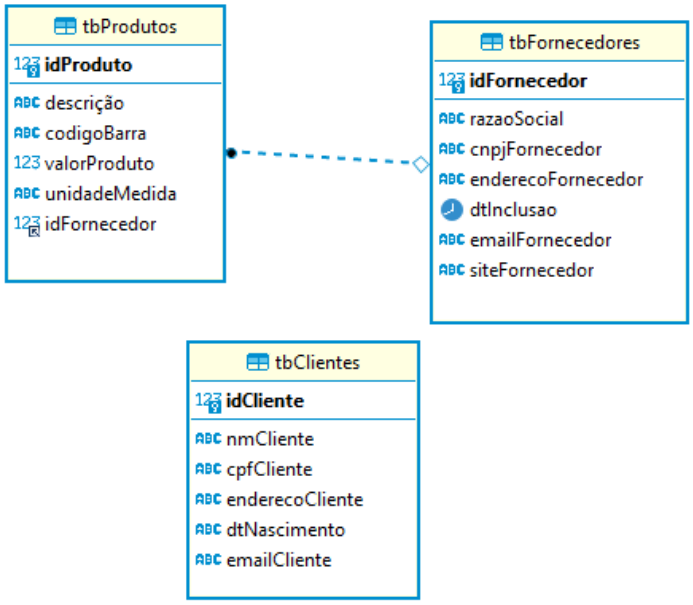
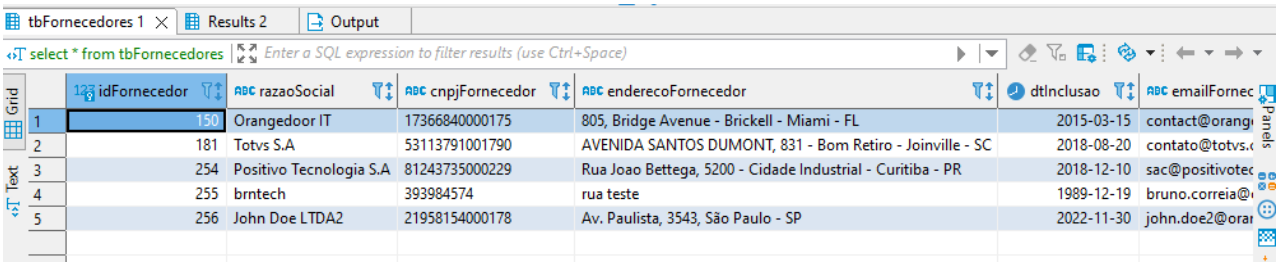
Aula 04

Na aula de hoje iremos continuar o desenvolvimento da API “empresa-s-api-bs”.

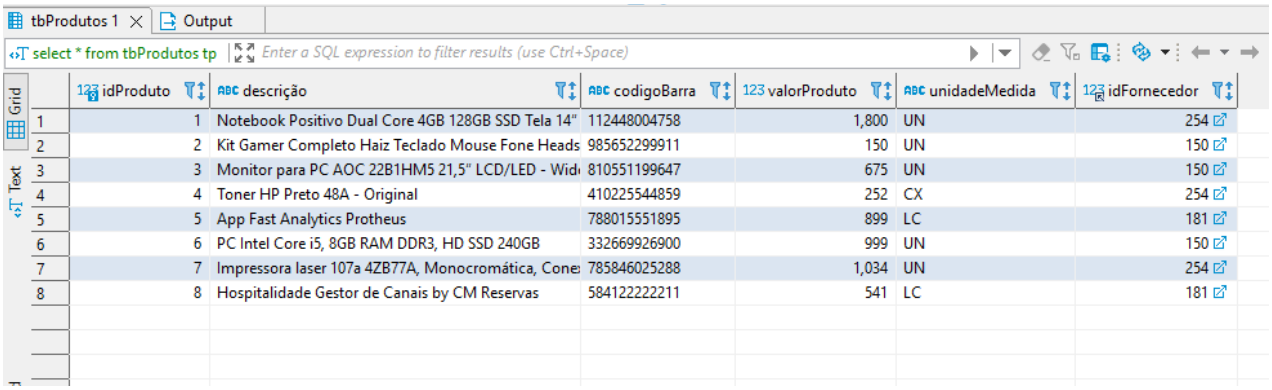
Segue o diagrama com as tabelas e campos que estamos utilizando para o desenvolvimento de nosso projeto



Dados da Tabela tbFornecedores:



Dados da Tabela tbProdutos:

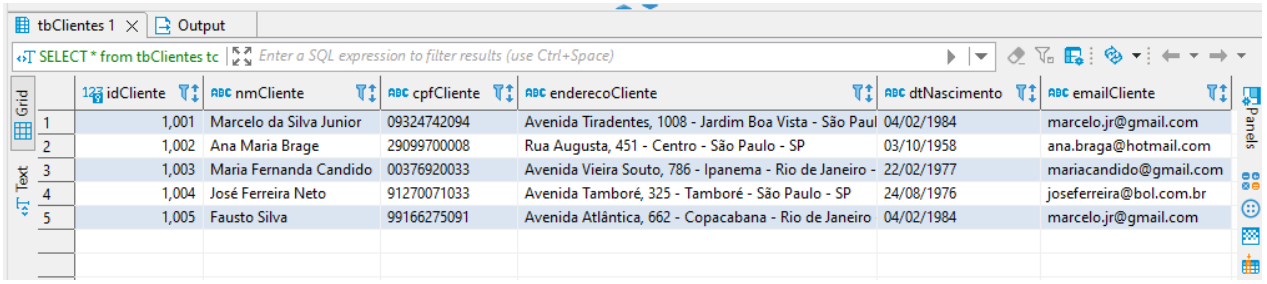


tbProdutos 1 | Output

select * from tbProdutos tp — Enter a SQL expression to filter results (use Ctrl+Space)

	idProduto	descrição	codigoBarra	valorProduto	unidadeMedida	idFornecedor
1	1	Notebook Positivo Dual Core 4GB 128GB SSD Tela 14"	112448004758	1,800	UN	254
2	2	Kit Gamer Completo Haiz Teclado Mouse Fone Heads	985652299911	150	UN	150
3	3	Monitor para PC AOC 22B1HM5 21,5" LCD/LED - Wid	810551199647	675	UN	150
4	4	Toner HP Preto 48A - Original	410225544859	252	CX	254
5	5	App Fast Analytics Protheus	788015551895	899	LC	181
6	6	PC Intel Core i5, 8GB RAM DDR3, HD SSD 240GB	332669926900	999	UN	150
7	7	Impressora laser 107a 4ZB77A, Monocromática, Cone	785846025288	1,034	UN	254
8	8	Hospitalidade Gestor de Canais by CM Reservas	584122222211	541	LC	181

Dados da Tabela tbClientes:



tbClientes 1 | Output

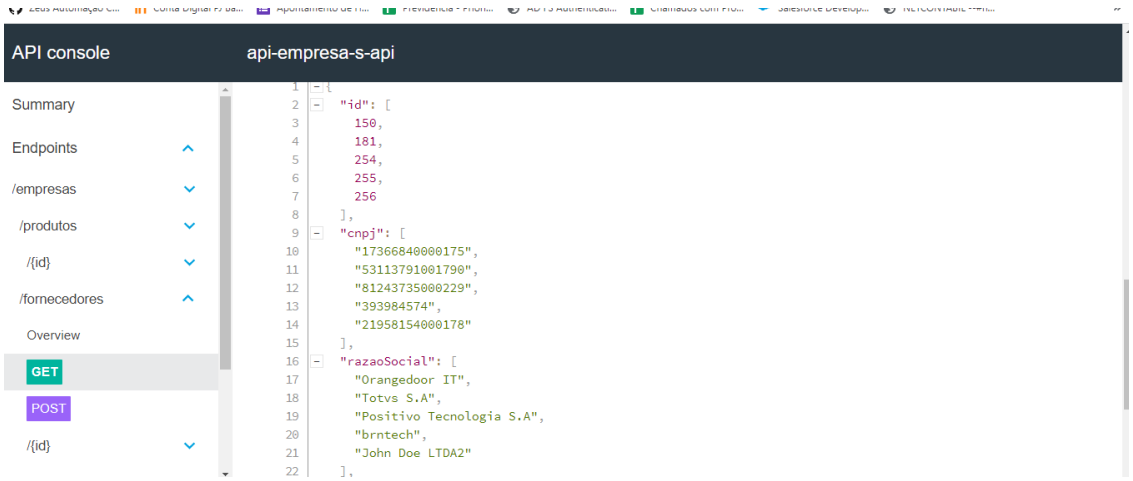
SELECT * from tbClientes tc — Enter a SQL expression to filter results (use Ctrl+Space)

	idCliente	nmCliente	cpfCliente	enderecoCliente	dtNascimento	emailCliente
1	1,001	Marcelo da Silva Junior	09324742094	Avenida Tiradentes, 1008 - Jardim Boa Vista - São Paul	04/02/1984	marcelo.jr@gmail.com
2	1,002	Ana Maria Brage	29099700008	Rua Augusta, 451 - Centro - São Paulo - SP	03/10/1958	ana.braga@hotmail.com
3	1,003	Maria Fernanda Candido	00376920033	Avenida Vieira Souto, 786 - Ipanema - Rio de Janeiro -	22/02/1977	mariacandido@gmail.com
4	1,004	José Ferreira Neto	91270071033	Avenida Tamboré, 325 - Tamboré - São Paulo - SP	24/08/1976	joseferreira@bol.com.br
5	1,005	Fausto Silva	99166275091	Avenida Atlântica, 662 - Copacabana - Rio de Janeiro	04/02/1984	marcelo.jr@gmail.com

No componente “Transform Message” iremos acrescentar a seguinte transformação do payload recebido com os dados dos fornecedores retornados do BD, porém neste caso não quero que seja retornado o campo “idFornecedor”, pois não iremos utilizar esta informação.

```
%dw 2.0
output application/json
---
{
	fornecedores:
	{
		site: payload.siteFornecedor default "",
		endereco: payload.enderecoFornecedor default "",
		dataInclusao: payload.dtInclusao as String default "",
		cnpj: payload.cnpjFornecedor default "",
		razaoSocial: payload.razaoSocial default "",
		email: payload.emailFornecedor default ""
	}
}
```

Após acrescentar o código, rode a API realize uma chamada. Veja o resultado



Os dados trazidos do BD estão sendo exibidos de maneira desorganizada.

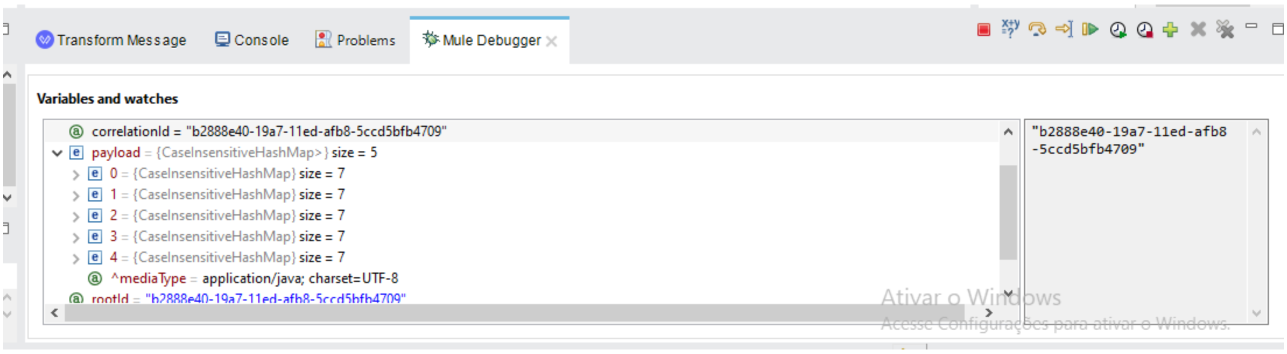
Função MAP

Função Dataweave utilizada para percorrer todos os itens em um Array. Se você vem de um background de desenvolvimento diferente, você pode conhecer esta função como “for” ou “forEach”. Como o DataWeave é uma linguagem de programação funcional, as instruções não estão sendo executadas em uma sequência (como “for” faria).

