conector deve ser compactado em uma pasta .zip (apenas neste formato). O arquivo criado neste manual foi denominado *enturma-connector-1.0.0.zip*.

2.2 Adição do conector ao ESB

No ESB será adicionado o arquivo *enturma-connector-1.0.0.zip*. Para isto, acesse o menu indicado na figura abaixo, escolha o arquivo .zip criado e faça o *upload* deste.

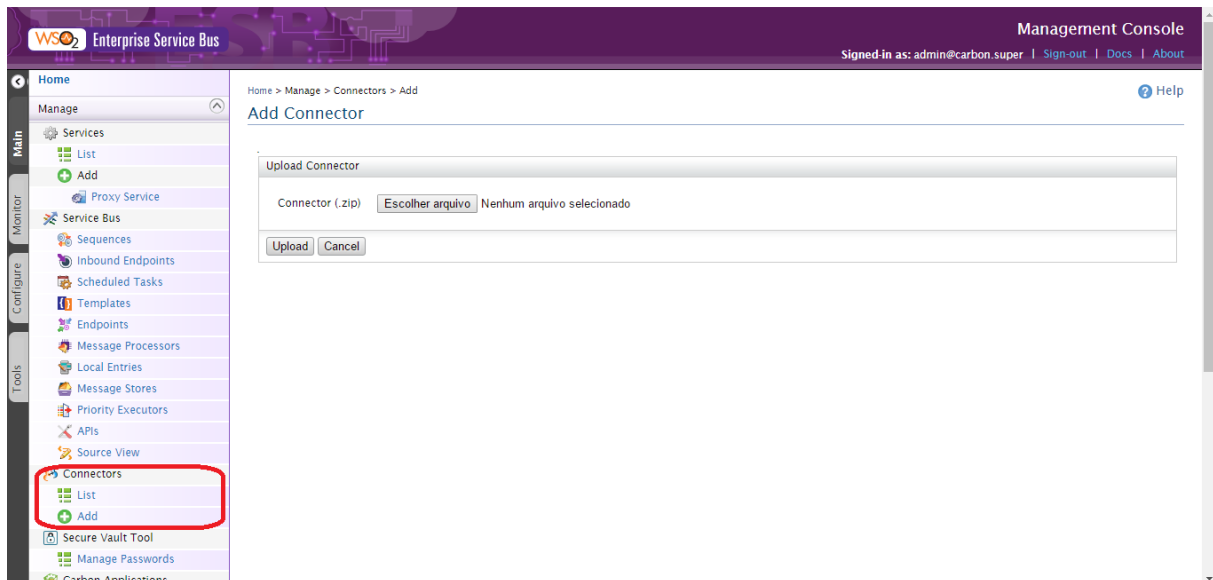


Figura 8: Adição do conector "enturma" ao ESB.

Ao realizar o *upload* do arquivo, as operações do serviço descritos no conector poderão ser acessadas se incluídas em um serviço disponibilizado no ESB. A criação do serviço customizado para acessar as operações do conector "enturma" será descrita logo adiante neste manual.

A lista de conectores deverá ser atualizada. Para que possa ser utilizado, o conector deverá ser ativado: apenas clique no *status* correspondente ao conector. As operações disponíveis para uso através do conector podem ser vistos ao clicar no conector identificado pelo nome descrito no arquivo *connector.xml*.

2.3 Criação do serviço no ESB

Para o uso de operações no ESB é necessária a criação do serviço e a declaração de suas operações. Algumas configurações do serviço podem ser feitas durante a criação deste no WSO2 ESB. Entre as configurações permitidas estão a declaração de propriedades necessárias para o acesso às operações, bem como a sequência de ações a serem tomadas pelo ESB, tanto ao receber uma requisição quanto ao enviar a resposta da mesma, a especificação do formato de entrada e saída das mensagens e do meio de transporte das mensagens.

O WSO2 ESB permite que sejam criados diferentes tipos de serviço *proxy* denominados transformador, seguro, baseado em WSDL, logging, passagem direta e customizado.

Para utilizar o conector criado para a aplicação EnTurma, foi criado um conector customizado: no menu "Main" no canto esquerdo, clicar em "Proxy Service" e em seguida em "Custom Proxy", conforme exibido na figura abaixo.

