

# Treinamento SOA com IIB

*Nível II*

**Março/2018**

CONTROLE DE VERSÕES

---

<b><u>Versão</u></b>	<b><u>Data</u></b>	<b><u>Autor (es)</u></b>	<b><u>Descrição das Alterações</u></b>
1.0	03/2018	Sergio Fonseca da Silva	Criação

## ÍNDICE

1	APRESENTAÇÃO .....	4
2	FRAGMENTANDO UM SERVIÇO.....	5
2.1	CONSTRUINDO SERVIÇO DO TIPO REST API. ....	5
2.2	IMPLEMENTANDO A CAMADA DE VALIDAÇÃO. ....	9
2.3	IMPLEMENTANDO A CAMADA ADAPTER – REQUEST.....	15
2.4	IMPLEMENTANDO A CAMADA DE ENABLE. ....	20
2.5	IMPLEMENTANDO A CAMADA ADAPTER – RESPONSE. ....	24
2.6	IMPLEMENTANDO A CAMADA DE MEDIATOR. ....	27
2.7	IMPLEMENTANDO O SUBFLOW PRINCIPAL.....	30
2.8	CONCLUSÃO .....	31

## 1 Apresentação

Após a familiarização com IIB e assimilado os conceitos básicos para utilização dos nodes e desenvolvimento de serviços. Será apresentado o modelo de estrutura corporativa, onde ocorre a separação de camadas por responsabilidade e contexto.

Para uma boa atuação nesse modelo de arquitetura de desenvolvimento é necessário a definição de alguns padrões de nomenclaturas que sejam claros para permitir a identificação da funcionalidade do componente e o contexto o qual o mesmo pertence, o quais serão apresentados mediante a evolução do treinamento.

Para isso, além dos conceitos básicos já apresentados no treinamento anterior, vamos construir broker schemas, mecanismo que irá permitir esse modelo de desenvolvimento, ainda existem outros recursos mais sofisticados que faz parte do escopo do próximo treinamento nível III.

Evidentemente que o modelo que veremos a seguir, requer aproximadamente entre 10 a 20 por cento de tempo a mais para implementação se comparado ao modelo tradicional de desenvolvimento baseado em entrega imediata de serviço.

Entretanto serão apresentados diversos benefícios que devem serem considerados pensando em curto e médio e longa duração de uso dos serviços, que visam a clareza do código, reutilização de fontes, fácil expansão e manutenção dos serviços em ambiente produtivo, sendo assim cabe a empresa decidir a sua adoção seja de forma completa, parcial ou a não aderência ao modelo.

Por se tratar de um curso que requer como pré-requisito a conclusão do curso anterior, focamos mais nas novas funcionalidade e deixaremos para segundo plano detalhes de desenvolvimento que já foram absorvidos no curso do nível I.

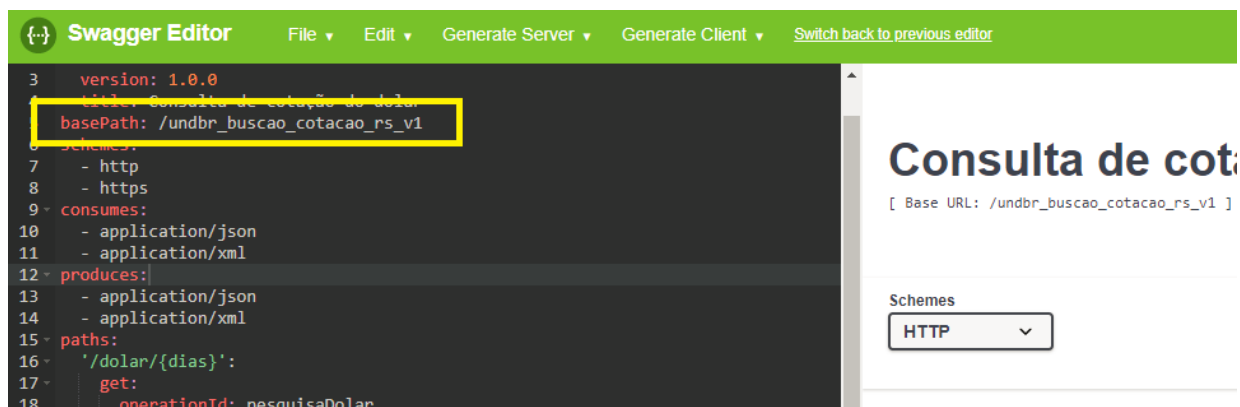
## 2 Fragmentando um serviço.

Tendo como referência o curso anterior, vamos construir o mesmo serviço de cotação de dólar, entretanto vamos adotar o novo modelo de arquitetura de desenvolvimento, dessa forma teremos um comparativo em relação ao modelo tradicional e a metodologia apresentada.

### 2.1 Construindo serviço do tipo REST API.

A primeira etapa em rumo ao novo padrão de desenvolvimento é reutilizarmos o Swagger adotado na apostila anterior, entretanto vamos alterar o basepath do mesmo com uma padronização de nomenclatura que será detalhada mais adiante, por hora faremos:

- 2.1.1. No browser, abrir o arquivo Swagger no editor on-line.
- 2.1.2. Altere o basePath para: **/undbr\_buscao\_cotacao\_rs\_v1**.
- 2.1.3. Realize o download do mesmo.
- 2.1.4. Disponibilize na pasta Swagger local da máquina.
- 2.1.5. Renomeei o nome do arquivo json conforme sua convencia, mas que tenha sentido em relação ao projeto.



**Importante:**

Nunca coloque a versão no nome do arquivo Swagger, o fato de trocar o nome o IIB perde a referência nos mapeamentos dos fluxos.

- 2.1.6. No **Workspace**, selecione **File, New, REST API**.
- 2.1.7. Aplicando a regra de nomenclatura ao projeto.

Vamos sugerir um padrão de nomenclatura para nome de projetos, em Name digite: **undbr\_busca\_cotacao\_rs\_v1**, sendo:

**und** = Nome da empresa proprietário do projeto no caso é a abreviação de Unimed.

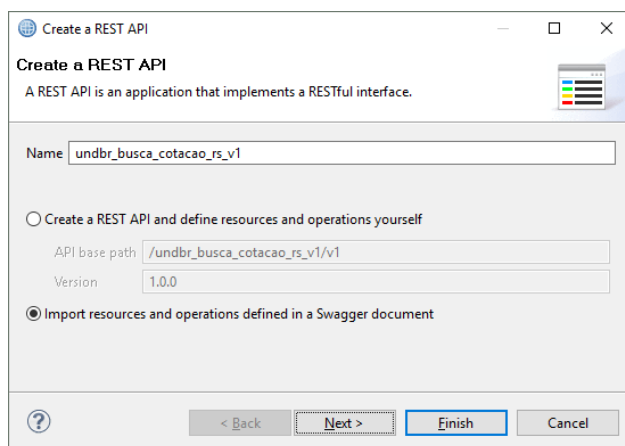
**br** = Pais de origem do desenvolvimento, no caso Brasil.

**\_** = Traço baixo, justificado porque alguns projetos do IIB não aceitam traço normal no nome do mesmo.

**busca\_cotacao** = Nome do projeto iniciado com letra minúscula, respeitando a separação de nome com traço baixo.

**rs** = Tipo do projeto no caso REST.

**v1** = Versão do serviço.



Create a REST API

A REST API is an application that implements a RESTful interface.

Name

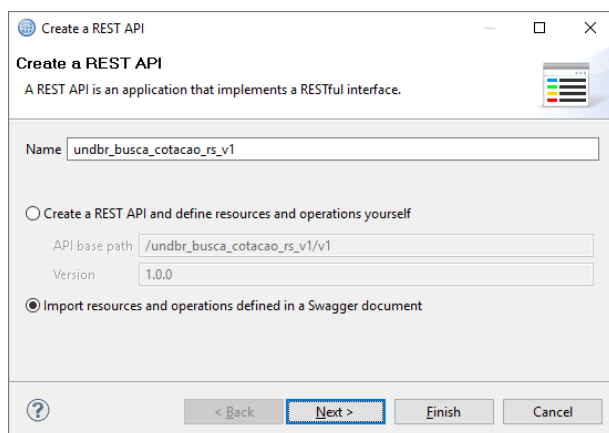
☐ Create a REST API and define resources and operations yourself

API base path

Version

☒ Import resources and operations defined in a Swagger document

#### 2.1.8. Selecione **Import Resources...**



Create a REST API

A REST API is an application that implements a RESTful interface.