Podemos realizar testes com o MAP utilizando a ferramenta “Playground” disponibilizada online pela MuleSoft

https://developer.mulesoft.com/learn/dataweave/playground

Notem que no exemplo anterior, o payload recebido com os dados do BD, vieram em uma lista (array). Neste caso teremos que usar a função “MAP” para realizar a leitura desta lista percorrendo todas as linhas.



Esta é a estrutura em que recebemos os dados, em cada linha temos uma lista (colunas) com “chave” e “valor”, onde a chave compreende o nome do campo e o valor os dados retornados.

Razao Social=”Valor”	cnpjFornecedor=”Valor”	IdFornecedor=”Valor”	enderecoFornecedor=”Valor”	dtInclusao=”Valor”	siteFornecedor=”Valor”	emailFornecedor=”Valor”
Razao Social=”Valor”	cnpjFornecedor=”Valor”	idFornecedor=”Valor”	enderecoFornecedor=”Valor”	dtInclusao=”Valor”	siteFornecedor=”Valor”	emailFornecedor=”Valor”
Razao Social=”Valor”	cnpjFornecedor=”Valor”	idFornecedor=”Valor”	enderecoFornecedor=”Valor”	dtInclusao=”Valor”	siteFornecedor=”Valor”	emailFornecedor=”Valor”
Razao Social=”Valor”	cnpjFornecedor=”Valor”	idFornecedor=”Valor”	enderecoFornecedor=”Valor”	dtInclusao=”Valor”	siteFornecedor=”Valor”	emailFornecedor=”Valor”
Razao Social=”Valor”	cnpjFornecedor=”Valor”	idFornecedor=”Valor”	enderecoFornecedor=”Valor”	dtInclusao=”Valor”	siteFornecedor=”Valor”	emailFornecedor=”Valor”

Para a transformação correta do nosso componente “Transform Message”, iremos digitar o código abaixo utilizando a função MAP

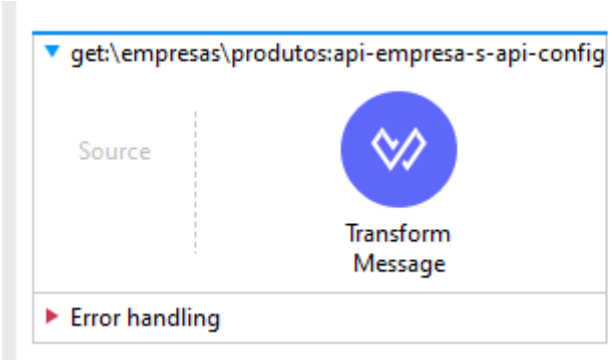
%dw 2.0

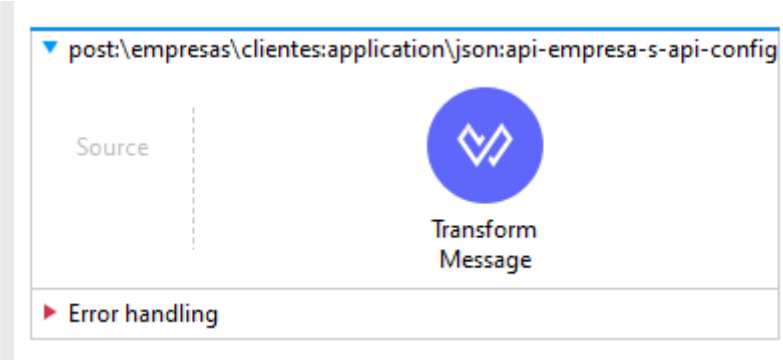
```
output application/json
---
{
fornecedores: payload map (valor , indice ) -> {
site: valor.siteFornecedor default "",
endereco: valor.enderecoFornecedor default "",
dataInclusao: valor.dtInclusao as String default "",
cnpj: valor.cnpjFornecedor default "",
razaoSocial: valor.razaoSocial default "",
email: valor.emailFornecedor default ""
}
}
```

Coloque a aplicação para rodar e execute o teste novamente para verificar o resultado.

Realize um teste também da chamada da API utilizando o Postam.

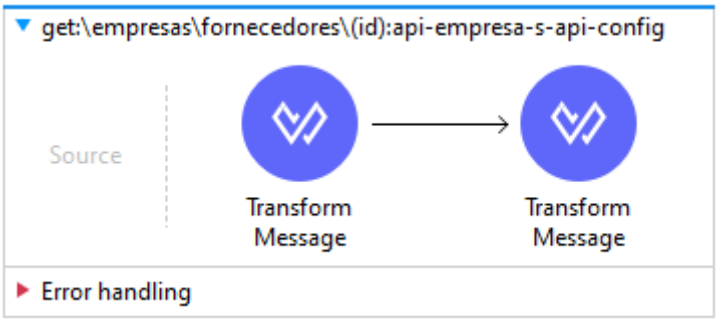
Após tudo certo, vamos replicar o desenvolvimento para os fluxos “get empresas/clientes” e “get empresas/produtos”





Após terminar de rodar a aplicação e realizar os testes.

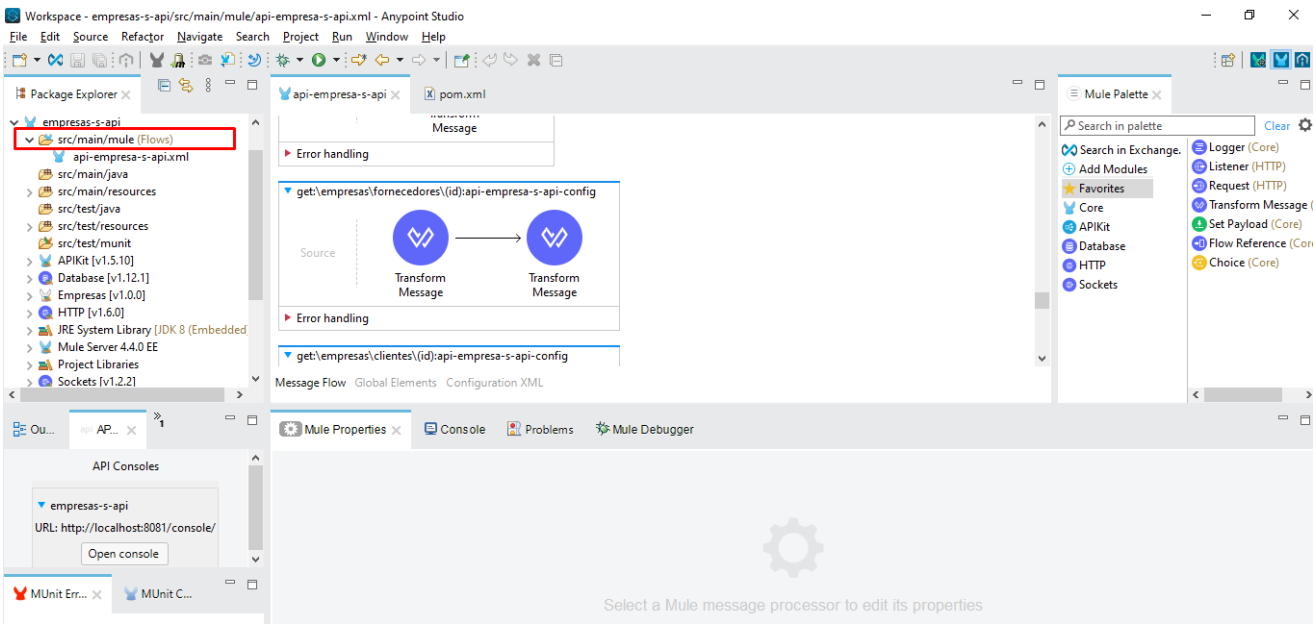
Agora iremos realizar o desenvolvimento para o fluxo de “get empresas/fornecedores/{id}”



Neste fluxo será necessário a passagem da URI Parameter “id” para que seja realizada uma busca no banco de dados pelo valor do “IdFornecedor”.

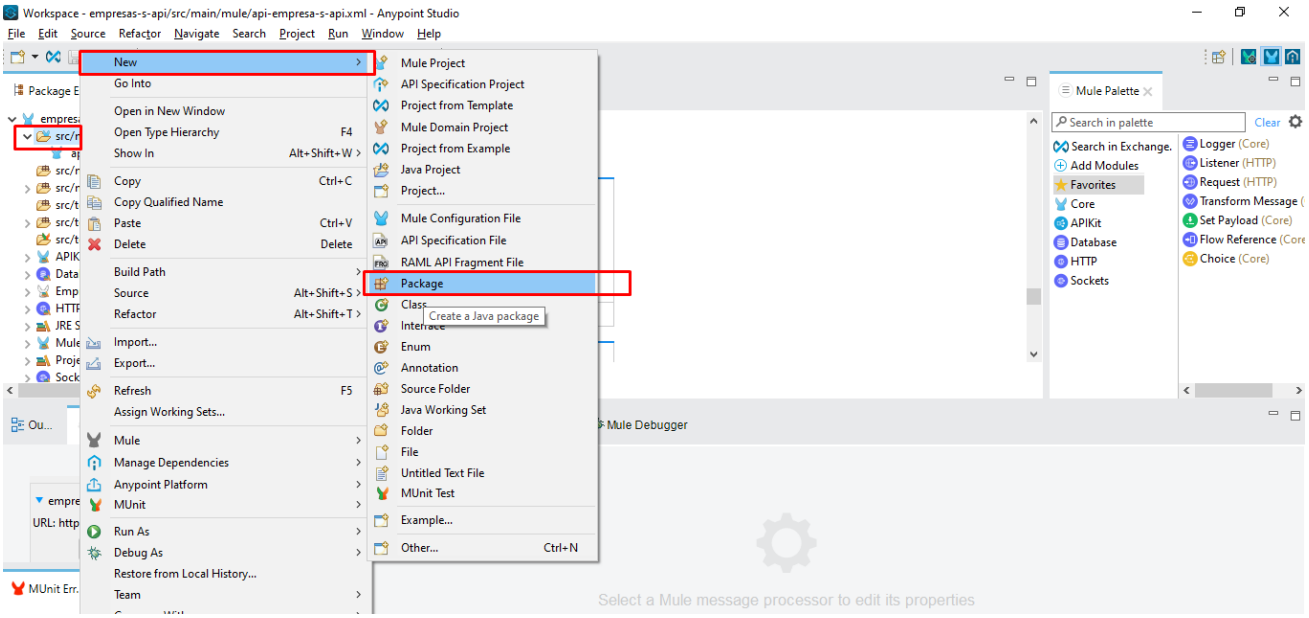
A partir de agora iremos criar uma estrutura mais organizada de nossa API, utilizando o conceito de “Subflows” (subfluxos).

