Artigo
Jhonata Rebouças · Maio 1, 2021 5min de leitura

Serviço adaptável a diferentes consultas SQL

O mesmo serviço com a possibilidade de receber várias consultas SQL diferentes e sempre entregar o resultado independente de quantas colunas distintas tenham essas diferentes consultas. Aqui demonstro como pode ser possível montar esse tipo de serviço utilizando o Service Bus da Intersystems.

Possível cenário (Desconsiderar o uso de um BI):

Vamos pensar em um painel real time onde iremos fornecer as informações de consumo de um material por região para o setor de compras e teremos as informações do nome do produto, fabricante e quantidade por exemplo.

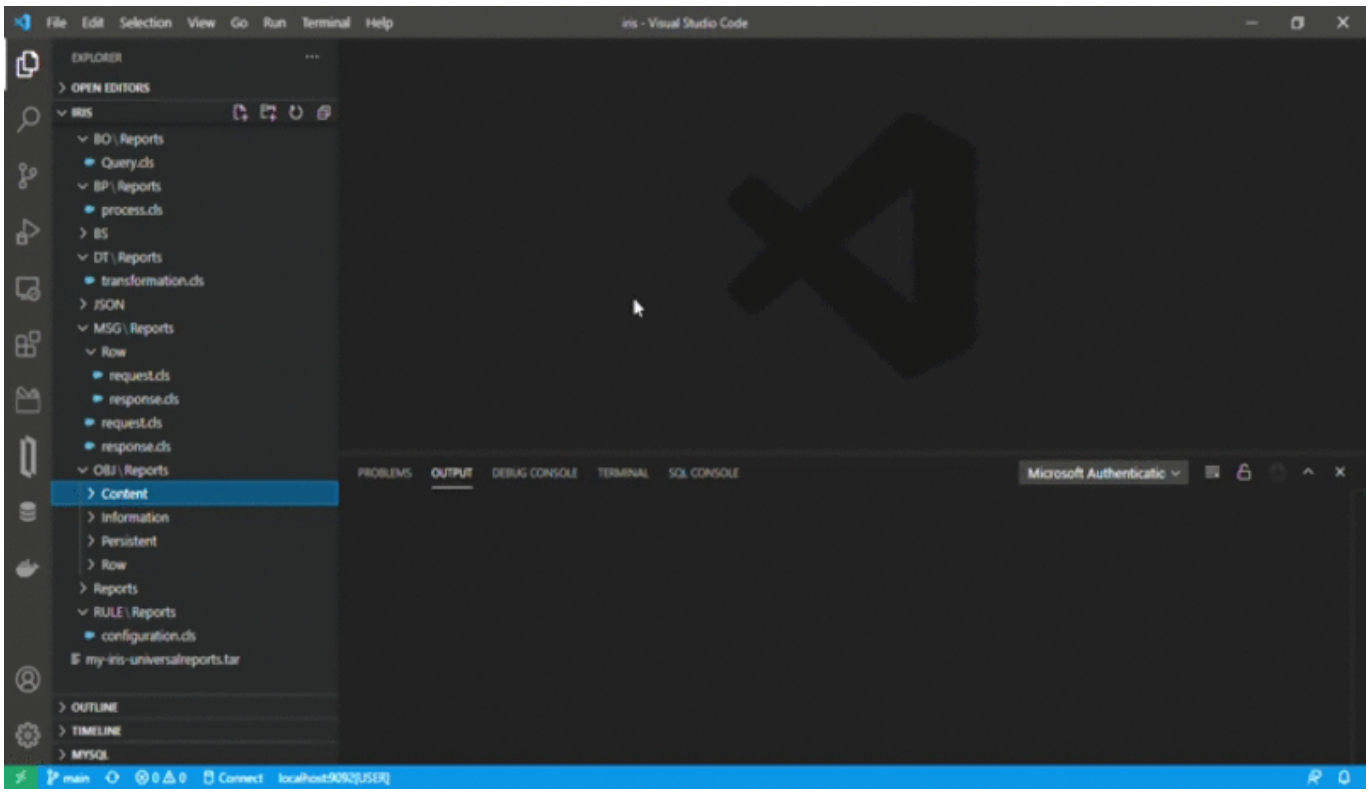
O problema começa quando pensamos que esse painel será modularizado e posteriormente teremos a necessidade de atender aos outros setores que buscam informações totalmente diferentes, afinal um setor como relacionamento pode necessitar de informações relacionadas aos clientes. Nesse caso, criar um novo serviço totalmente do zero seria um grande desperdício de recurso e consequentemente financeiro.