Name

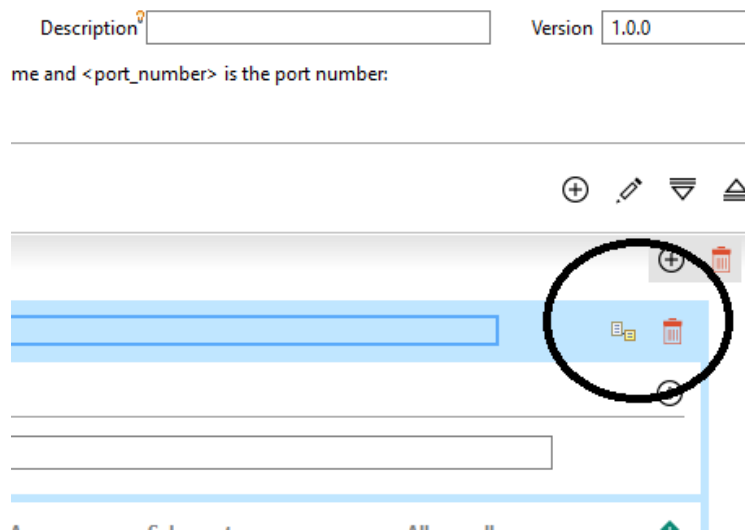
☐ Create a REST API and define resources and operations yourself

API base path

Version

☒ Import resources and operations defined in a Swagger document

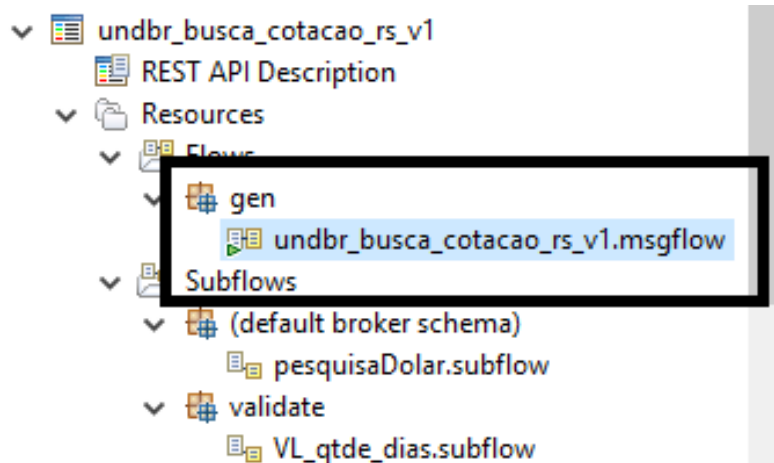
### 2.1.9. Implementando o *msgflow*.



2.1.10. Implementando os tratamentos de exceção.

2.1.11. Duplo clique no flow ***undbr\_busca\_cotacao\_rs\_v1***.

2.1.12. Role a página para baixo no arquivo json ate o bloco ***Error Handling***.

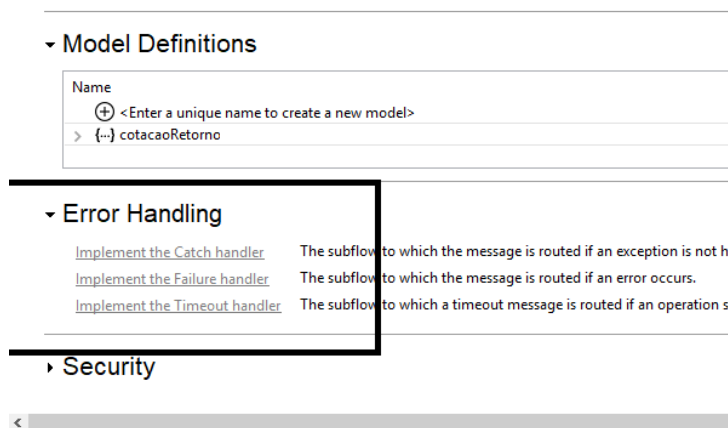


2.1.13. Execute duplo clique nos links abaixo, para criar os subflow correspondentes:

**2.1.13.1. *Implements the Cactch handler.***

**2.1.13.2. *Implements the Failure handler.***

**2.1.13.3. *Implements the Timeout handler.***



**Observação:**

No momento oportuno vamos implementar os fluxos de mensagens para tratamento de erro.



## 2.2 Implementando a camada de validação.

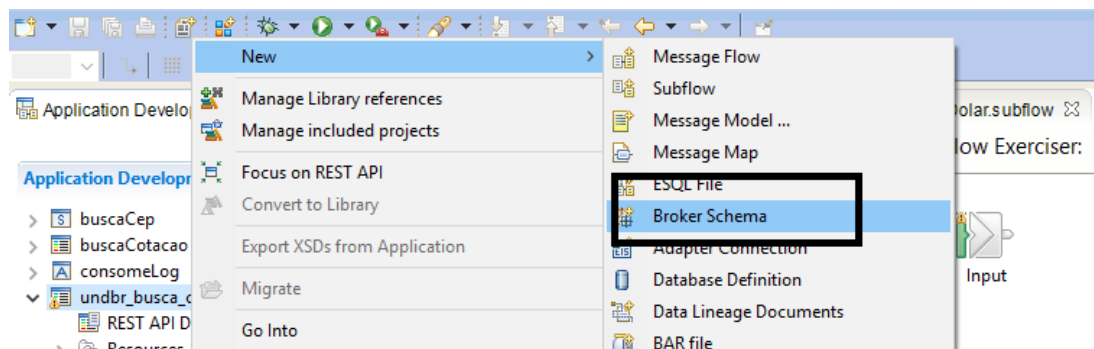
A primeira atividade que o serviço de integração deve realizar é a validação dos dados de entrada, dessa forma caso exista alguma inconformidade com os mesmos o serviço notifica o consumidor e suspende a execução do serviço, economizando processamento desnecessário.

Dando ênfase no nosso modelo de desenvolvimento vamos criar um Broker Schema que é utilizado para definirmos a organização estrutural de nosso serviço.

2.2.1. Selecione o projeto.

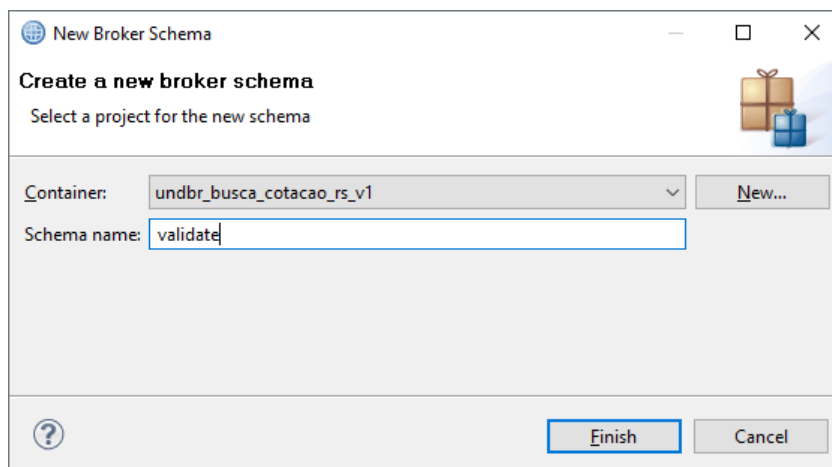
2.2.2. Clique em **New**

2.2.3. Clique em **Broker Schema**.

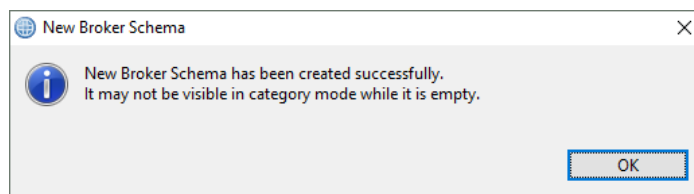


2.2.4. Certifique que o **container** selecionado e o serviço que estamos definindo.

2.2.5. Em **Schema name**, digite o nome de nosso pacote: **validade**.



2.2.6. Ao clicar em **Finish**, será apresentada a seguinte mensagem.

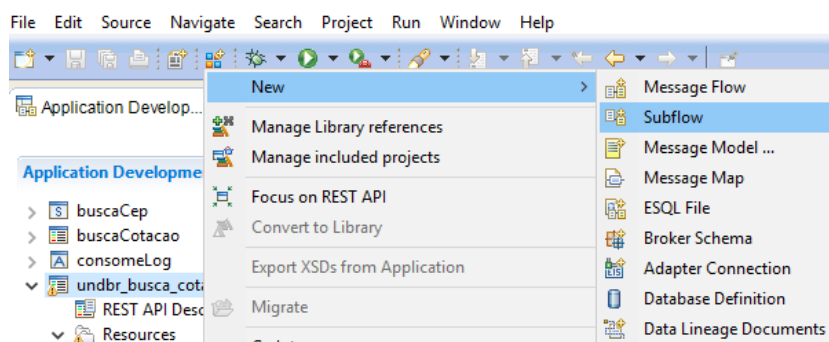


Agora vamos construir nosso subflow de validação, essa validação consiste em verificar se o consumidor do serviço enviou o parâmetro de entrada correspondente a quantidade de dias, caso negativo vamos interromper o fluxo e notificar o consumidor. Para isso:

2.2.7. Selecione o projeto.

2.2.8. Clique em **New**.

2.2.9. Clique em **Subflow**.



2.2.10. Siga a seguinte regra de nomenclatura.

Vamos sugerir um padrão de nomenclatura para subflow, em Name digite: **VL\_qtde\_dias**, sendo:

**VL** = Convencionado informar duas letras grafadas em maiúsculo, para identificam o **pattern**, nesse caso VL significa **Validate**.