Iremos criar uma nova package (pacote) chamada fornecedores, dentro para pasta do Flow principal

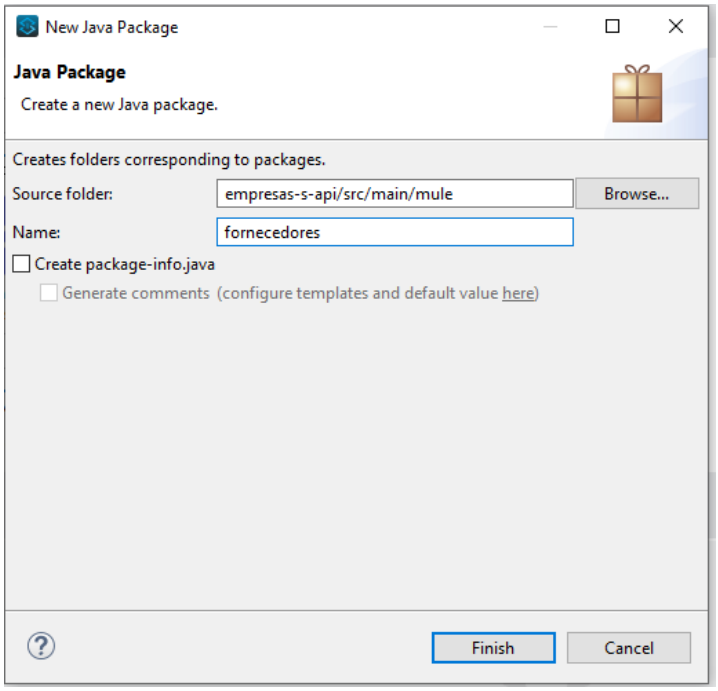


Clique com o botão direito do mouse em cima da pasta “src/main/mule/”,

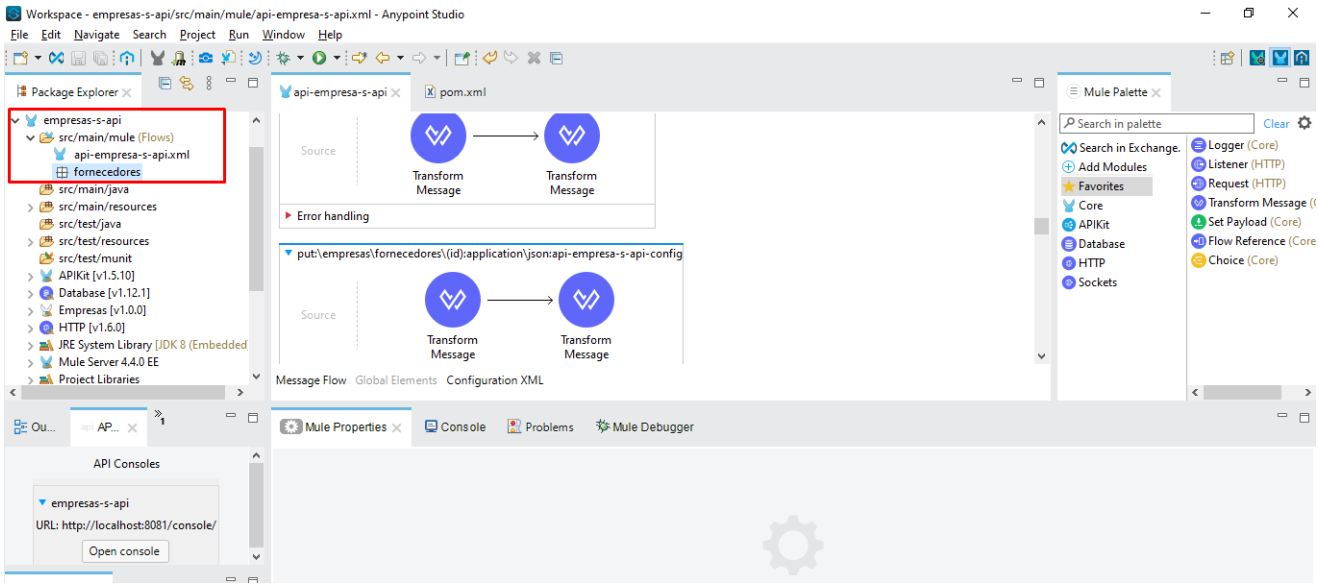
New / Package



Digite o nome “fornecedores” para o package e verifique se está sendo criado na pasta correta conforme imagem abaixo

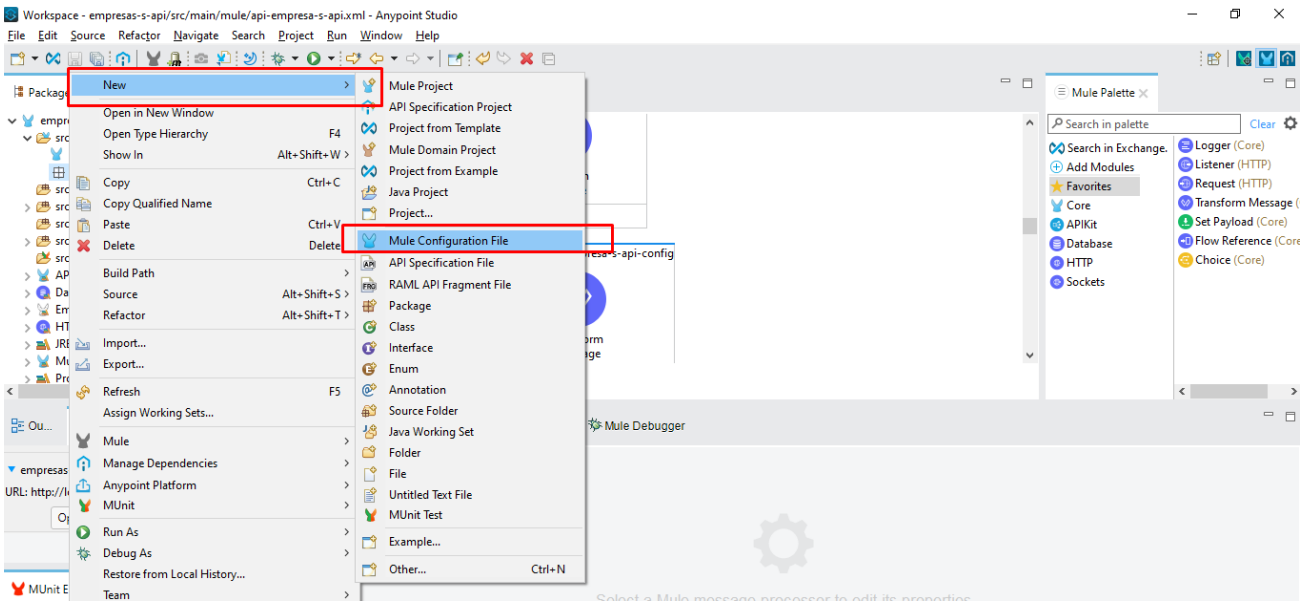


Ficará nesta estrutura

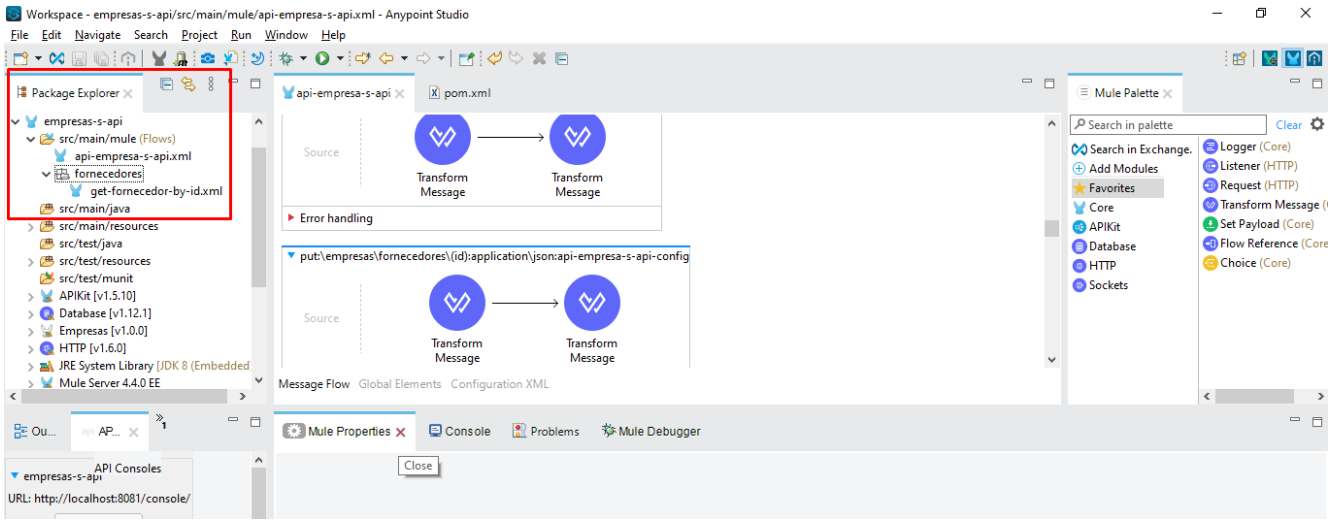


Dentro da package “fornecedores” precisaremos criar um novo arquivo mule xml chamado “get-fornecedor-by-id”

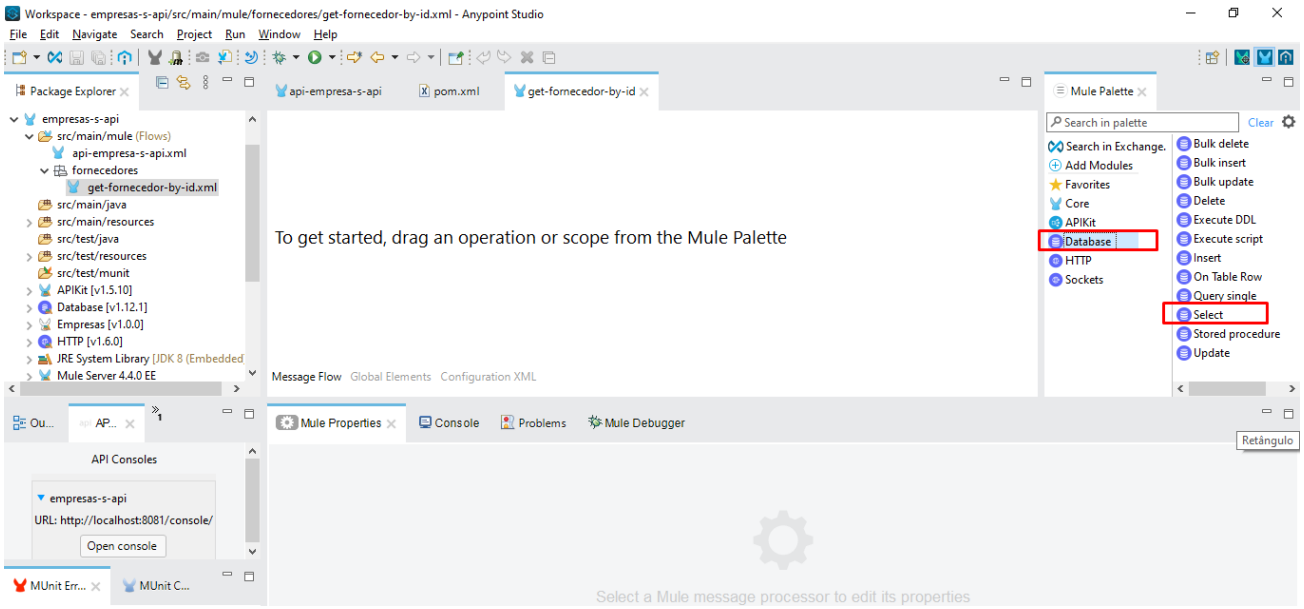
Clique com o botão direito do mouse em cima da package “fornecedores”, New / Mule Configuration File e digite o nome do arquivo “get-fornecedor-by-id”