Agora que temos o cenário podemos pensar em uma solução e no caso vamos utilizar o Ensemble da intersystems.
O payload deve ser pensado para expandir e suportar o máximo de colunas possíveis imaginando que um momento pode atender a uma demanda de duas colunas e em outra pode atender a uma demanda de nove ou mais colunas.
A sugestão é adicionar essas colunas em um array e seus dados em outro array relacionado dessa forma pode haver uma flexibilização em sua estrutura.

```
{
    "columns":[
        {
            "description": "String", //Nome da coluna
            "data": [
                {
                    "value": "String" //Valor da coluna
                }
            ]
        }
    ]
}
```

Utilizando o Ensemble e SQL:

Vamos definir a montagem dos objetos e mensagens que serão necessários para criar um payload semelhante, com a adição de algumas informações que serão úteis a identificação do painel, mas podem ser eliminadas ou adaptadas dependendo da necessidade.
Será adicionado informações relacionadas a identificação e descrição do painel que pode ser útil no layout.
Não será abordado o passo a passo da criação dos objetos e mensagens pois não possuem nada de diferente do padrão. **disponível no git hub*



O Business Operation

Possui um pequeno detalhe que fará a grande diferença. Não é necessário se preocupar com as informações adicionais sobre o painel, somente com a consulta SQL. No exemplo usamos uma consulta direta na base do caché, porém é possível utilizar bancos como Oracle, MySql, SqlServer e outros que pode inclusive utilizar Stored Procedures que se torna mais organizado e eficiente.
Ao realizar a consulta é interessante ter conhecimento de quantas colunas essas esse painel retornará e a consulta deverá respeitar esse valor e ordem para que seja montado um retorno correto.

```
For i=1:1:pRequest.columns {
     Set sc=result.Execute()
           While result.Next(.sc) {
                Set tData = ##class(IRIS.JHONATA.OBJ.Reports.Row.row).%New()
                Set tData.row = result.Get("row"_i)
                Do pResponse.data.Insert(tData)
           }
 }
```