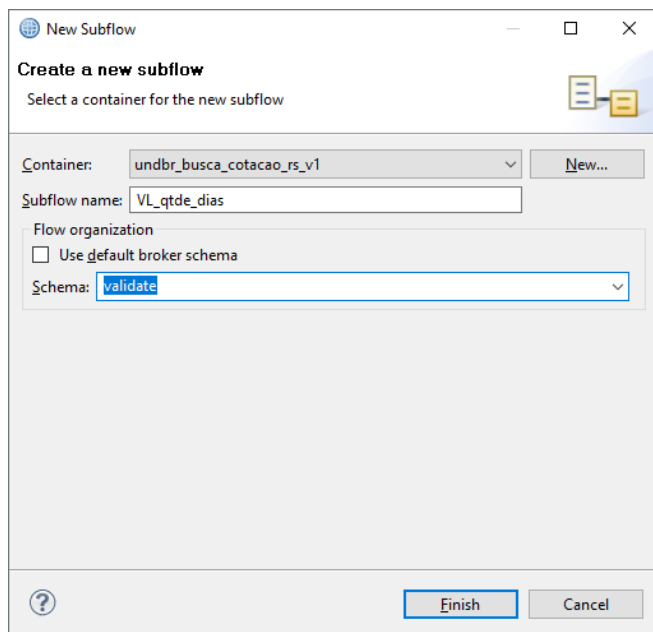
**\_** = Traço baixo separando palavras.

**qtde\_dias** = Nome que identifica o objetivo da validação, nunca utilizar abreviações que possam comprometer o entendimento e o objetivo do subflow.

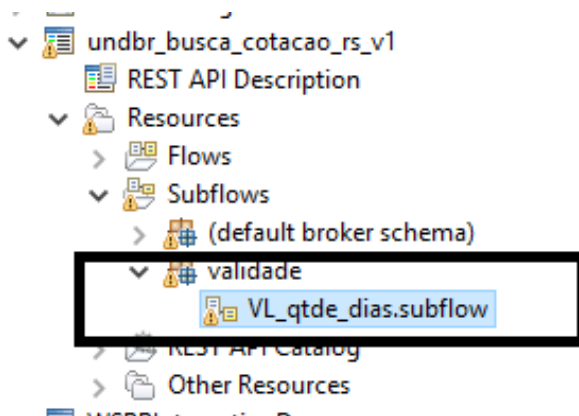
2.2.11. Certifique que o Container selecionado é o serviço em implementação.

2.2.12. Digite o seguinte nome para o subflow: **VL\_qtadeDias**

- 2.2.13. Deselecione a opção: **Use default broker schema.**
- 2.2.14. Em schema selecione o broker schema que criamos, nesse caso: **validate.**
- 2.2.15. Clique em **Finish.**



- 2.2.16. Ao concluir as etapas teremos:



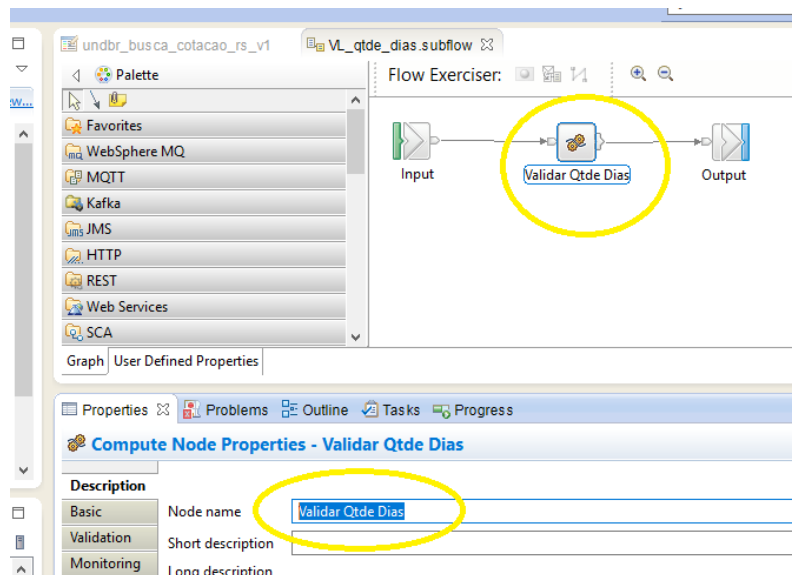
**Observação:**

Criamos um Broker Schema especializado para validação, e colocamos o subflow dentro dessa estrutura, esse procedimento será repetido em todas atividades que forem necessários para criarmos esse tipo de organização de projeto.

**Validando uma requisição.**

O serviço de cotação recebe um parâmetro de entrada, esse valor corresponde a quantidade dias que será retornado na requisição, portanto precisamos limitar a quantidade de cotações que será retornada em 30 dias. Portanto vamos implementar um node compute (ESQL), para realizar a validação do parâmetro de entrada, para isso:

- 2.2.17. Duplo clique no subflow **VL\_qtde\_dias**, para abrir a área de trabalho.
- 2.2.18. Selecione o node de compute e arraste para área de trabalho.
- 2.2.19. Selecione aba de **description** e em node name informe: **Validar Qtde Dias**.



- 2.2.20. Duplo clique no node de **compute node**, será aberto a área de edição dos comandos **ESQL**.
- 2.2.21. Entre o begin e o end, implemente o seguinte código **ESQL** de validação.

```
SET OutputRoot = InputRoot;
SET OutputLocalEnvironment = InputLocalEnvironment;

IF CAST(InputLocalEnvironment.REST.Input.Parameters.dias as INT) > 30 THEN
    SET OutputRoot.JSON.Data.error = 'Qtidade dias informado, superior ao numero máximo permitido!';
END IF;

RETURN TRUE;
```

2.2.22. Abaixo código final do ESQL no node de compute.

```
ESQL VL_qtde_dias_Compute.esql
BROKER SCHEMA validate

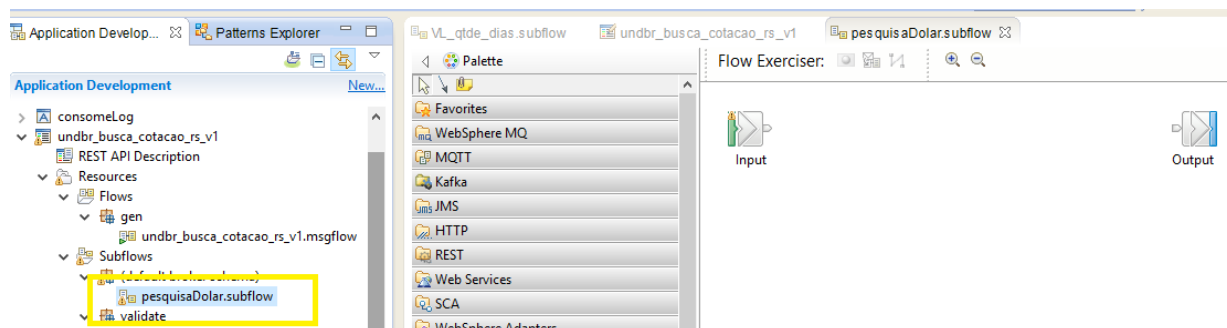
CREATE COMPUTE MODULE VL_qtde_dias_Compute
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN
    SET OutputRoot = InputRoot;
    SET OutputLocalEnvironment = InputLocalEnvironment;

    IF CAST(InputLocalEnvironment.REST.Input.Parameters.dias as INT) > 30 THEN
        THROW USER EXCEPTION MESSAGE 400 VALUES (601, 'Qtidade dias informado, superior ao numero máximo');
    END IF;

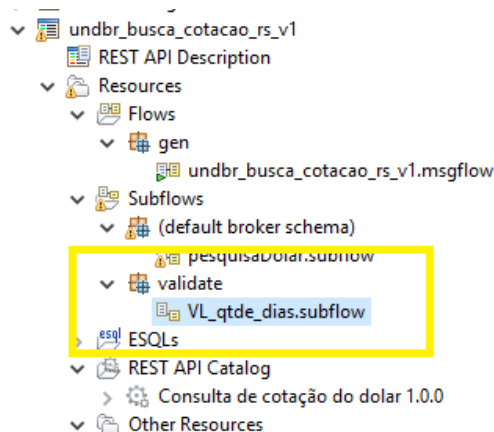
    RETURN TRUE;
END;
END MODULE;
```

Com as etapas anterior implementamos o subflow de validação, porem precisamos disponibilizar o mesmo no flow principal da aplicação, então:

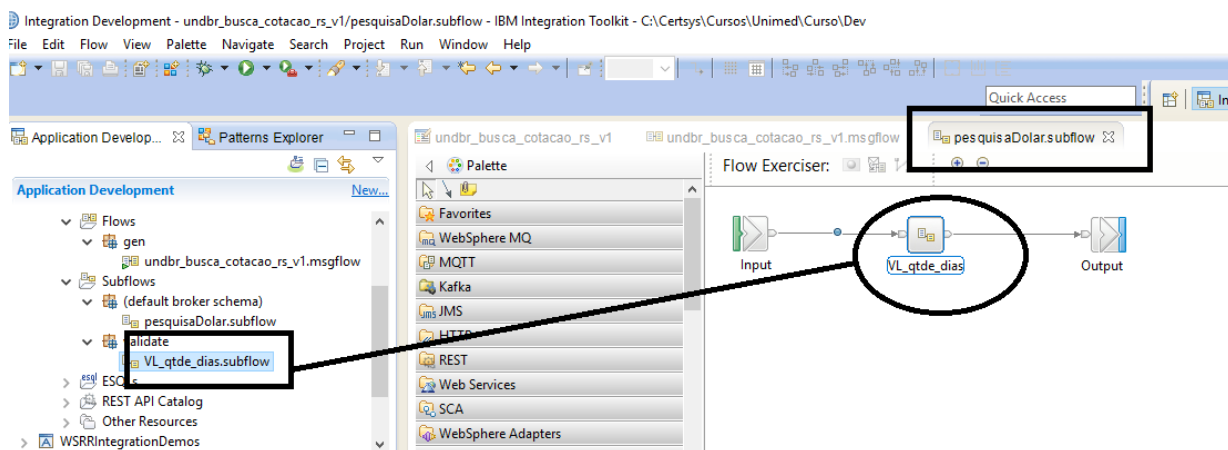
2.2.23. Na lateral esquerda do toolkit, navegue até encontrar o subflow “**pesquisaDolar.subflow**”, clique duas vezes para abrir a área de desenvolvimento.