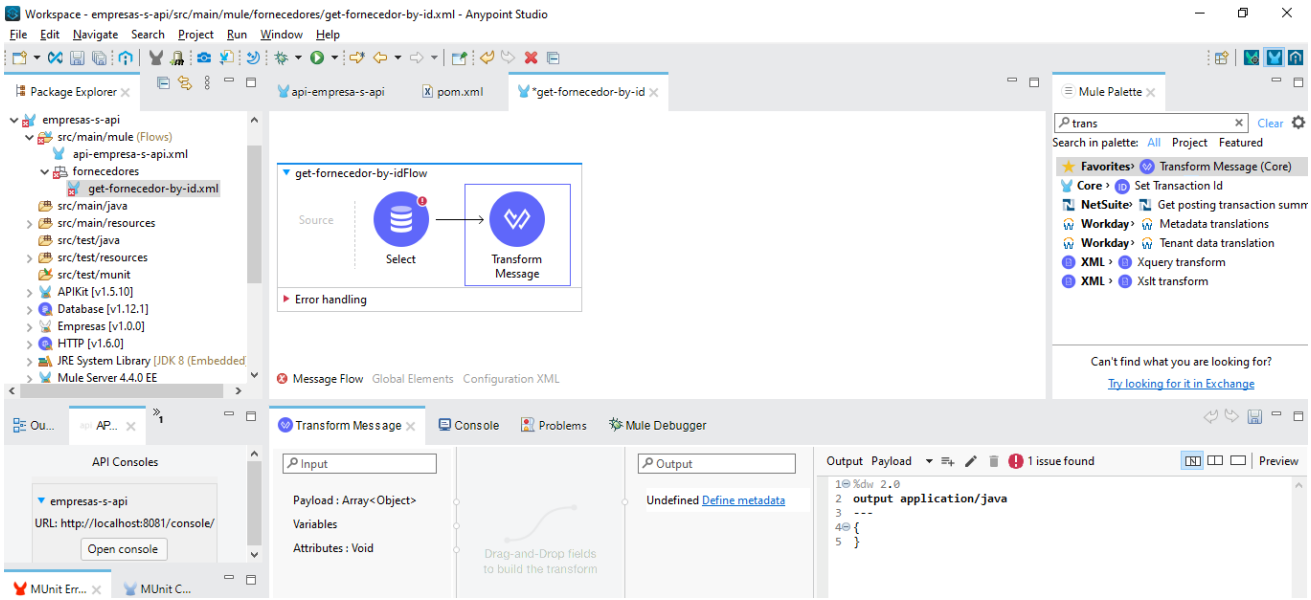
Deverá ficar conforme estrutura abaixo. Caso o arquivo criado tenha ficado fora da package, você arrastá-lo para dentro da package.



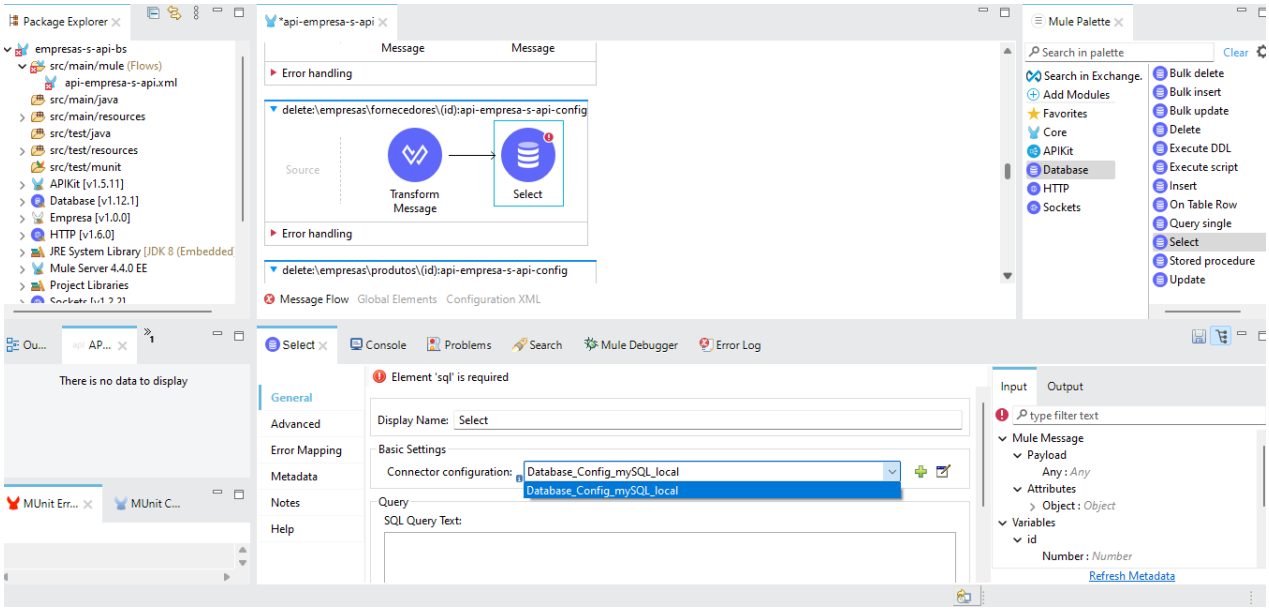
No arquivo iremos criar o nosso Subflow “get-fornecedor-by-id”, onde incluiremos o componente “Database Select” para realizar a consulta no BD. Basta clicar no componente e arrastá-lo para a área em branco do *canvas*. Vamos incluir também um componente “Transform Message” para transformar os dados de nossa consulta.



Devendo ficar desta maneira



No componente Database Select iremos incluir a mesma configuração do fluxo anterior, para a conexão com o BD MyQSL.

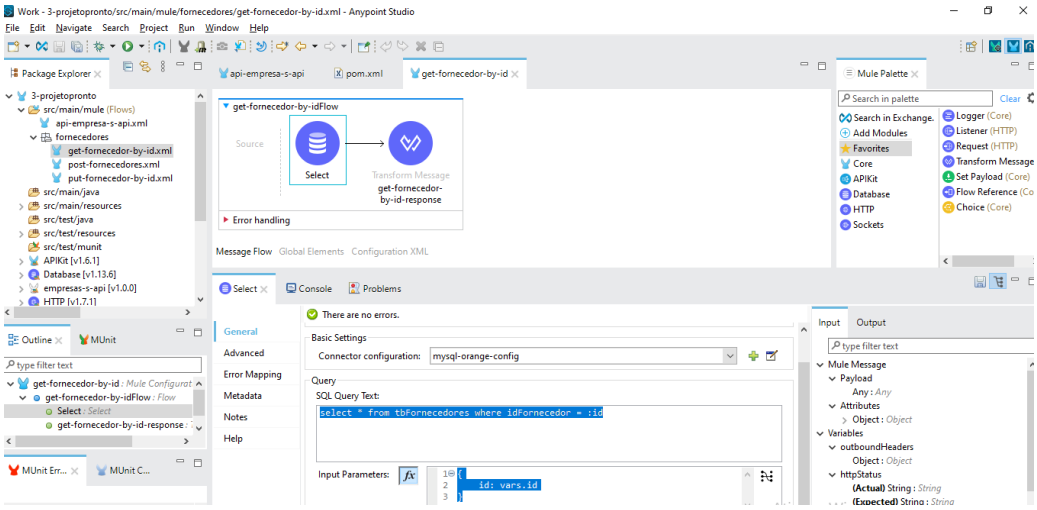


No campo SQL Query Text, digitar o select abaixo:

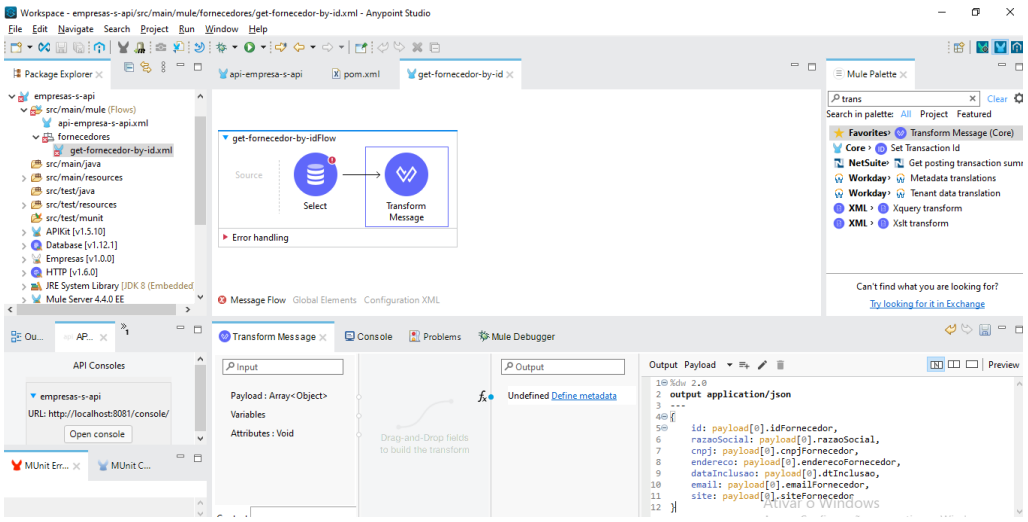
*select * from tbFornecedores where idFornecedor = :id*

E no campo “Input Parameters” clique no “fx” e digite o código abaixo:

```
{
id: vars.id
}
```



No componente “Transform Message” insira o código abaixo para realizar a transformação dos dados retornados do banco de dados



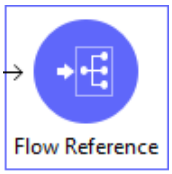

```
%dw 2.0
output application/json
---
{
id: payload[0].idFornecedor,
razaoSocial: payload[0].razaoSocial,
cnpj: payload[0].cnpjFornecedor,
endereco: payload[0].enderecoFornecedor,
dataInclusao: payload[0].dtInclusao,
email: payload[0].emailFornecedor,
```

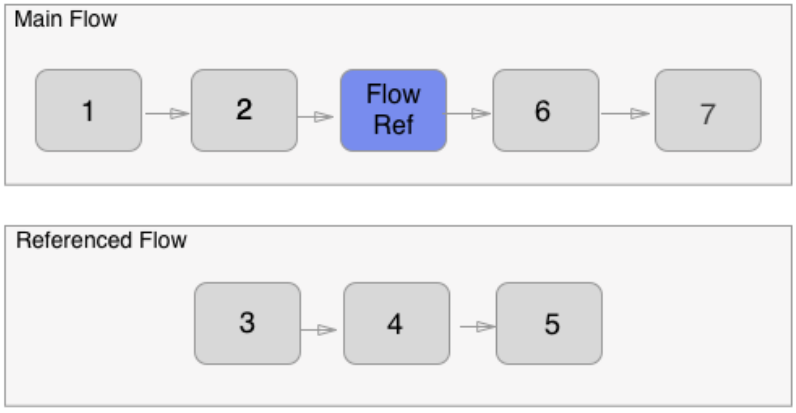
site: payload[0].siteFornecedor

}

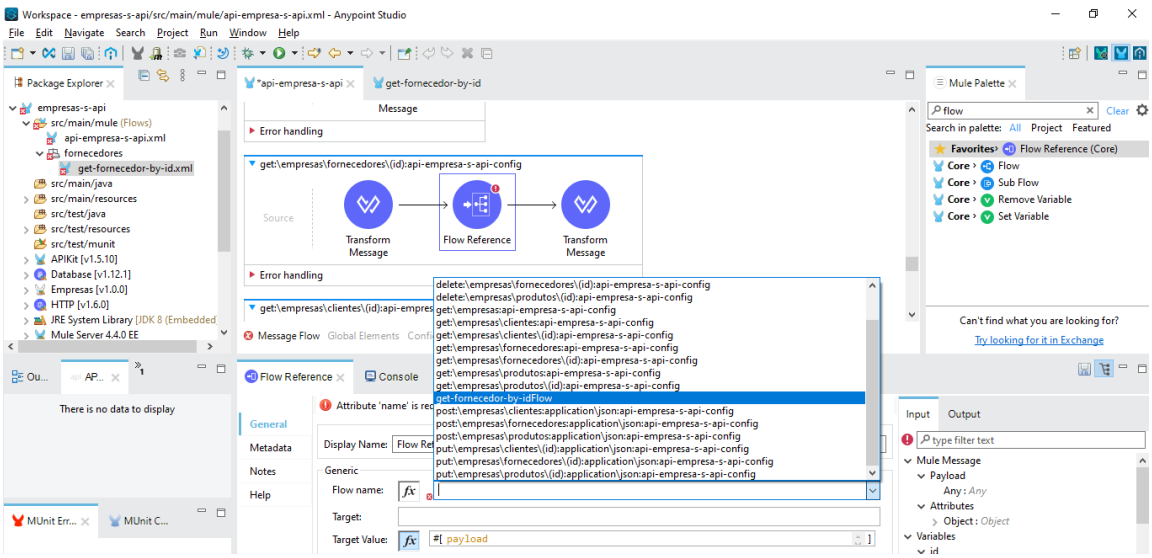
Voltando ao nosso fluxo principal, iremos utilizar um novo componente chamado “Flow Reference”

Flow Reference: Componente responsável por fazer o roteamento do evento para outro flow ou subflow, onde posteriormente este evento volta para continuar de onde foi roteado.