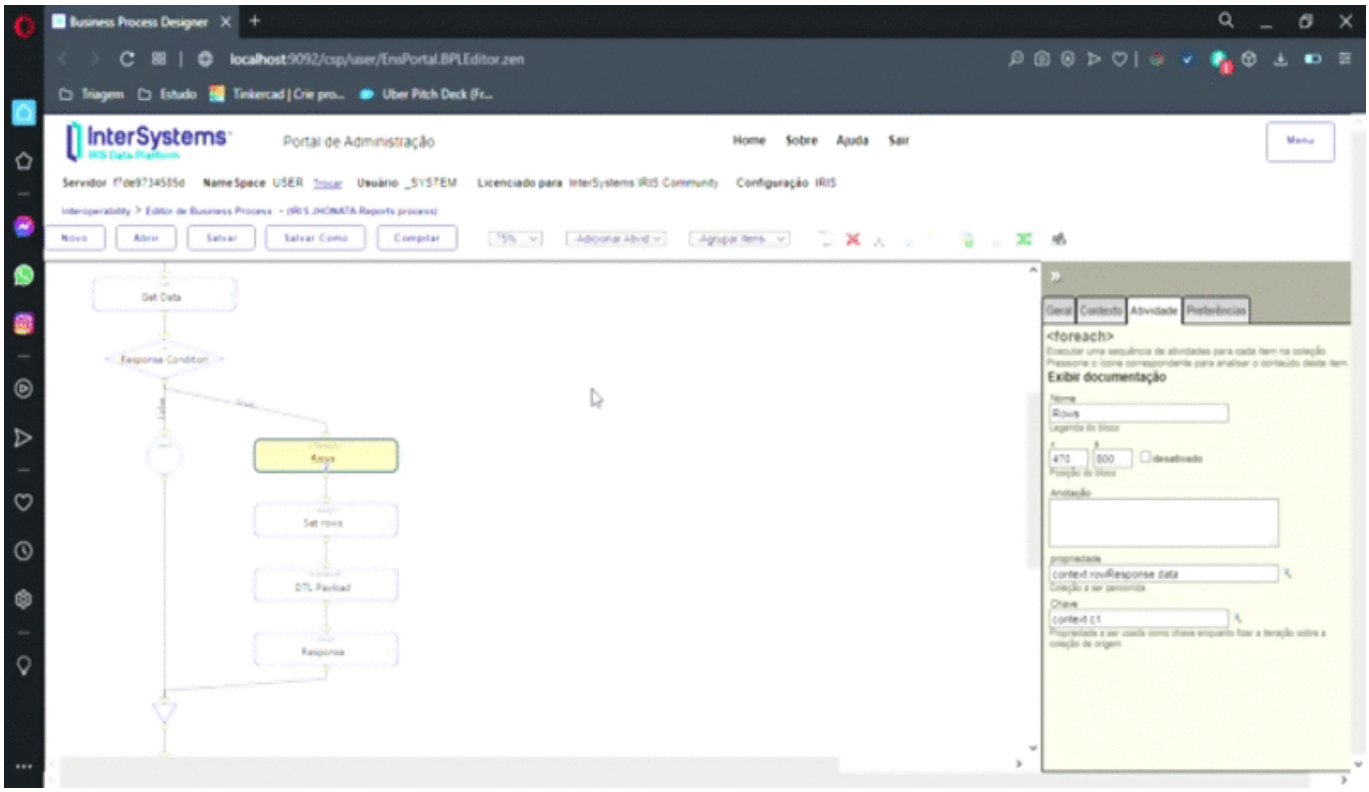
Note que o resultado da consulta é esperado como um alias sequenciado, **result.Get("row"i)** , comparando com a quantidade de colunas garantimos que todos os dados serão recebidos de forma ordenada antes de passar para a próxima coluna.
A consulta ficará semelhante:

```
select name as row1,
    code as row2,
    description as row3
from table
```