2.2.24. Ainda na lateral esquerda do toolkit, navegue até encontrar o subflow “**VL\_qtde\_dias**”.



2.2.25. Selecione a estrutura Subflows “**VL\_qtde\_dias**” e araste para a área de trabalho entre os nodes input e output.



2.2.26. Realize as ligações entre os nodes.

## 2.3 Implementando a camada adapter – request.

A camada de adaptação será responsável em normalizar os dados, sempre ocorrem em dois sentidos, quando o serviço do barramento é invocado pelo consumidor, recebe essas informações no formato que foi exposto pelo barramento, em alguns cenários utiliza-se o modelo canônico como exposição.

Entretanto o modelo canônico que é interno da empresa, raramente é equivalente ao formato de entrada de dados do serviço do legado, portanto realizamos a adaptação dos dados para o formato esperado pelo provedor, esse é o sentido do consumidor para o barramento, ou seja: o sentido de entrada de dados no barramento.

O mesmo ocorre quando os dados são retornados pelo provedor, esses dados são enviados no formato que o legado desenvolveu o serviço, o que também raramente é igual ao formato que foi definido no barramento, sendo canônico ou não canônico, esse é o sentido de retorno, ou seja: do provedor para o barramento.

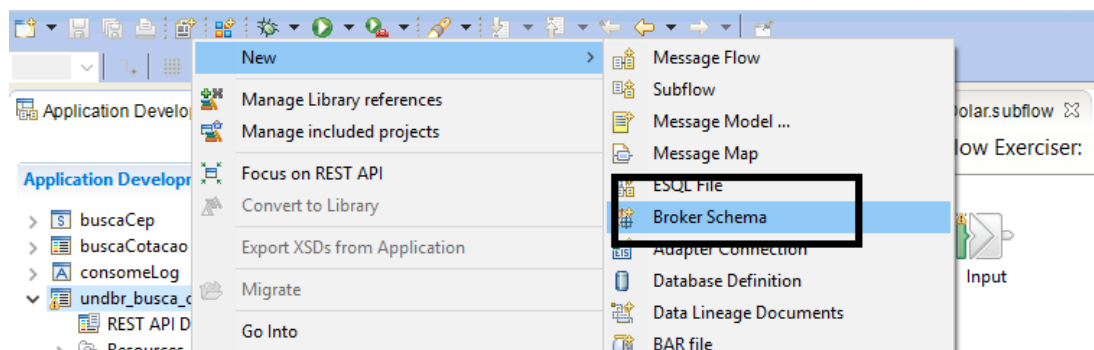
Logo teremos mapeamento do consumidor para o barramento e do barramento para o provedor e o inverso do provedor para o barramento e do barramento para o consumidor.

Vamos construir nosso primeiro adapter que corresponde no sentido de entrada, ou seja do consumidor para o barramento, para isso:

2.3.1. Selecione o projeto.

2.3.2. Clique em **New**

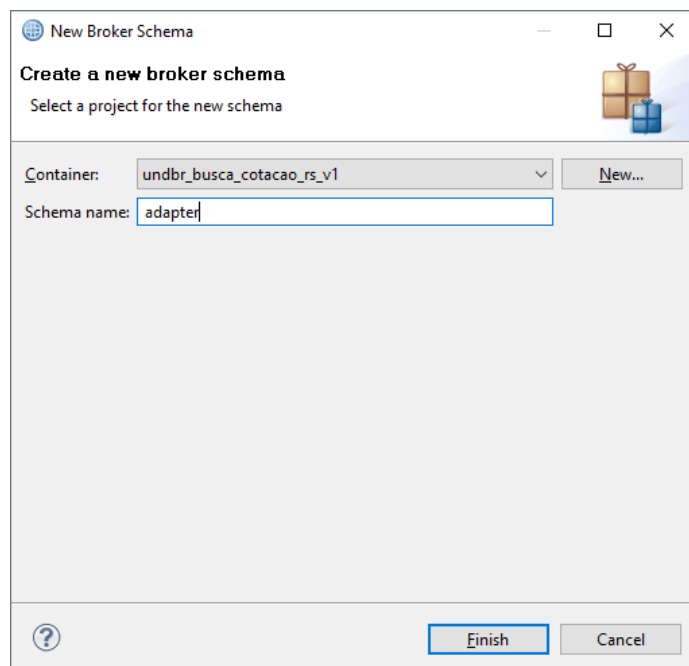
2.3.3. Clique em **Broker Schema**.



2.3.4. Certifique que o **container** selecionado e o serviço que estamos definindo.

2.3.5. Em **Schema name**, digite o nome de nosso pacote: **adapter**.

2.3.6. Clique em **Finish**.

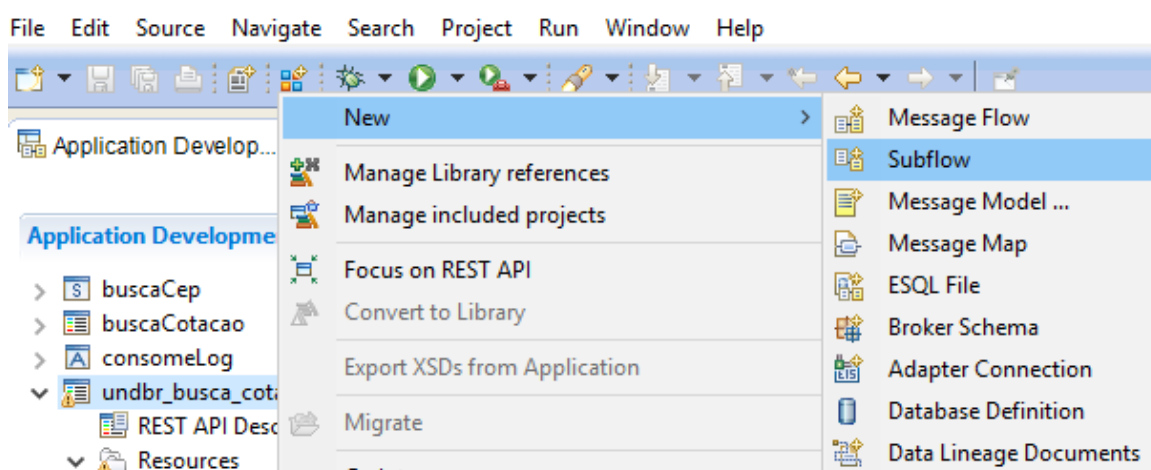


Agora vamos construir nosso subflow de adaptação. Para isso:

2.3.7. Selecione o projeto.

2.3.8. Clique em **New**.

2.3.9. Clique em **Subflow**.





2.3.10. Siga a seguinte regra de nomenclatura.

Vamos sugerir um padrão de nomenclatura para subflow, em Name digite: **AD\_consumer\_to\_provider**, sendo:

**AD** = Convencionado informar duas letras grafadas em maiúsculo, para identificam o **pattern**, nesse caso AD significa **Adapter**.

**\_** = Traço baixo separando palavras.

**consumer\_to\_provider** = Nome que identifica o objetivo do mapeamento, nesse caso **"consumer\_to\_provider"**, nunca utilizar abreviações que possam comprometer o entendimento e o objetivo do subflow.

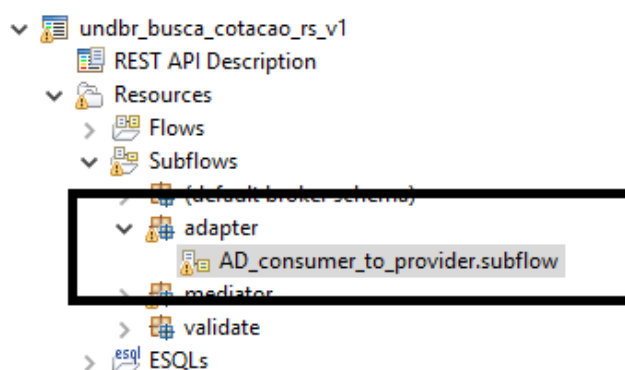
2.3.11. Certifique que o Container selecionado é o serviço em implementação.