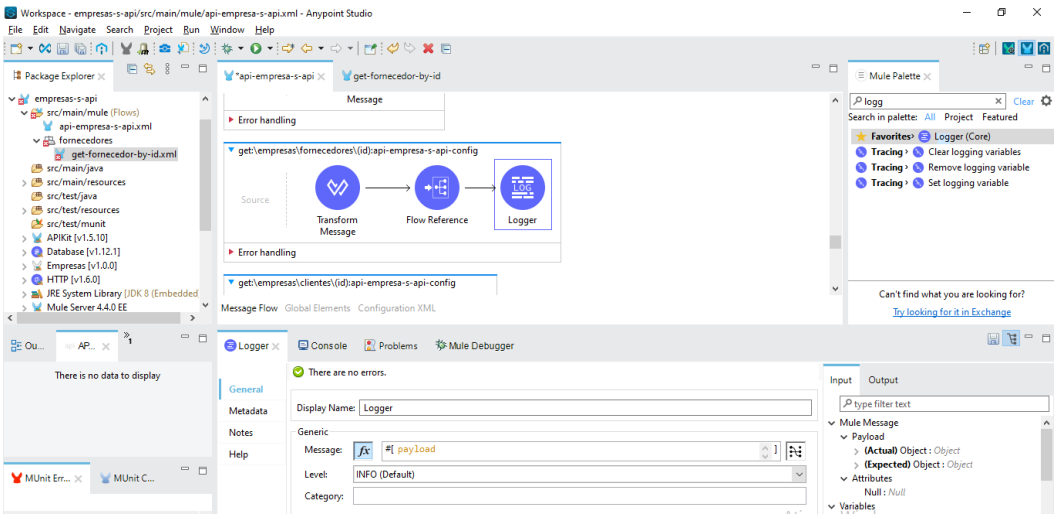




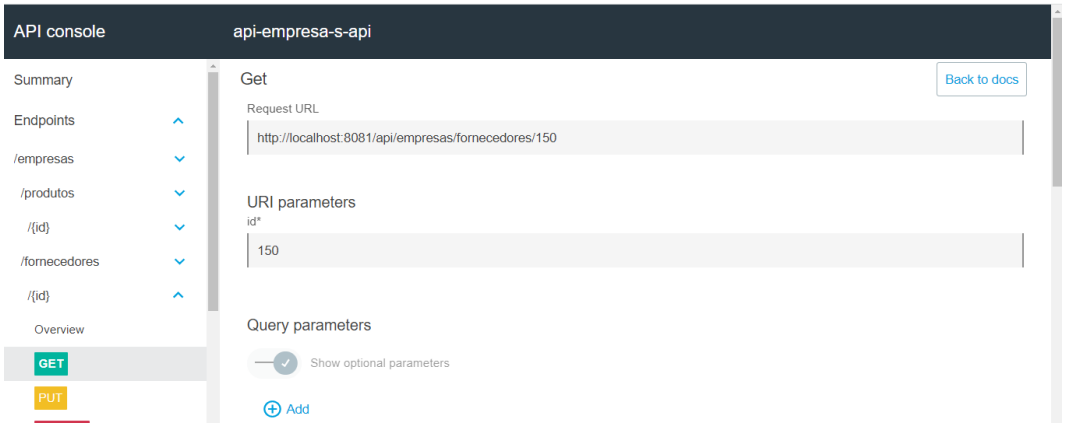
Incluir no Fluxo principal o componente “Flow Reference” e alterar a propriedade Flow Name selecionando o subflow criado “get-fornecedor-by-idFlow "



Apague o último componente “Transform Message” pois não vamos utilizá-lo, inclua um componente “Logger” e coloque na propriedade message “payload” para que seja logado as informações do payload retornado



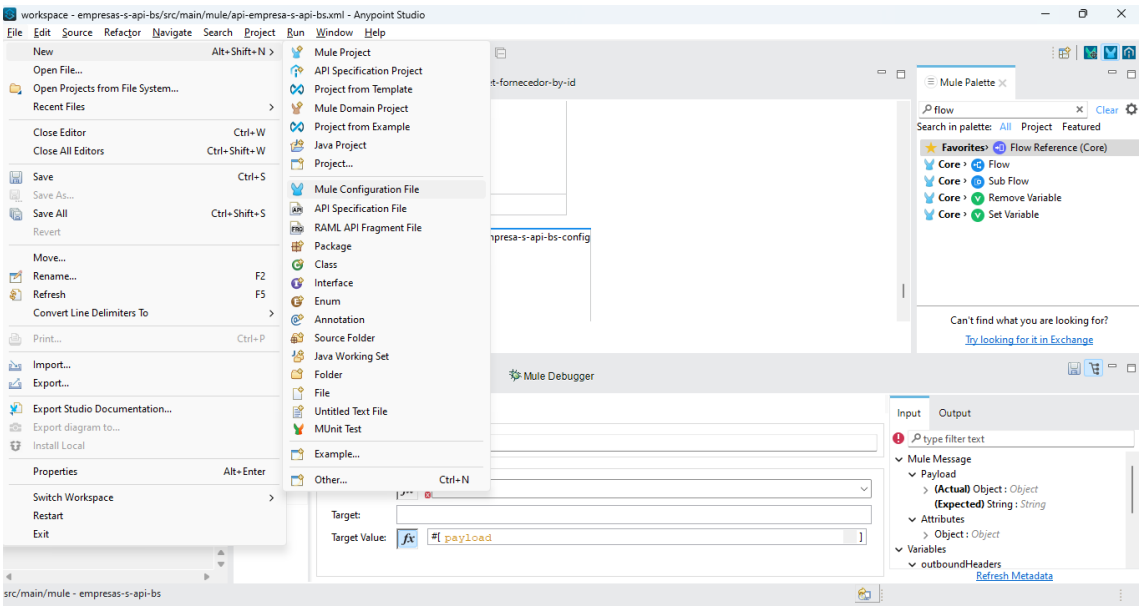
Salve a sua aplicação e realize os testes.



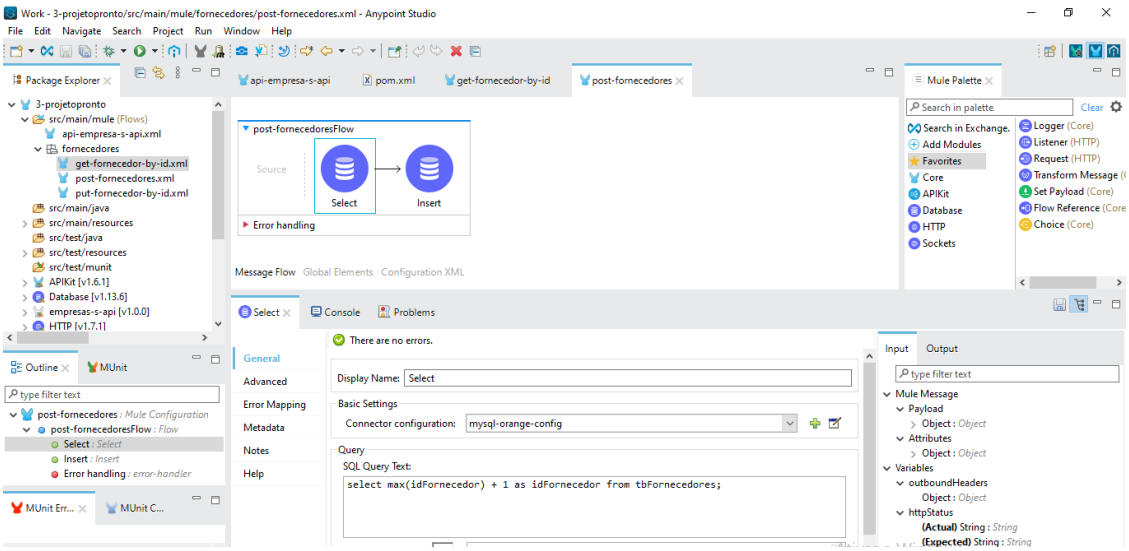
Caso ocorra tudo certo, aplique o mesmo desenvolvimento para os outros fluxos “get empresas/clientes/{id}” e “get empresas/produtos/{id}”

Vamos agora para o desenvolvimento do nosso método POST Fornecedores (inclusão de fornecedores).

Crie o arquivo “post-fornecedores” clicando no menu File/New/Mule Configuration File dentro da package “fornecedores”



No subFlow “post-fornecedores” iremos incluir os componentes “Database Select” e o componente “Database Insert”



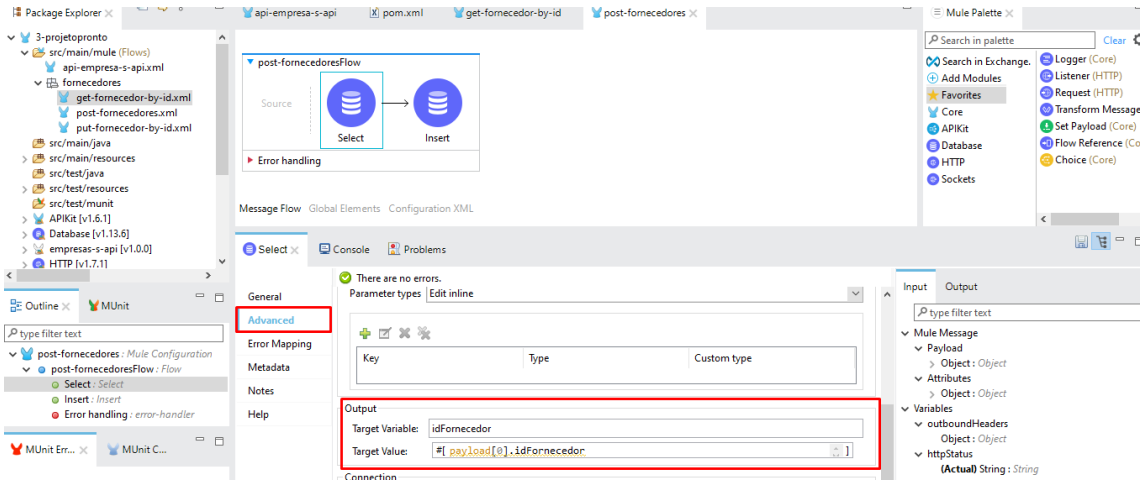
No componente “Database Select” realize as configurações de conexão conforme exemplos anteriores e no campo SQL Query Text inclua o seguinte select:

select max(idFornecedor) + 1 as idFornecedor from tbFornecedores;

Na aba “Advanced” do componente “Database Select” inclua os valores nos campos conforme imagem.