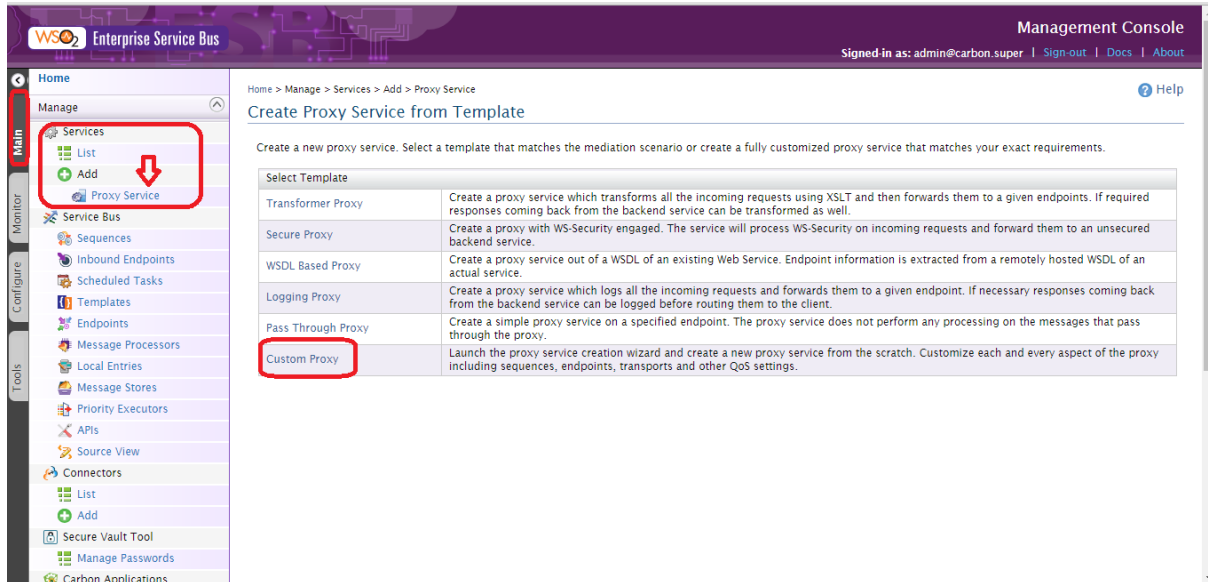


Figura 9: Como criar um serviço de proxy no WSO2 ESB.

Ao seguir os passos anteriormente descritos, a tela a seguir será exibida. Neste manual será utilizada a opção de escrever no arquivo de configuração, que é possível ao clicar no link destacado na figura abaixo.

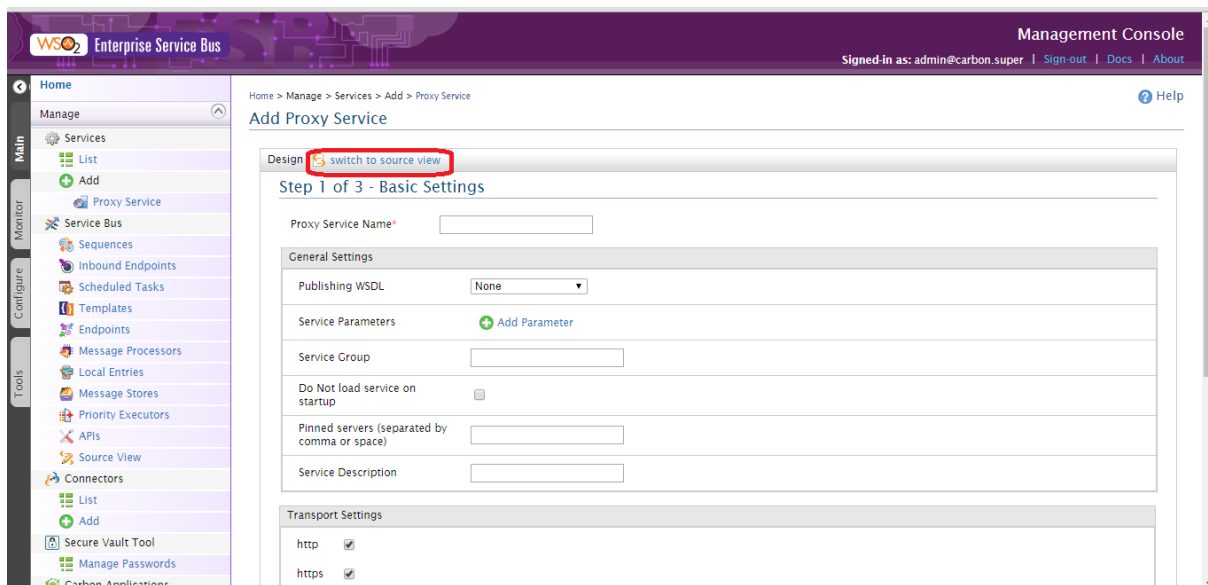


Figura 10: Mudar modo de visualização do arquivo de configuração do serviço.