2.3.12. Digite o seguinte nome para o subflow: **AD\_consumer\_to\_provider**.

2.3.13. Deselecione a opção: **Use default broker schema**.

2.3.14. Em schema selecione o broker schema que criamos, nesse caso: **adapter**.

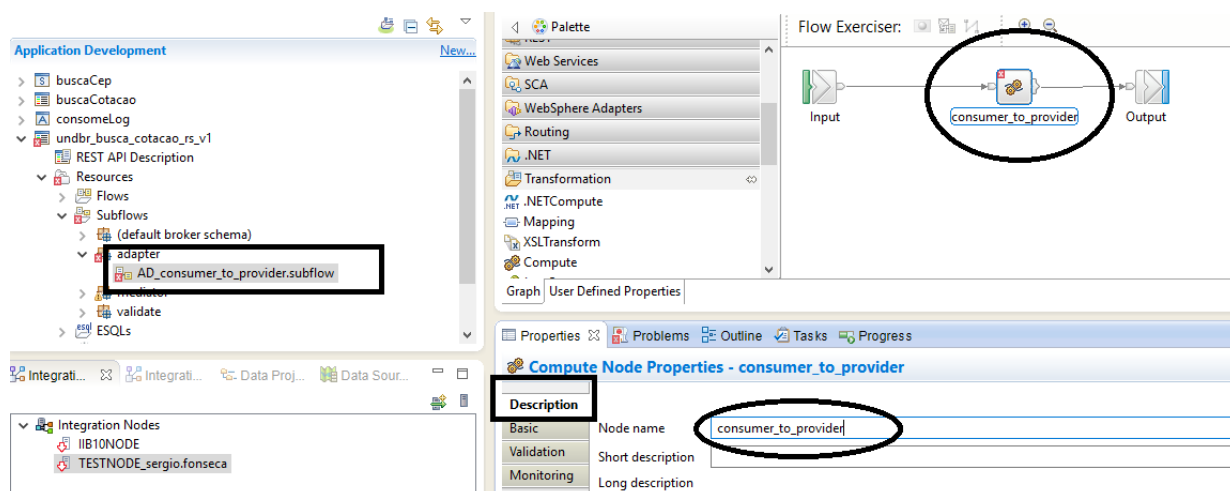
2.3.15. Clique em **Finish**.



Vamos implementar o subflow de adaptação, para isso como não usamos modelo canônico, basta utilizarmos o node de ESQL.

2.3.16. Dublo clique no subflow **AD\_consumer\_to\_provider**.

2.3.17. Arraste o node de compute para a área de trabalho.



2.3.18. Implemento o código ESQL abaixo entre o **begin** e o **end**.

```
-- Endpoint do provedor que teremos que acessar
-- http://api.bcb.gov.br/dados/serie/bcdata.sgs.1/dados/ultimos/10?formato=json
--
-- Atribuindo os dados de entrada do node com os dados de saída do node,
-- utilizado para filtrar os dados que será enviados para a próxima etapa
--
SET OutputRoot = InputRoot;
SET OutputLocalEnvironment = InputLocalEnvironment;
--
-- Setando content-type
--
-- Declarando variáveis
--
DECLARE uri_parte1 CHAR;
DECLARE dias CHAR;
DECLARE uri_parte2 CHAR;
DECLARE endpoint CHAR;
--
-- Concatenando a URL para compor o endpoint
--
SET uri_parte1 = 'http://api.bcb.gov.br/dados/serie/bcdata.sgs.1/dados/ultimos/';
SET dias = InputLocalEnvironment.REST.Input.Parameters.dias;
SET uri_parte2 = '?formato=json';
SET endpoint = uri_parte1 || dias || uri_parte2;
--
-- Setando o endpoint no node de HTTP Request
--
SET OutputLocalEnvironment.Destination.HTTP.RequestURL = endpoint;
RETURN TRUE;
```

### 2.3.19. Não esqueça de setar o compute mode.

The screenshot displays the IBM Integration Bus (IIB) Application Development console. The left pane shows the project structure under 'Application Development', with 'AD\_consumer\_to\_provider.subflow' selected under the 'adapter' folder. The right pane shows the 'Flow Exerciser' for 'AD\_consumer\_to\_provider\_Compute.esql', highlighting the 'consumer\_to\_provider' node. Below the flow exerciser, the 'Properties' pane is open, showing the 'Compute Node Properties - consumer\_to\_provider'. The 'Basic' tab is selected, and the 'Compute mode\*' property is set to 'LocalEnvironment and Message'.

**Properties - Compute Node Properties - consumer\_to\_provider**

Property	Value
Basic	
Validation	Connect before flow starts
Monitoring	Transaction*
ESQL module	(adapter)AD_consumer_to_provider_Compute
Compute mode*	LocalEnvironment and Message

## 2.4 Implementando a camada de enable.

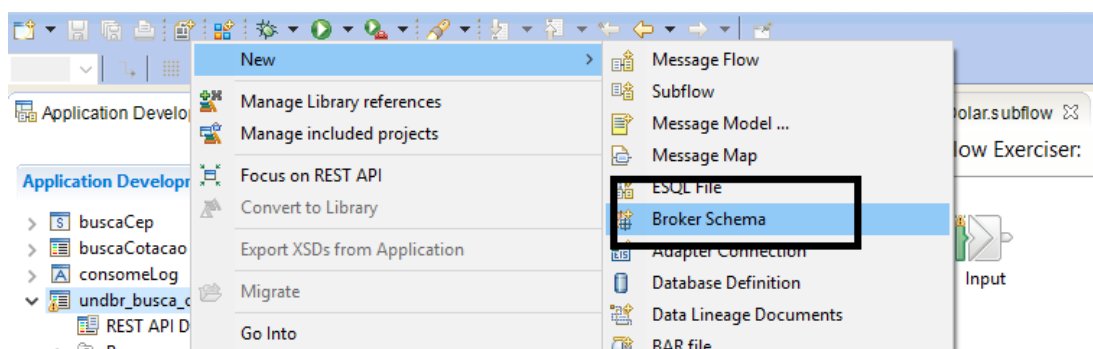
A camada de enable será responsável em realizar as chamadas atômicas aos serviços expostos pelos legados.

Vamos construir nosso primeiro enable, para isso:

2.4.1. Selecione o projeto.

2.4.2. Clique em **New**

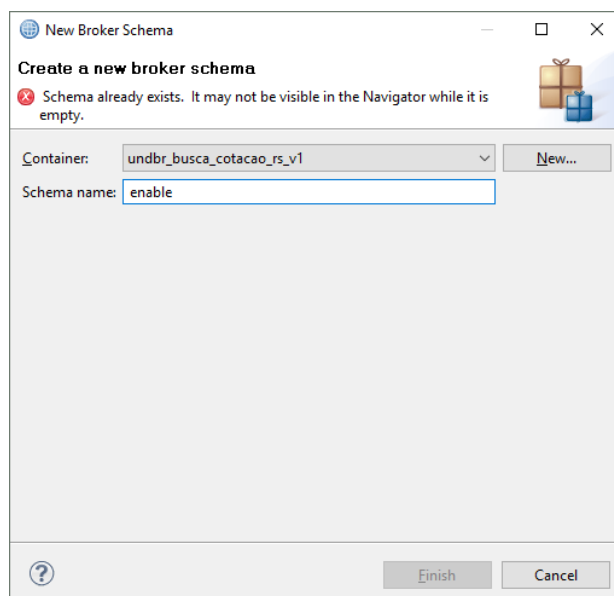
2.4.3. Clique em **Broker Schema**.



2.4.4. Certifique que o **container** selecionado e o serviço que estamos definindo.

2.4.5. Em **Schema name**, digite o nome de nosso pacote: **enable**.

2.4.6. Clique em **Finish**.

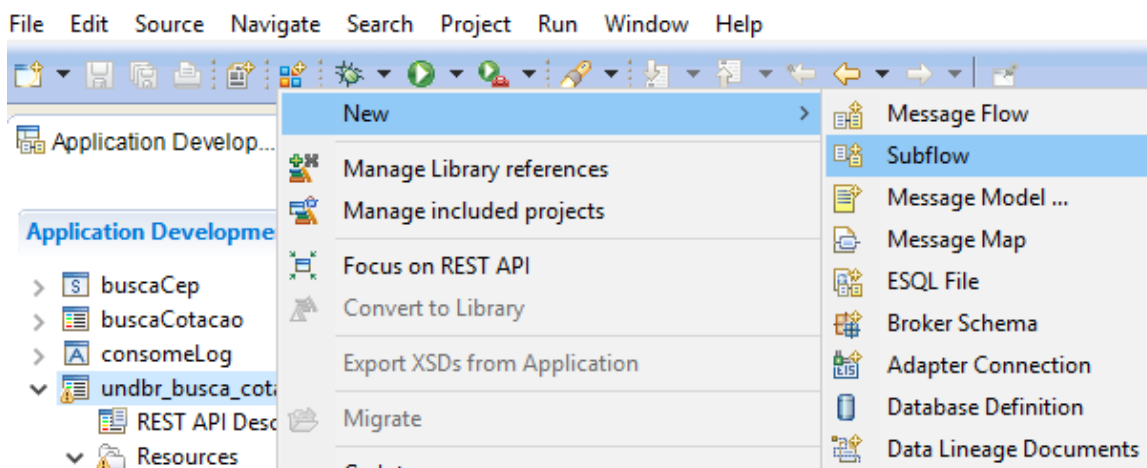


Agora vamos construir nosso subflow de enable. Para isso:

2.4.7. Selecione o projeto.

2.4.8. Clique em New.

2.4.9. Clique em Subflow.



2.4.10. Siga a seguinte regra de nomenclatura.

Vamos sugerir um padrão de nomenclatura para subflow, em Name digite: **EN\_pesquisaDolar**, sendo:

**EN** = Convencionado informar duas letras grafadas em maiúsculo, para identificam o **pattern**, nesse caso EN significa **Enable**.