Target variable: *idFornecedor*

Target Value (clique fx): *payload[0].idFornecedor*



No componente “Database Insert” realize as configurações de conexão conforme realizado no componente “Database Select” e nos campos abaixo inclua os códigos:

SQL Query Text:

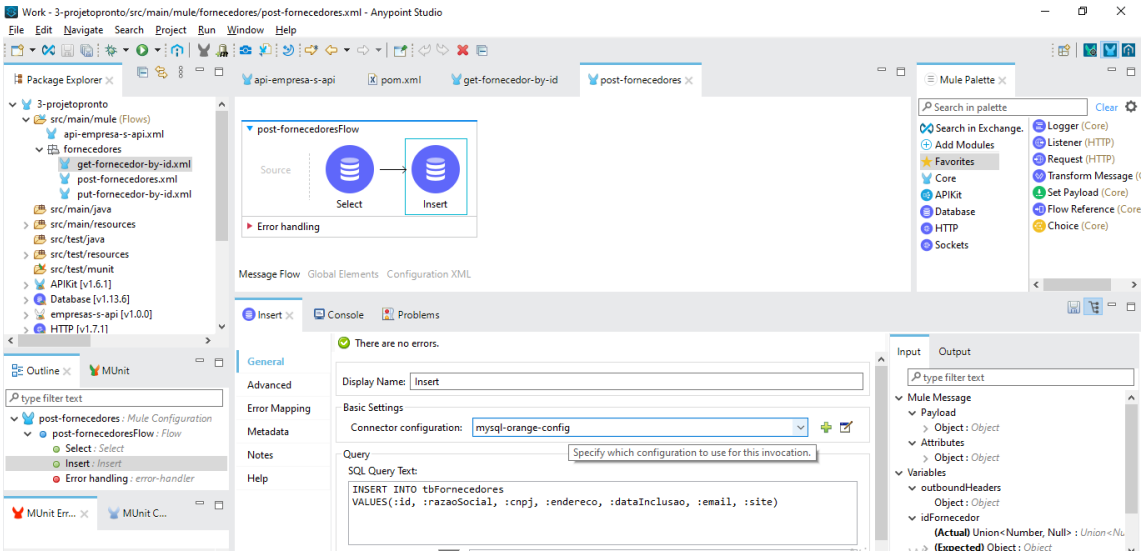
```
INSERT INTO tbFornecedores
VALUES(:id, :razaoSocial, :cnpj, :endereco, :dataInclusao, :email, :site)
```

No “Input Parameters:” Clique no botão “fx”

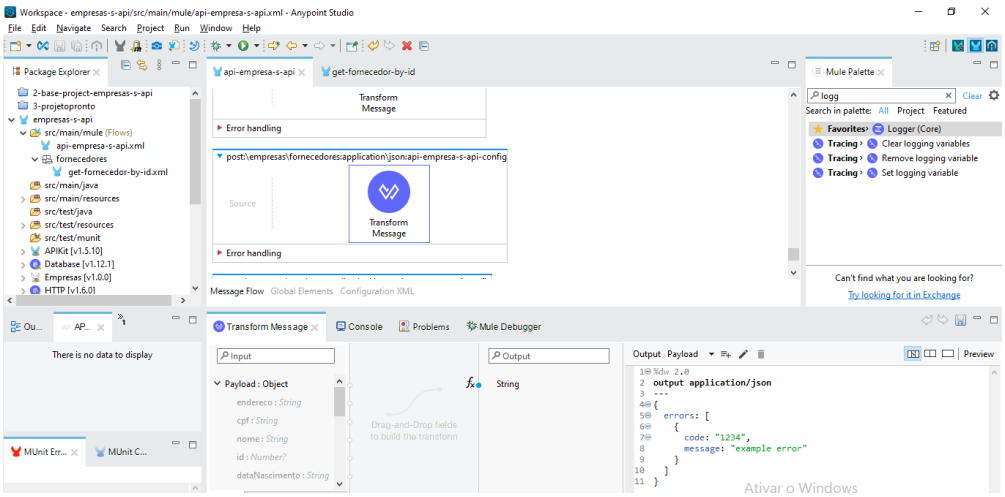
```
%dw 2.0
output application/json
---
{
id: vars.idFornecedor,
razaoSocial: payload.razaoSocial,
cnpj: payload.cnpj,
endereco: payload.endereco,
dataInclusao: payload.dataInclusao,
email: payload.email,
```

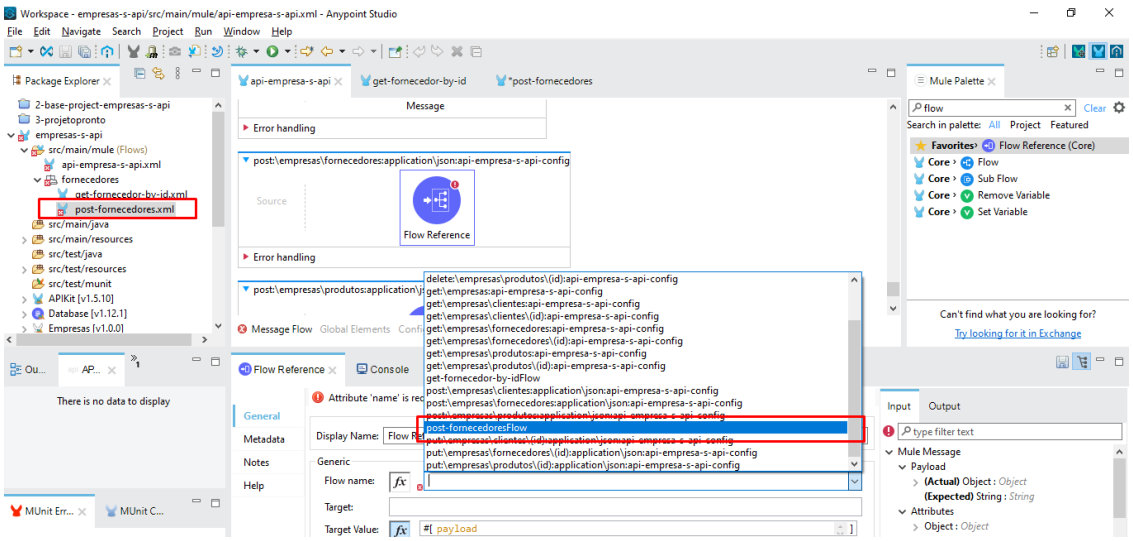
site: payload.site

}



No fluxo “post:empresas\fornecedores” apague o componente “Transform Message” e inclua um componente “Flow Reference” apontando para o subFlow criado “post-fornecedores”

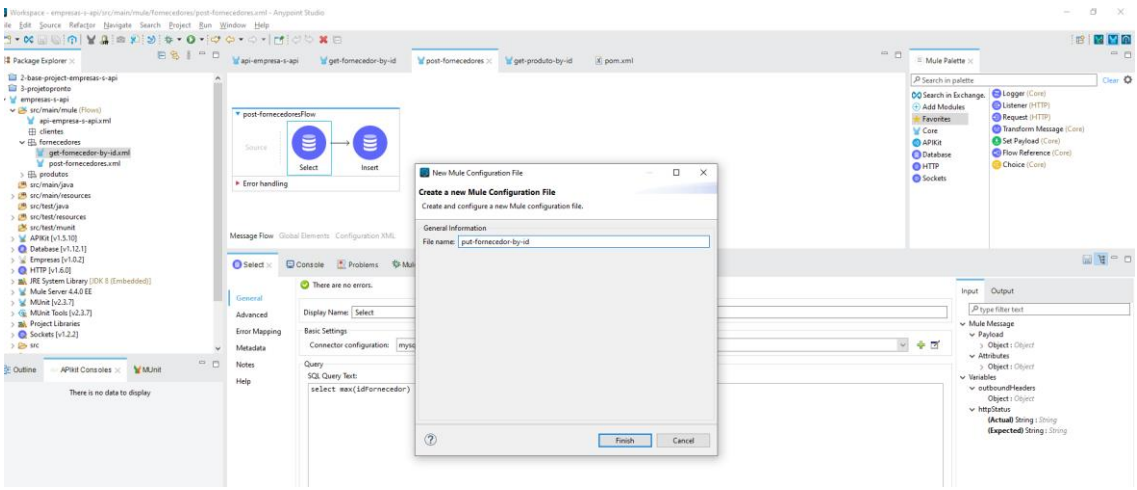




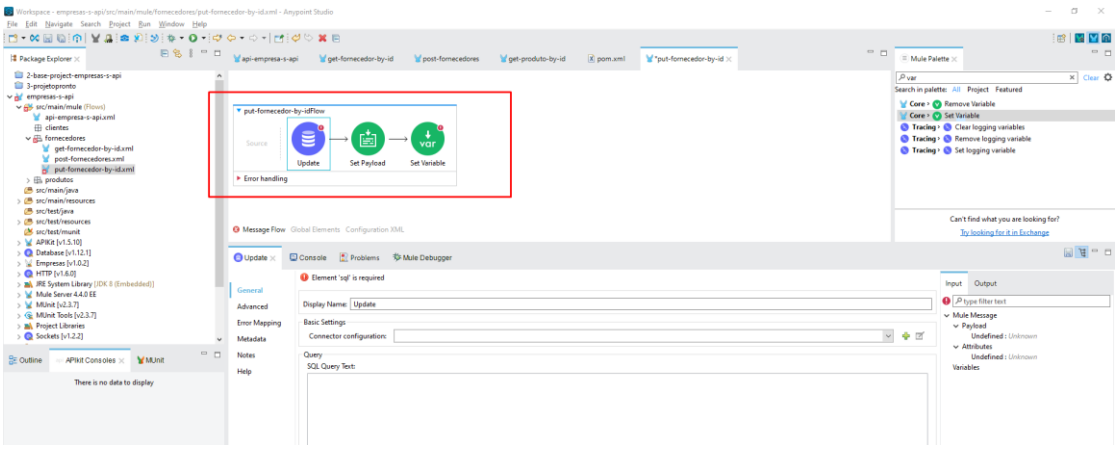
Coloque a aplicação para rodar e realize os testes.