A configuração do serviço que foi escrito está no apêndice (LINK PARA ANEXO) deste documento. Este serviço foi escrito e, ao ser salvo, pode ser visto na lista de serviços

disponíveis no ESB (na aba "*Main*" no canto esquerdo, acessar o menu *Manage* -> *Services* -> *List*).

2.4 Uso do serviço ESB em uma aplicação

Para exemplificar o acesso a operações de um serviço, foi criada uma aplicação cliente que faz uso do conector criado nas seções anteriores deste manual: a aplicação cliente fará uma requisição utilizando a URL do serviço, especificando a operação e fornecendo os valores necessários para esta ser executada. O resultado exibido será o texto puro obtido, que poderia ser manipulado e utilizado para a visualização de resultados de maneira iterativa.

A aplicação cliente foi escrita em Python (3.4.3) com o auxílio do *framework* Django (1.9.4). A imagem abaixo apresenta a tela onde a operação de visualização de *rankings* de acordo com dados sobre a educação brasileira. O *ranking* é exibido de acordo com o ano e a turma (nível de escolaridade) definidos.

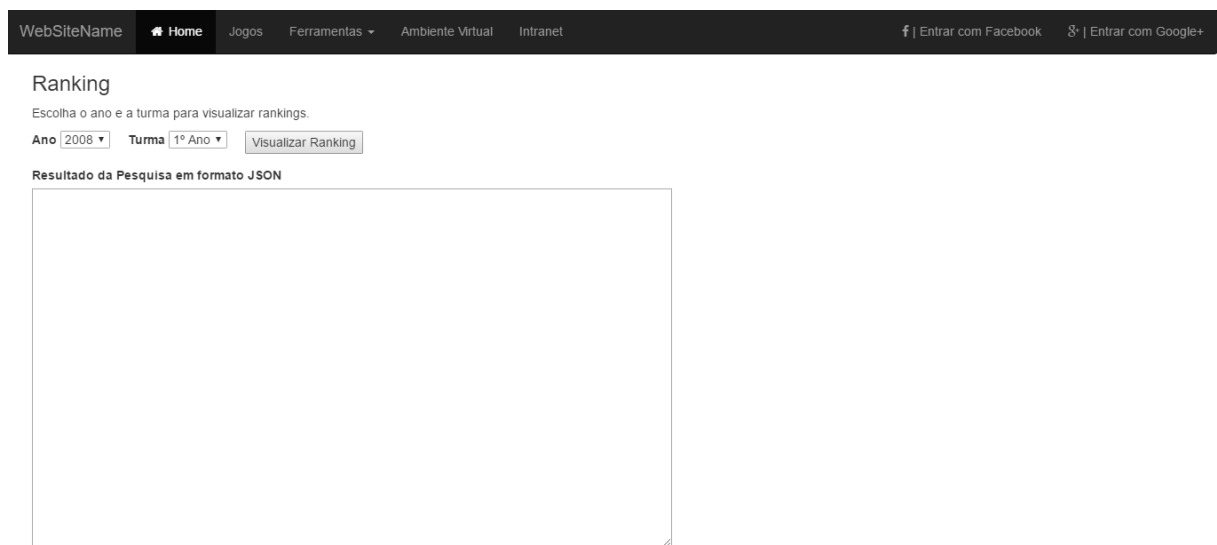


Figura 11: EnTurma - Visualizar Ranking

O método da aplicação cliente que faz a requisição dos dados referentes ao ranking dados ano e turma está escrito abaixo:

```
def requisita_ranking(request):  
    resultado = ''  
    if request.method == 'POST':  
        ano = request.POST["ano"]  
        turma = request.POST["turma"]  
  
        url = 'http://localhost:8280/services/enturmaConnector'
```

```
cabecalho = { 'Action': 'urn:getRanking',  
              'Content-type': 'application/json',  
              'Cache-Control': 'no-cache' }  
dados = { 'baseUrl': 'http://www.projetoenturma.com.br',  
          'year': ano, 'grade': turma }  
resposta = requests.post(url, json=dados, headers=cabecalho)  
resultado = resposta.text  
  
contexto = RequestContext(request, { 'resultado': resultado })  
  
return render_to_response('app/ranking.html',  
                          context_instance=contexto)
```

- **url**: indica a url do serviço.
- **cabecalho**: em formato JSON, contém a definição da operação (*Action*), o tipo de conteúdo (geralmente *application/* + tipo de mensagem), neste caso é o JSON e outras flags.
- **dados**: em formato JSON, contém os valores dos parâmetros necessários para a execução da operação.