**\_** = Traço baixo separando palavras.

**pesquisaDolar** = Nome que identifica o objetivo do mapeamento, nesse caso a operação/verbo **“pesquisaDolar”**, nunca utilizar abreviações que possam comprometer o entendimento e o objetivo do subflow.

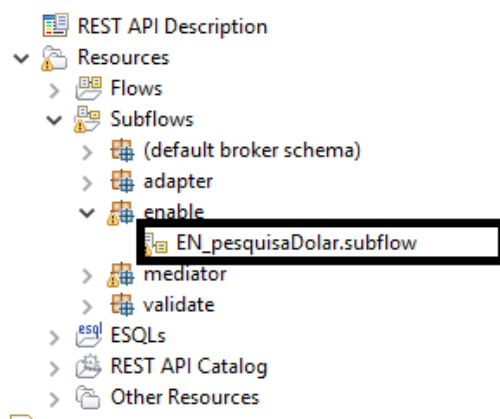
2.4.11. Certifique que o Container selecionado é o serviço em implementação.

2.4.12. Digite o seguinte nome para o subflow: **EN\_pesquisaDolar**.

2.4.13. Deselecione a opção: **Use default broker schema**.

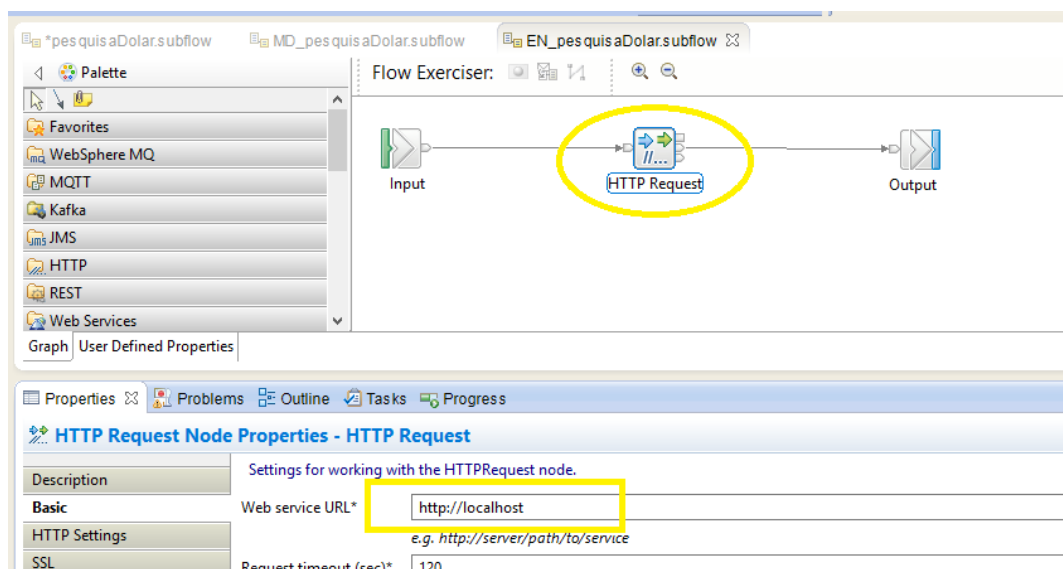
2.4.14. Em schema selecione o broker schema que criamos, nesse caso: **enable**.

2.4.15. Clique em **Finish**.

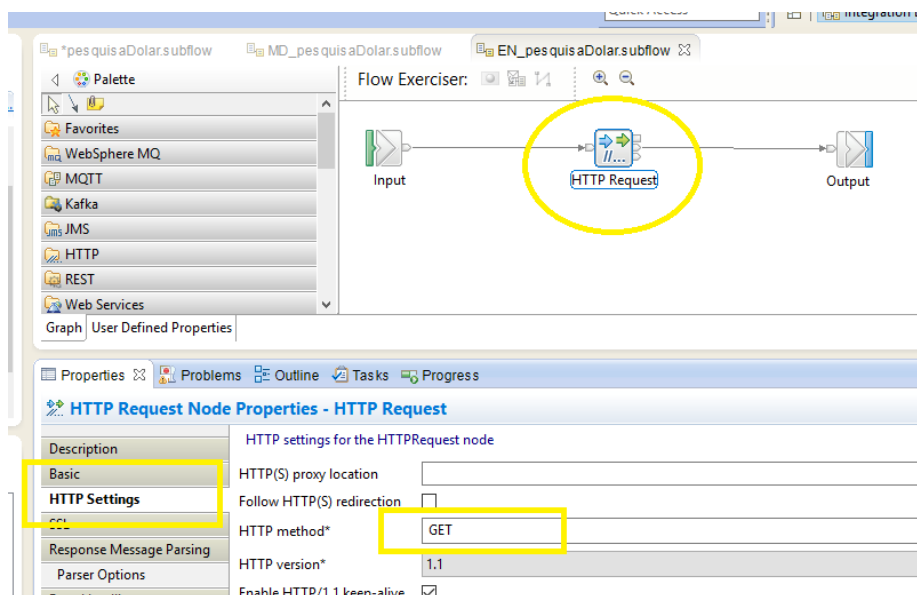


2.4.16. Dublo clique no subflow **EN\_pesquisaDolar.subflow**.

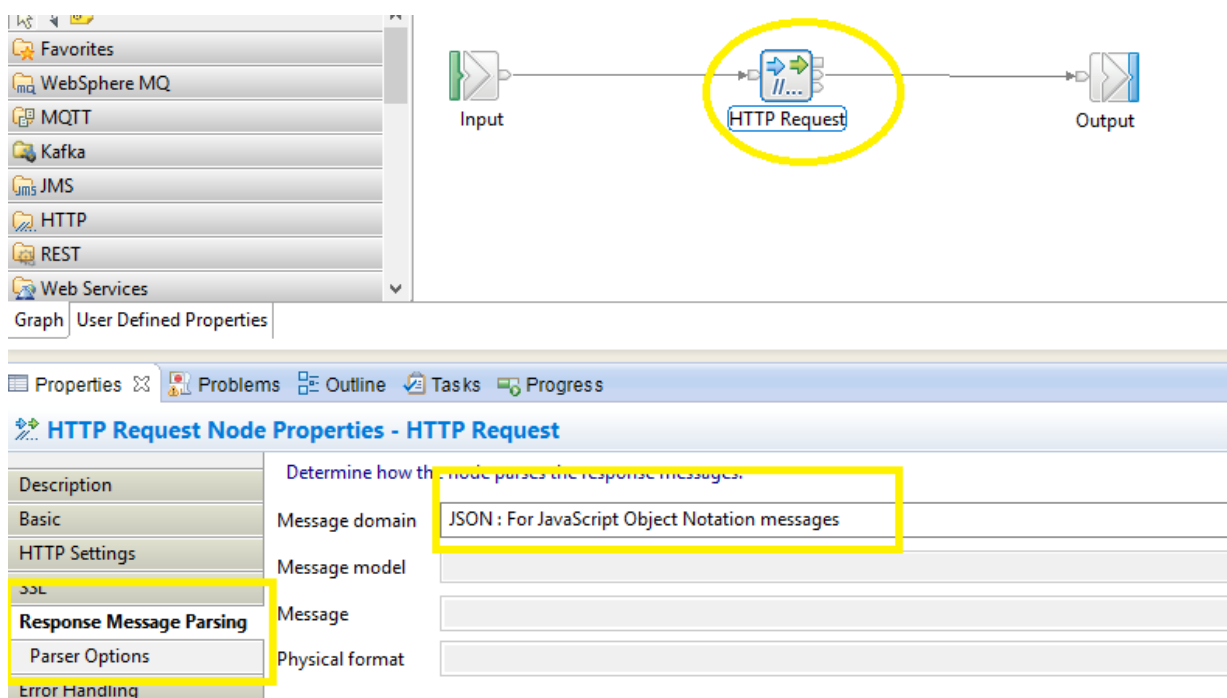
2.4.17. Em **Properties**, selecione **Basic** e preencha o campo **Web Service URL** com o valor **http://localhost**.



- 2.4.18. Em **Properties**, selecione **HTTP Settings** e selecione o campo **HTTP Method** selecione **GET**.