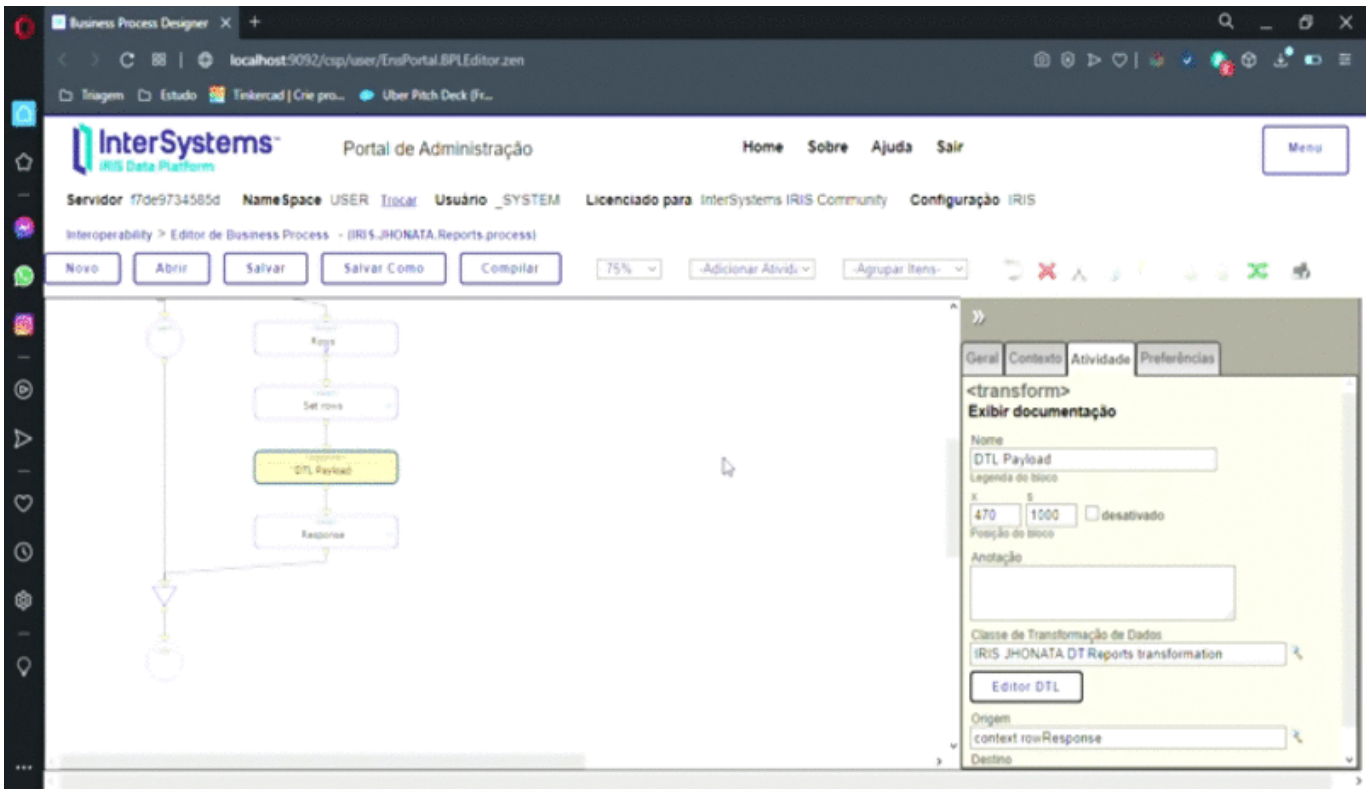
No Operation é interessante receber a procedure ou query como propriedade no request, isso porque poderemos enviar essa informação amarrada a uma regra criada posteriormente e não teremos a necessidade de alterar a classe ou mesmo reiniciar a cada novo painel criado.