Em Python/Django, existe a biblioteca Requests, que permite a integração de aplicações desenvolvidas nesta plataforma com serviços disponíveis na WEB através de requisições HTTP e HTTPS. Esta biblioteca foi utilizada visto sua simplicidade e o fato de o serviço criado para acesso às operações da aplicação EnTurma usar HTTP e HTTPS para transporte de mensagens (recebimento de requisição e envio de respostas).

Na mini-aplicação cliente, a resposta é exibida dentro da caixa de texto em seu formato original. Para que o serviço possa ser encontrado, é necessário que o servidor WSO2 ESB esteja ativo.

Figura 12: EnTurma - Resposta da Requisição para Visualizar Ranking

O ESB tem funcionalidades que permitem visualizar os *logs* de transações que ocorrem no ESB. Assim, quando a requisição para um determinado serviço é feita, os *logs* relacionados aos recebimento da requisição e envio da resposta podem ser vistos, conforme mostrado na figura abaixo.

Figura 13: WSO2 - Logs de requisições no ESB.

Referências

TUTORIAL - Enterprise Service Bus 4.6.0 - WSO2 Documentation. Disponível em: <https://docs.wso2.com/display/ESB460/Tutorial>. Citado na página 9.

WORKING with Connectors - Enterprise Service Bus 4.8.0 - WSO2 Documentation. Disponível em: <https://docs.wso2.com/display/ESB480/Working+with+Connectors>. Citado na página 11.

Apêndices

APÊNDICE A – Configuração do Serviço de Proxy Customizado no ESB

```

<?xml version="1.0" encoding="UTF-8"?>
<proxy xmlns="http://ws.apache.org/ns/synapse"
    name="enturmaConnector"
    transports="http,https"
    statistics="enable"
    trace="enable"
    startOnLoad="true">
    <target>
        <inSequence>
            <property name="grade" expression="json-eval($.grade)"/>
            <property name="year" expression="json-eval($.year)"/>
            <property name="state" expression="json-eval($.state)"/>
            <property name="test_type" expression=
                "json-eval($.test_type)"/>
            <property name="public_type" expression=
                "json-eval($.public_type)"/>
            <property name="local" expression="json-eval($.local)"/>
            <property name="first_year" expression=
                "json-eval($.first_year)"/>
            <property name="first_state" expression=
                "json-eval($.first_state)"/>
            <property name="first_test_type" expression=
                "json-eval($.first_test_type)"/>
            <property name="first_public_type" expression=
                "json-eval($.first_public_type)"/>
            <property name="first_local" expression=
                "json-eval($.first_local)"/>
            <property name="second_year" expression=
                "json-eval($.second_year)"/>
            <property name="second_state" expression=
                "json-eval($.second_state)"/>
            <property name="second_test_type" expression=
                "json-eval($second_.test_type)"/>
        
```

```

<property name="second_public_type" expression=
  "json-eval($.second_public_type)"/>
<property name="second_local" expression=
  "json-eval($.second_local)"/>
<property name="baseUrl" expression="json-eval($.baseUrl)"/>
<enturma.init>
  <baseUrl>{$ctx:baseUrl}</baseUrl>
</enturma.init>
<switch source="get-property('transport', 'Action')">
  <case regex="urn:getReport">
    <enturma.getReport>
      <year>{$ctx:year}</year>
      <state>{$ctx:state}</state>
      <grade>{$ctx:grade}</grade>
      <test_type>{$ctx:test_type}</test_type>
      <public_type>{$ctx:public_type}</public_type>
      <local>{$ctx:local}</local>
    </enturma.getReport>
  </case>
  <case regex="urn:getRanking">
    <enturma.getRanking>
      <year>{$ctx:year}</year>
      <grade>{$ctx:grade}</grade>
    </enturma.getRanking>
  </case>
  <case regex="urn:compareReports">
    <enturma.compareReports>
      <grade>{$ctx:grade}</grade>
      <first_year>{$ctx:first_year}</first_year>
      <first_state>{$ctx:first_state}</first_state>
      <first_test_type>{$ctx:first_test_type}
      </first_test_type>
      <first_public_type>{$ctx:first_public_type}
      </first_public_type>
      <first_local>{$ctx:first_local}</first_local>
      <second_year>{$ctx:second_year}</second_year>
      <second_state>{$ctx:second_state}</second_state>
      <second_test_type>{$ctx:second_test_type}
      </second_test_type>
    </enturma.compareReports>
  </case>
</switch>

```

```
        <second_public_type>{$ctx:second_public_type}
        </second_public_type>
        <second_local>{$ctx:second_local}</second_local>
    </enturma.compareReports>
</case>
</switch>
<respond/>
</inSequence>
<outSequence>
    <log/>
    <send/>
</outSequence>
</target>
<description/>
</proxy>
```