- 2.4.19. Em **Properties**, selecione **Response Message Parsing** e preencha o campo **Message Domain** selecione **JSON: For JavaScript...**



## 2.5 Implementando a camada adapter – response.

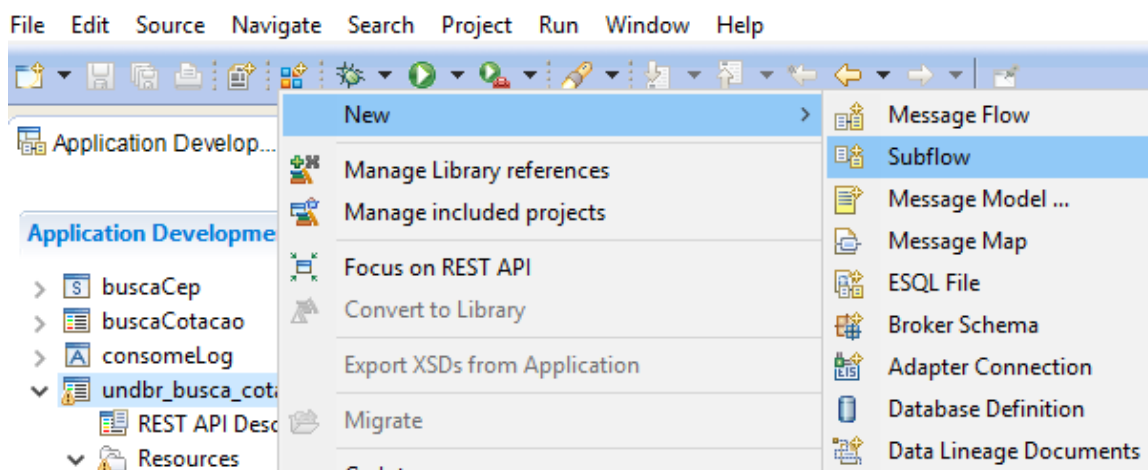
Temos agora que retornar os dados da requisição que o barramento realizou no legado, assim será necessário realizar o mapeamento de resposta, entenda que é a resposta do legado que devemos responder para o consumidor.

Agora vamos construir nosso subflow de adaptação para a resposta. Para isso:

2.5.1. Selecione o projeto.

2.5.2. Clique em **New**.

2.5.3. Clique em **Subflow**.



2.5.4. Siga a seguinte regra de nomenclatura.

Vamos sugerir um padrão de nomenclatura para subflow, em Name digite: **AD\_provider\_to\_consumer**, sendo:

**AD** = Convencionado informar duas letras grafadas em maiúsculo, para identificam o **pattern**, nesse caso AD significa **Adapter**.

**\_** = Traço baixo separando palavras.

**provider\_to\_consumer** = Nome que identifica o objetivo do mapeamento, nesse caso **“provider\_to\_consumer”**, nunca utilizar abreviações que possam comprometer o entendimento e o objetivo do subflow.

2.5.5. Certifique que o Container selecionado é o serviço em implementação.

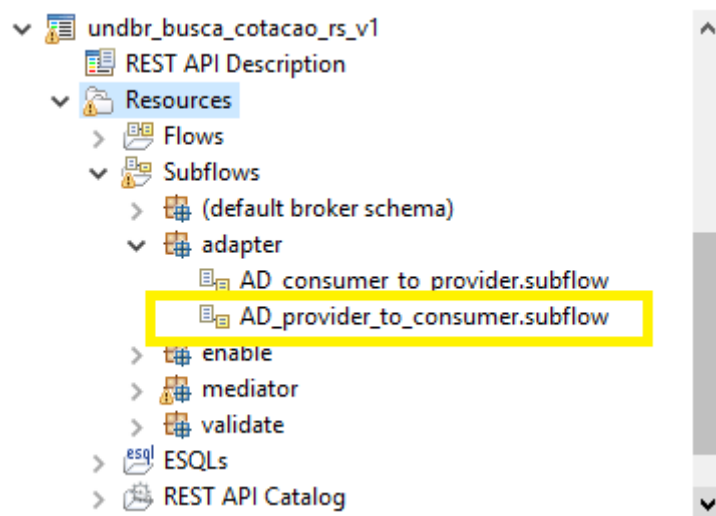
2.5.6. Digite o seguinte nome para o subflow: **AD\_provider\_to\_consumer**.

2.5.7. Deselecione a opção: **Use default broker schema**.

2.5.8. Em schema selecione o broker schema que criamos, nesse caso: **adapter**.



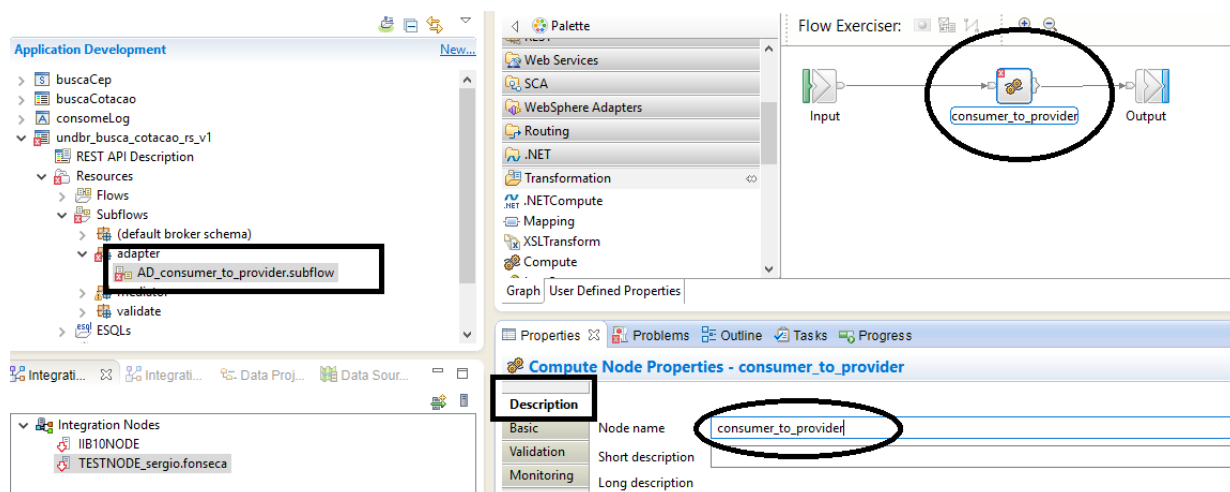
2.5.9. Clique em **Finish**.



Vamos implementar o subflow de adaptação, para isso como não usamos modelo canônico, basta utilizarmos o node de ESQL.

2.5.10. Dublo clique no subflow **AD\_provider\_to\_consumer**.

2.5.11. Arraste o node de compute para a área de trabalho.



### 2.5.12. Implemento o código ESQL abaixo entre o **begin** e o **end**.

```
DECLARE I INTEGER CARDINALITY (InputRoot.JSON.Data.*[]);  
DECLARE J INTEGER 1;  
  
WHILE J <= I DO  
    SET OutputRoot.JSON.Data.cotacaoRetorno[J].data = InputRoot.JSON.Data.Item[J].data;  
    SET OutputRoot.JSON.Data.cotacaoRetorno[J].valor = InputRoot.JSON.Data.Item[J].valor;  
  
    SET J = J + 1;  
END WHILE;  
  
RETURN TRUE;
```

### 2.5.13. Não esqueça de setar o compute mode.

The screenshot displays the IBM Integration Bus Studio interface. On the left, the 'Application Development' tree shows a project named 'undbr\_busca\_cotacao\_rs\_v1' with a subflow 'AD\_consumer\_to\_provider.subflow' highlighted. The main workspace shows a flow diagram with an 'Input' node, a 'consumer\_to\_provider' node (circled in black), and an 'Output' node. Below the flow diagram, the 'Properties' panel is open for the 'Compute Node Properties - consumer\_to\_provider'. The 'Basic' tab is selected, and the 'Compute mode\*' property is set to 'LocalEnvironment and Message' (highlighted with a black box). Other properties visible include 'Data source', 'Connect before flow starts', 'Transaction\*', 'ESQL module', and 'Adapted: AD\_consumer\_to\_provider\_Compute'.

## 2.6 Implementando a camada de mediator.

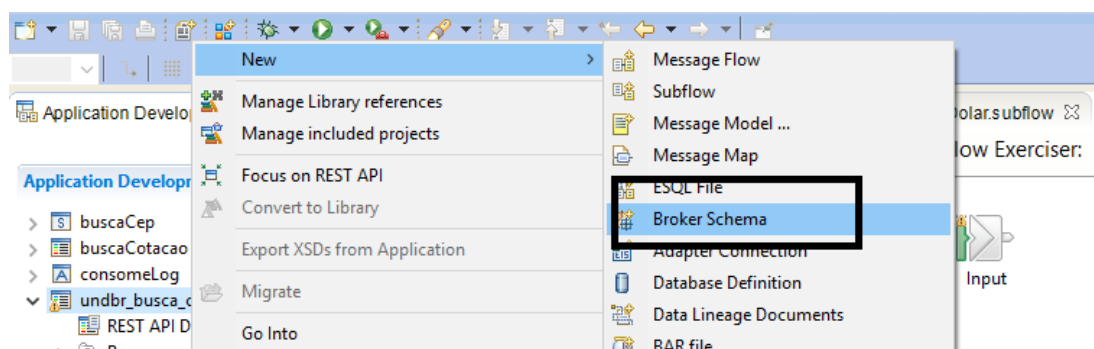
Agora que criamos todas as camadas necessários do projeto, podemos implementar a camada mediator, camada essa que tem a responsabilidade de mediar a requisição que chega no barramento com as demais camadas ou subflows.