Neste momento vamos trabalhar com o nosso fluxo de Alterações de dados no BD

Iremos criar um novo subFlow com o nome “put-fornecedores-by-id” dentro da package fornecedores



No novo subFlow iremos utilizar os componentes abaixo no fluxo de alteração (PUT)

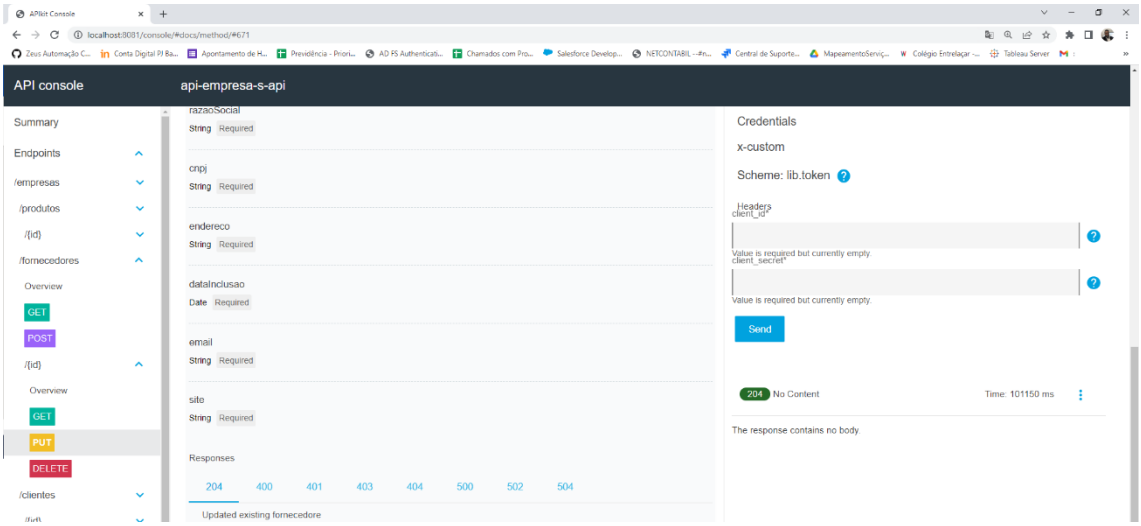


Componente Database Update: Componente responsável por realizar alterações de dados no BD.

Componente Set Payload: Componente que permite atualizar payload. O payload pode ser uma string literal ou uma expressão DataWeave. No entanto, não é recomendado para expressões ou transformações complexas.

Componente Set Variable: Componente que permite criar ou atualizar uma variável para armazenar valores para uso no fluxo de uma aplicação Mule

Vamos utilizar os componentes “Set Payload e Set Variables para dar uma melhorada no Response de nossa chamada, pois senão estaria vindo assim:



No componente “Database Update” iremos realizar as alterações nas propriedades abaixo

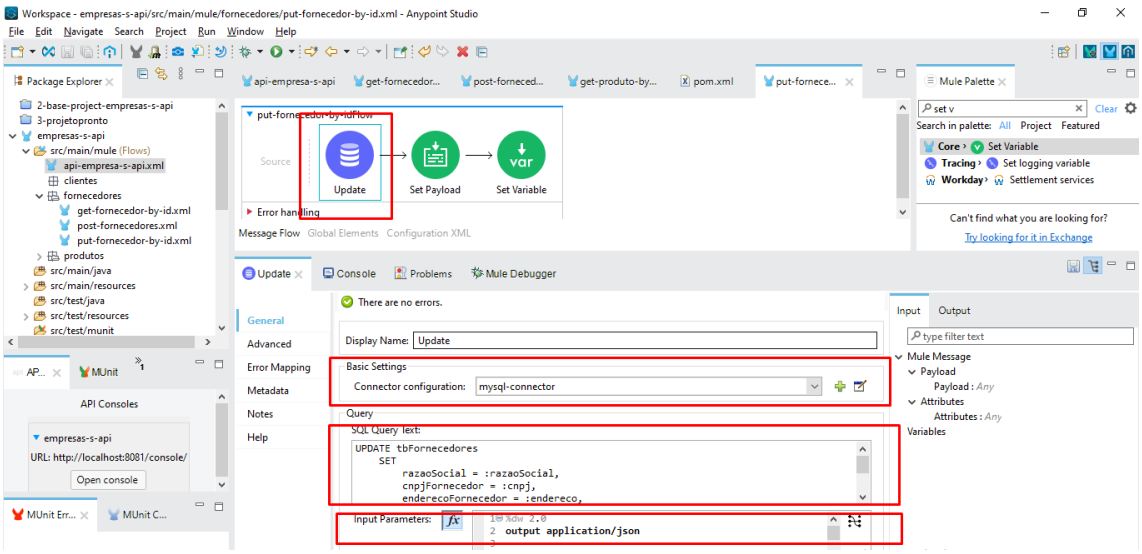
- Connector configuration: Selecione a conexão com o MySQL já criada anteriormente “mysql-connector” (salva assim no meu projeto)

- SQL Query Text:

```
UPDATE tbFornecedores
SET
        razaoSocial = :razaoSocial,
        cnpjFornecedor = :cnpj,
        enderecoFornecedor = :endereco,
        dtInclusao = :dataInclusao,
        emailFornecedor = :email,
        siteFornecedor = :site
WHERE
idFornecedor = :id
```

- Input Parameters:

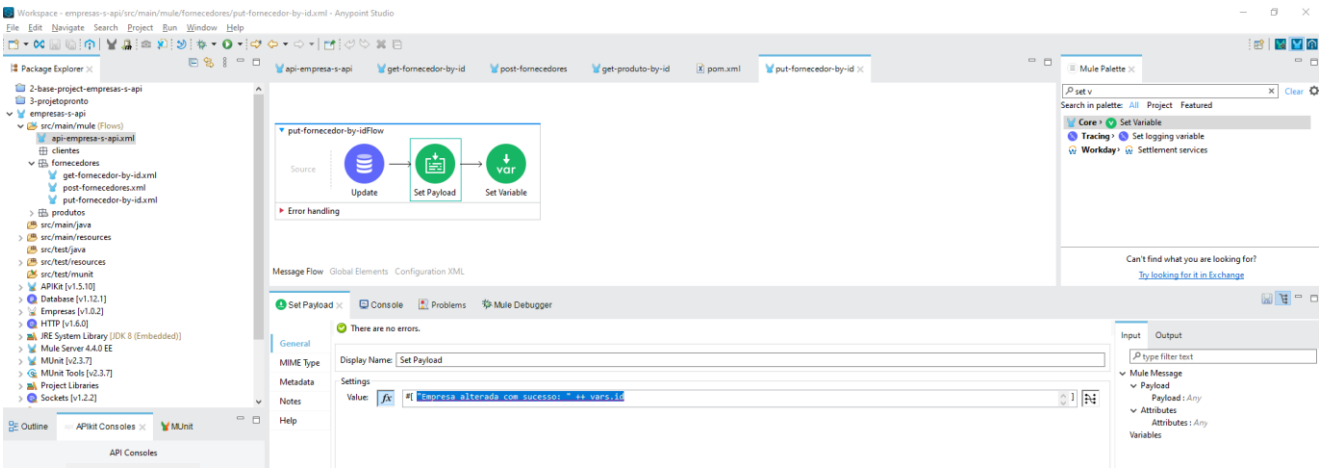
```
%dw 2.0
output application/json

---
{
id: vars.id,
razaoSocial: payload.razaoSocial,
cnpj: payload.cnpj,
endereco: payload.endereco,
dataInclusao: payload.dataInclusao,
email: payload.email,
site: payload.site
}
```



No componente ”Set Payload” iremos alterar a seguinte propriedade:

- *Value:*

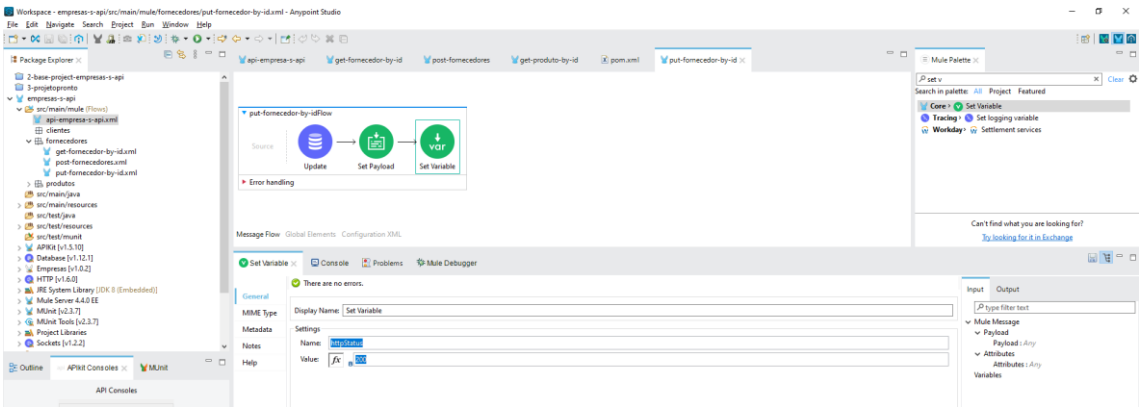
"Empresa alterada com sucesso: " ++ vars.id



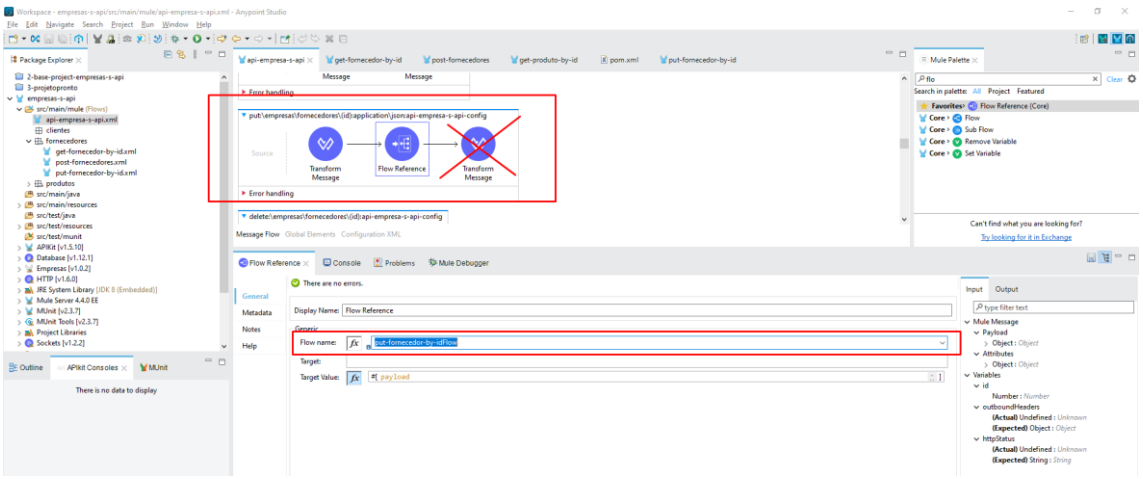
No componente “Set Variable” iremos alterar as seguintes propriedades

- *Name: httpStatus*

- *Value: 200*

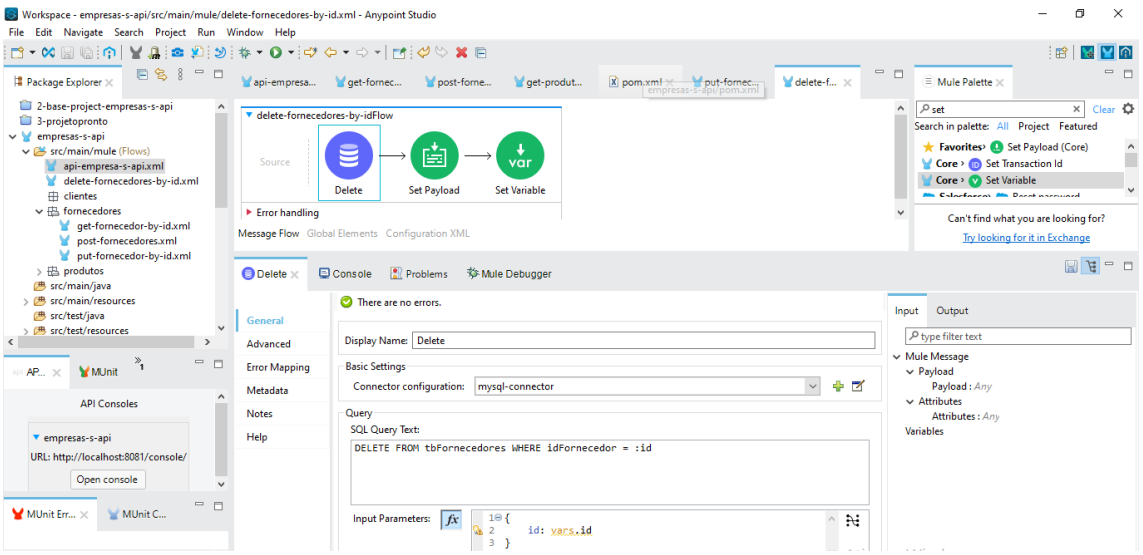


No Flow principal iremos incluir o componente “Flow Reference” e alterar a propriedade Flow Name apontando para o novo subFlow criado “put-fornecedor-by-idFlow”. Vamos também excluir o segundo componente “Transform Message”, pois não vamos utilizá-lo.