O Business Process

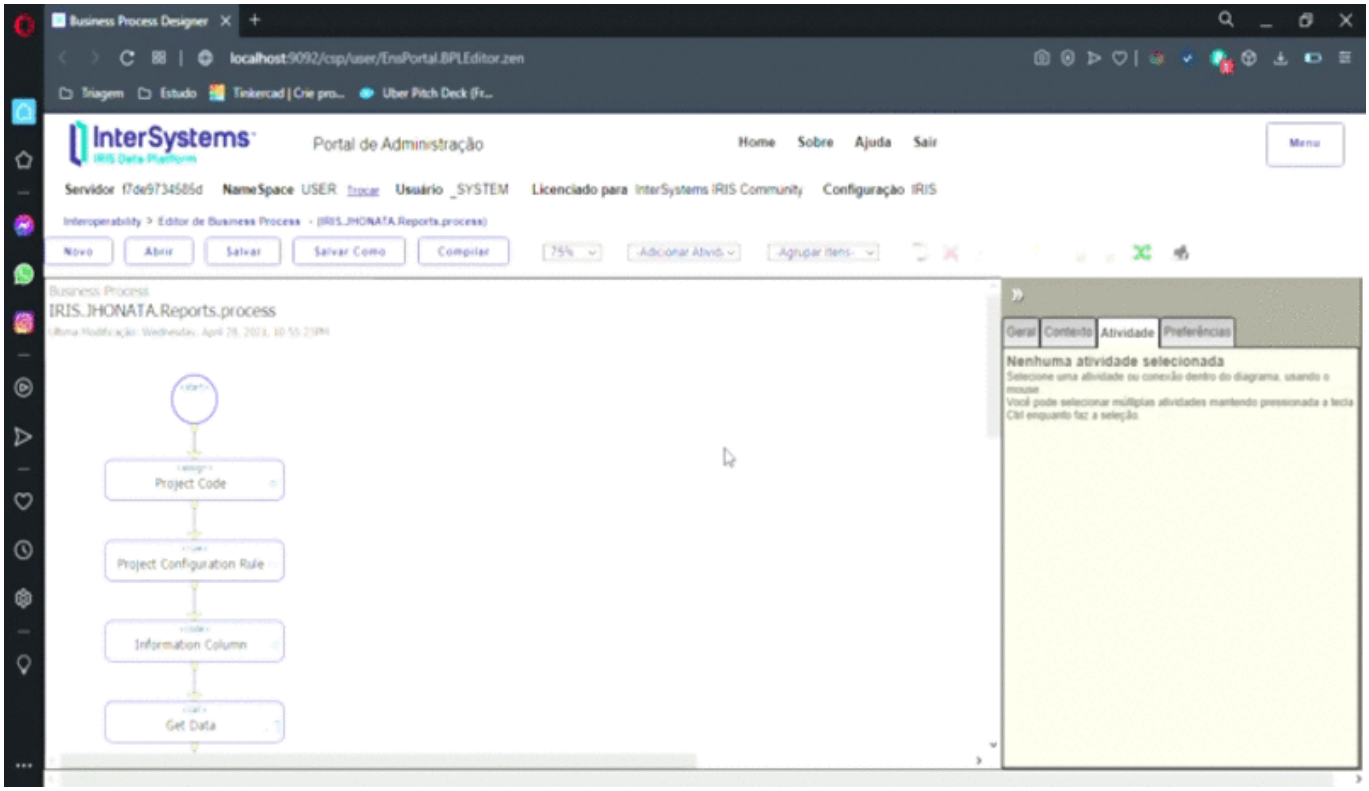
No process temos bastantes detalhes para permitir que tenhamos a adição de novas consultas para os painéis que por ventura forem surgindo sem a necessidade de interferir no que já está em produção. Para não ficar ainda mais extenso, vamos focar somente em algumas atividades principais.
Utilizamos um **foreach** para percorrer os resultados e contabilizar o seu número total de resultados que será importante para identificar em que momento a informação passa a ser pertencente à próxima coluna.



Outro ponto importante é a utilização de um **Data Transformation** para organizar a lista de resultados em colunas com seus respectivos dados associados à coluna correta.