Dando ênfase no nosso modelo de desenvolvimento vamos criar um Broker Schema que é utilizado para definirmos a organização estrutural de nosso serviço.

2.6.1. Selecione o projeto.

2.6.2. Clique em **New**

2.6.3. Clique em **Broker Schema**.



2.6.4. Certifique que o container selecionado e o serviço que estamos definindo.

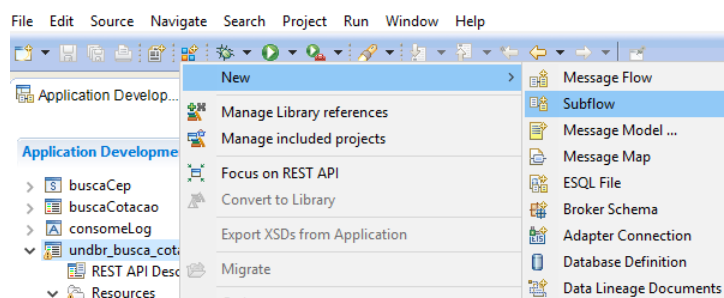
2.6.5. Em **Schema name**, digite o nome de nosso pacote: **mediator**.

Agora vamos construir nosso subflow de mediação. Para isso:

2.6.6. Selecione o projeto.

2.6.7. Clique em **New**.

2.6.8. Clique em **Subflow**.



2.6.9. Siga a seguinte regra de nomenclatura.

Vamos sugerir um padrão de nomenclatura para subflow, em Name digite: **MD\_pesquisaDolar**, sendo:

**MD** = Convencionado informar duas letras grafadas em maiúsculo, para identificam o **pattern**, nesse caso MD significa **Mediator**.

**\_** = Traço baixo separando palavras.

**pesquisaDolar** = Nome que identifica o nome da operação que está sendo implementada nesse caso "**pesquisaDolar**", nunca utilizar abreviações que possam comprometer o entendimento e o objetivo do subflow.

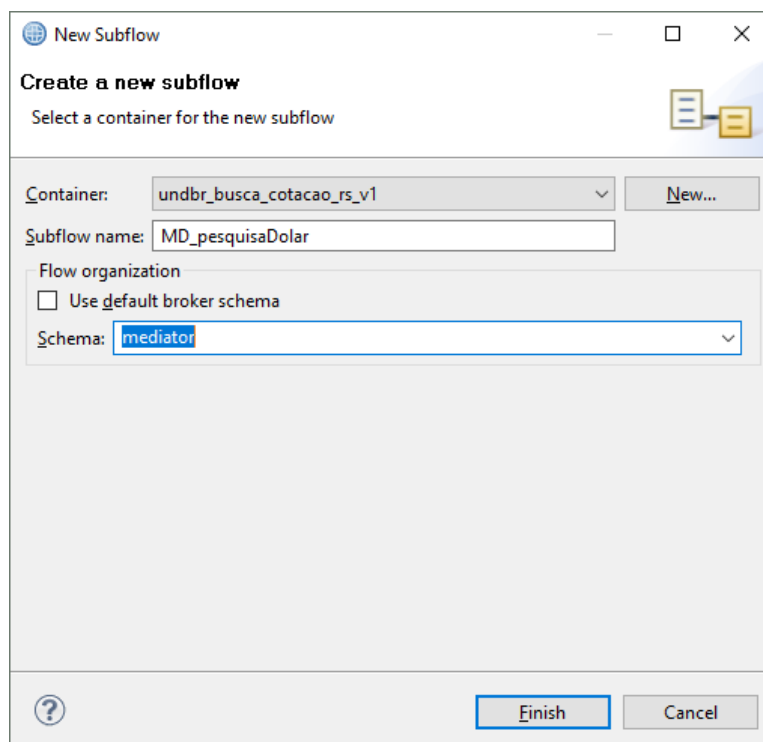
2.6.10. Certifique que o Container selecionado é o serviço em implementação.

2.6.11. Digite o seguinte nome para o subflow: **MD\_pesquisaDolar**.

2.6.12. Deselecione a opção: **Use default broker schema**.

2.6.13. Em schema selecione o broker schema que criamos, nesse caso: mediator.

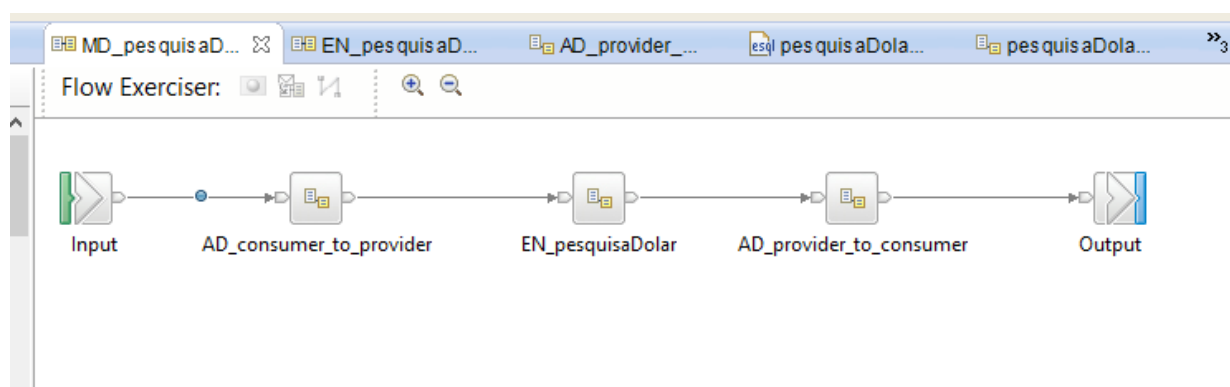
2.6.14. Clique em **Finish**.



Agora que criamos todas as camadas necessários do projeto, podemos implementar a camada mediator, camada essa que tem a responsabilidade de isolar a requisição que chega no barramento com as demais camadas, que entendemos também com os demais subflows.

Nessa camada, temos apenas que disponibilizar os subflows abaixo, para isso, arraste os mesmos para a área de trabalho e realize suas ligações.

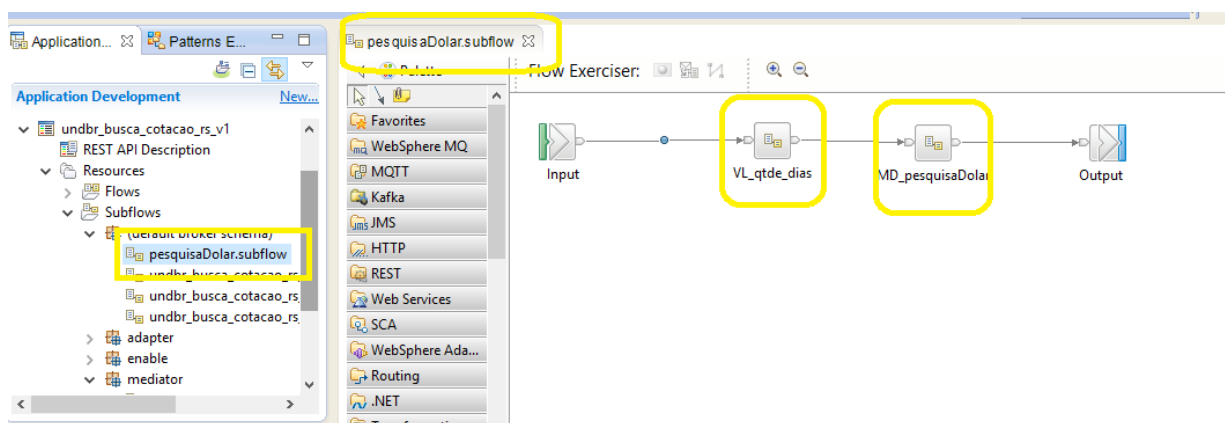
- ***AD\_Consumer\_to\_Provider***
- ***EM\_pesquisaDolar***
- ***AD\_Provider\_to\_Consumer***



## 2.7 Implementando o subflow principal

Para finalizar, disponibilize a camada mediador no fluxo principal:

- 2.7.1. Na esquerda selecione o subflow **pesquisaDolar.subflow**.
- 2.7.2. Duplo clique no mesmo para abrir a área de trabalho
- 2.7.3. Selecione o arraste o subflows **MD\_pesquisaDolar** da camada mediador.
- 2.7.4. Na conclusão teremos a seguinte imagem.



## 2.8 Conclusão

Resumindo, concluímos o projeto com a separação de camadas mediante o contexto e responsabilidade de cada funcionalidade.