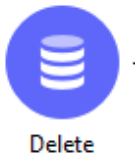


Agora vamos rodar e testar nossa API

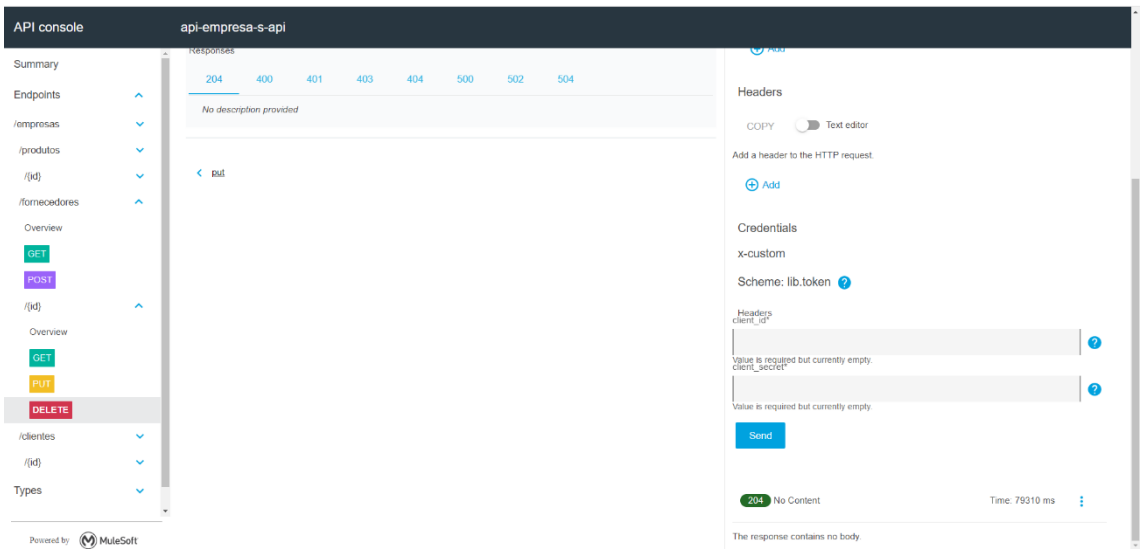
No novo subFlow iremos utilizar os componentes abaixo no fluxo de delete (DELETE)



Componente Database Delete: Componente responsável por realizar a exclusão de dados no BD.



Notem que neste caso também iremos utilizar os componentes “Set Payload” e “Set Variables” para melhorar a visualização do Response, pois senão viria desta maneira.



No componente “Database Delete” iremos alterar as seguintes propriedades

- Connector configuration: Selecione a conexão com o MySQL já criada anteriormente “mysql-connector” (salva assim no meu projeto)

- SQL Query Text:

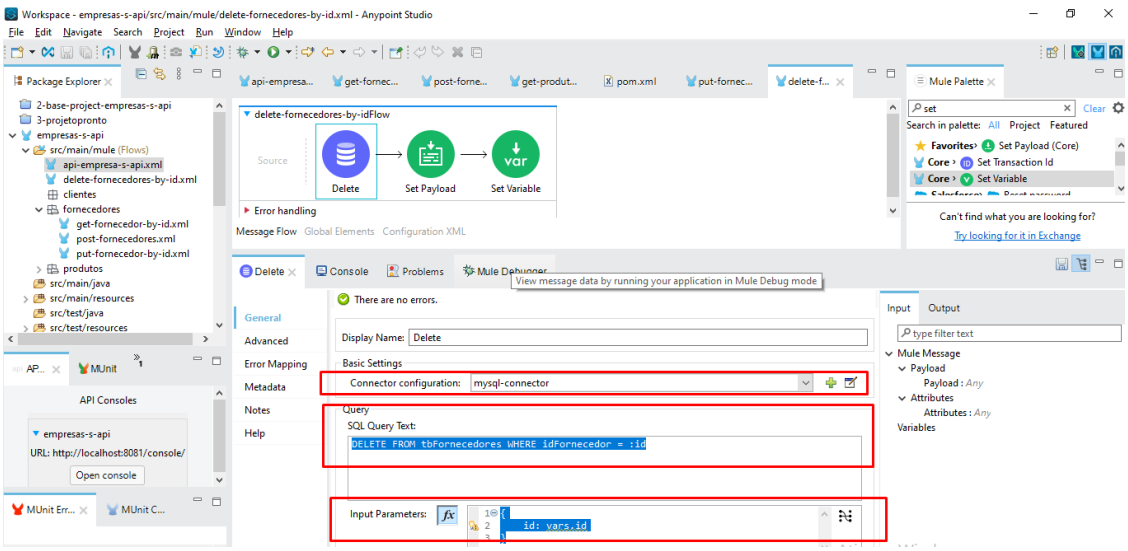
DELETE FROM tbFornecedores WHERE idFornecedor = :id

- Input Parameters:

{

id: vars.id

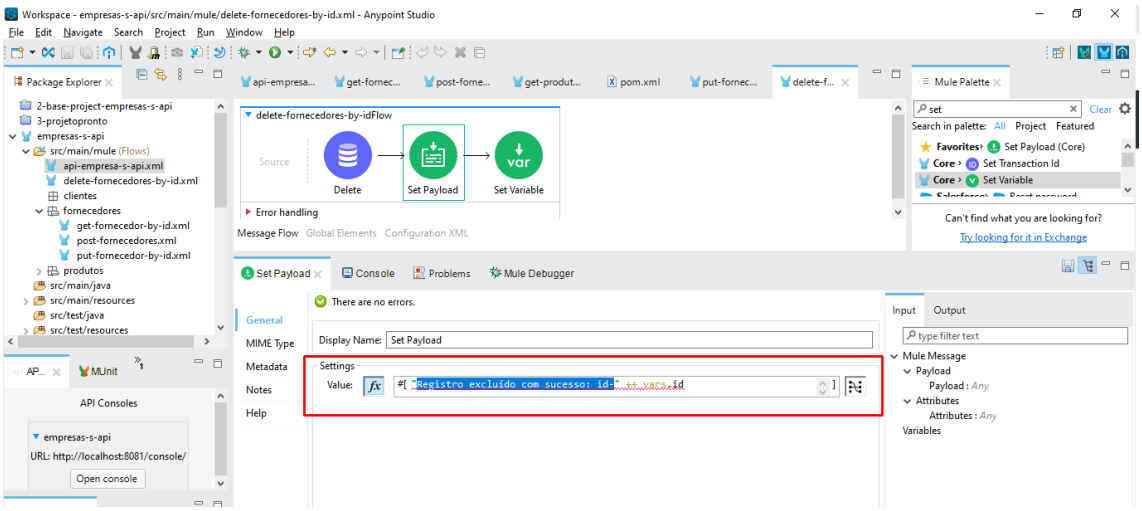
}



No componente ”Set Payload” iremos alterar a seguinte propriedade:

- Value:

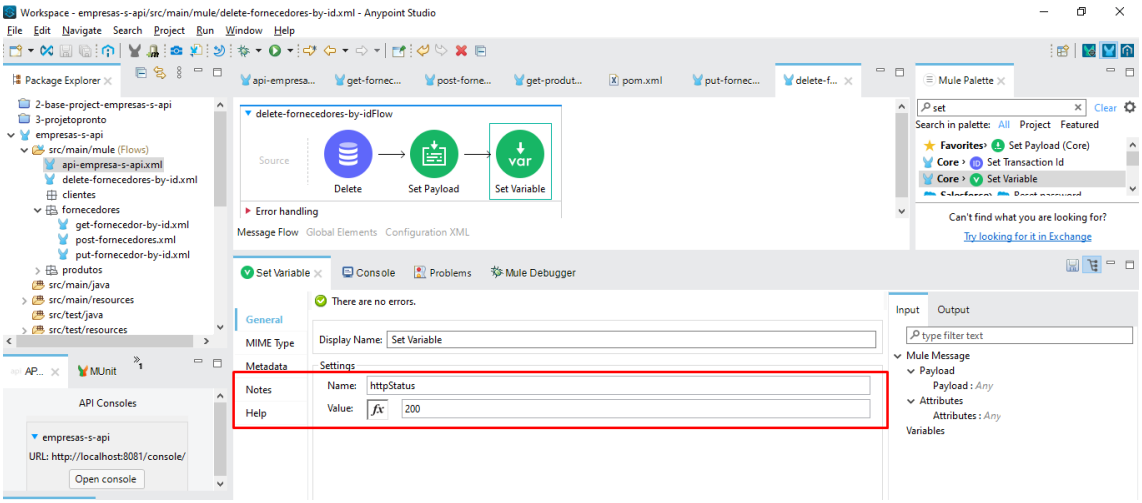
” Registro excluído com sucesso: id-” ++ vars.id



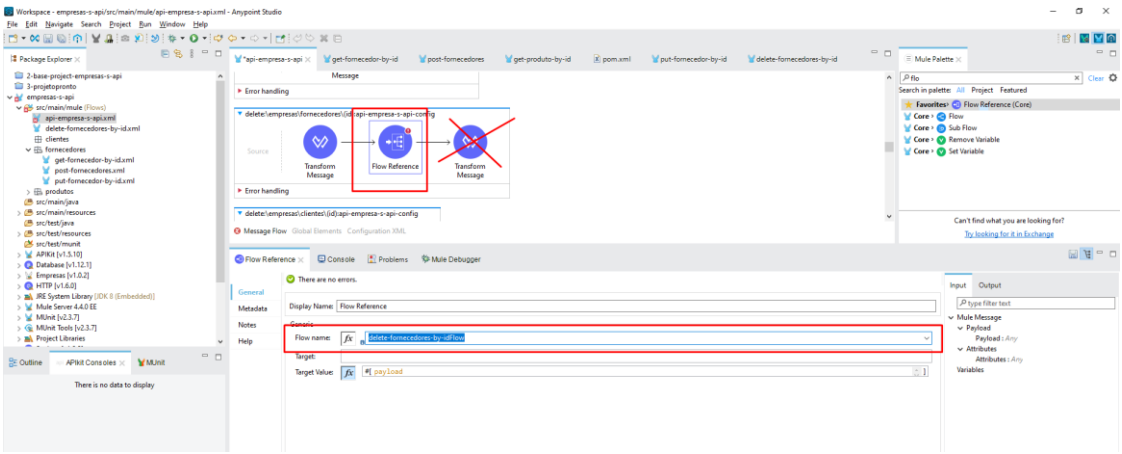
No componente “Set Variable” iremos alterar as seguintes propriedades

- *Name: httpStatus*

- *Value: 200*



No Flow principal iremos incluir o componente “Flow Reference” e alterar a propriedade Flow Name apontando para o novo subFlow criado “delete-fornecedor-by-idFlow”. Vamos também excluir o segundo componente “Transform Message”, pois não vamos utilizá-lo.



Agora iremos rodar e testar nossa API.