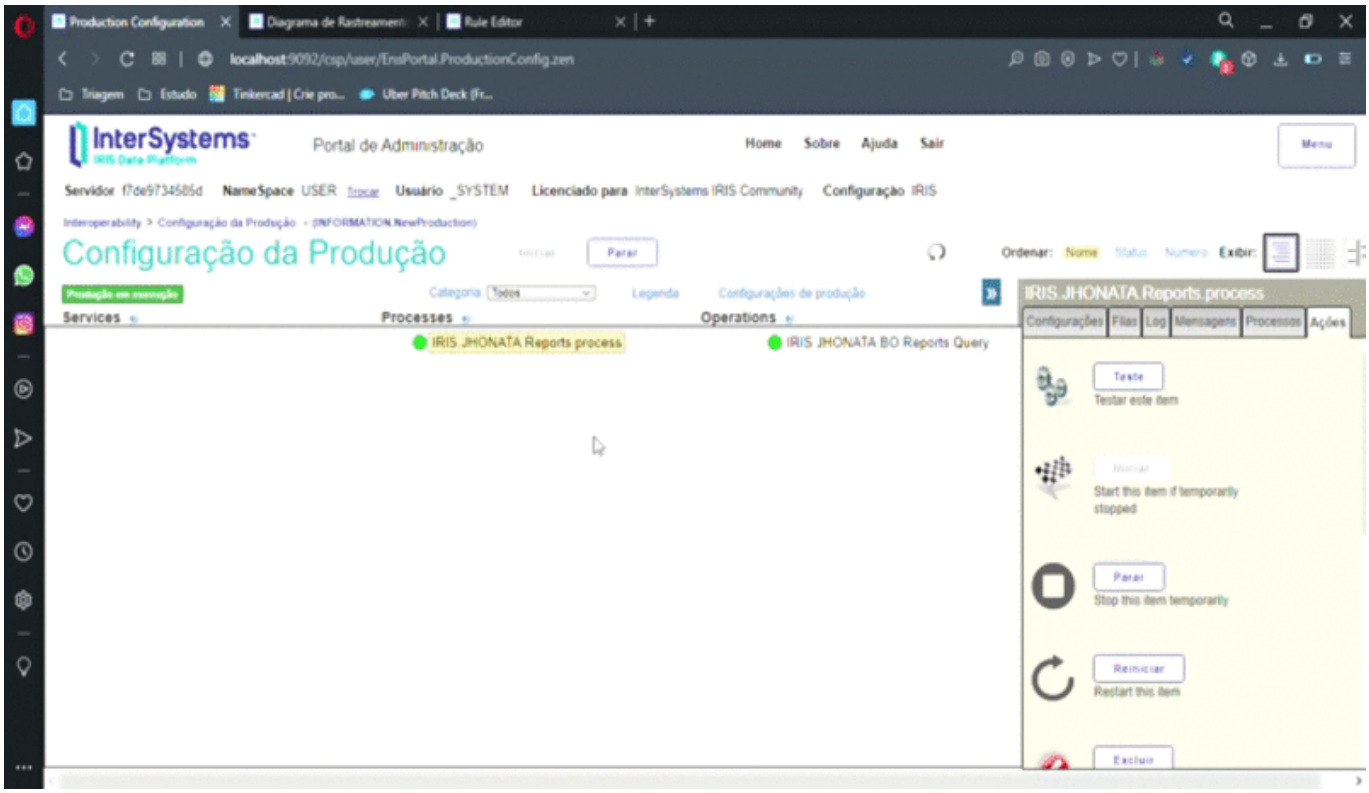


Para que não seja necessário parar o serviço ou impactar seu uso a cada nova funcionalidade ou painel toda parte de consulta ou procedure deve ficar em uma regra identificada pelo tópico do projeto, essa será a única informação necessária no request. Nessa regra pode estar contido informações estáticas do serviço como descrição e nome do painel.
O nome das colunas também devem estar presentes na regra em uma única string com delimitadores e na mesma ordem das colunas retornadas na query para que possa ser contado e utilizado na lógica do Data transformation.



Resultado final:

Pode ser utilizado de várias maneiras, por exemplo através de um serviço REST que informará somente o TOPICO desejado.
Na demonstração existe somente uma regra com uma consulta realizada no caché, e em seguida é gerada uma nova regra com uma nova consulta que já entra pronto para funcionar sem interrupção dos demais.



O Projeto

Todo o projeto pode ser baixado no **GIT HUB** https://github.com/jhonatabf/irisuniversalreport e a tabela de exemplo está na pasta OBJ/Persistent

#Operação de negócios #Processo de negócio #Regras do negócio #DTL #REST API #Servidor de aplicação #SQL #Caché #Ensemble #InterSystems IRIS

URL de origem:https://pt.community.intersystems.com/post/servi%C3%A7o-adapt%C3%A1vel-diferentes-consultas-sql