Assistente de consulta

Para criar as consultas ao banco de dados, o Cronapp possui seu Assistente de consulta visual que simplifica esse processo.

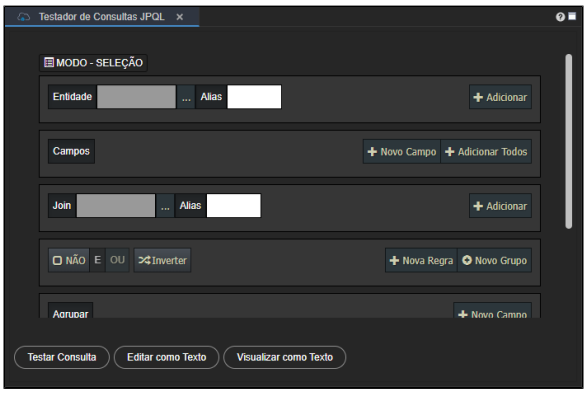

Figura 1 - Assistente de Consulta

Nesta página

- Acesso ao Assistente
  - Blocos de programação
  - Fonte de Dados
  - Testador de Consultas
- Estrutura da Consulta
  - Entidade
    - JPQL e SQL Nativo
    - Fonte de dados
  - Campos
    - JPQL
    - SQL Nativo
    - Fonte de dados
  - Join
  - Regras
  - Filtro
  - Agrupar
  - Having
  - Ordenar
    - JPQL e SQL Nativo
    - Fonte de dados
  - SQL Nativo
  - Modo de Consulta
  - Aplicar dados da requisição
  - Habilitar paginação
- Diferença entre consultas
  - JPQL e SQL
  - REST

Acesso ao Assistente

Atualmente existem três formas de acesso ao Assistente de consulta visual:

- Por meio de alguns blocos de programação do tipo servidor, que estão na categoria Banco de Dados.
- Através do assistente de consulta de uma Fonte de Dados do tipo **Entidade** e **SQL Nativo**.
- Utilizando o Testador de Consultas JPQL.

Vamos detalhar abaixo cada uma dessas formas de acesso.

Blocos de programação

Uma das formas de acesso ao Assistente de consulta visual é por meio de alguns blocos de programação da categoria Banco de Dados. Para abrir o Assistente, basta clicar no ícone de engrenagem de cada bloco (Figura 1.1). O bloco Abrir consulta (destaque 1 da Figura 1.1) exibe a mesma estrutura do Assistente visual detalhada no tópico Estrutura da Consulta (Figura 2). Já os blocos de manipulação de dados (DML - *Data Manipulation Language*) possuem algumas diferenças. Para mais detalhes, consulte a documentação específica de cada bloco abaixo.

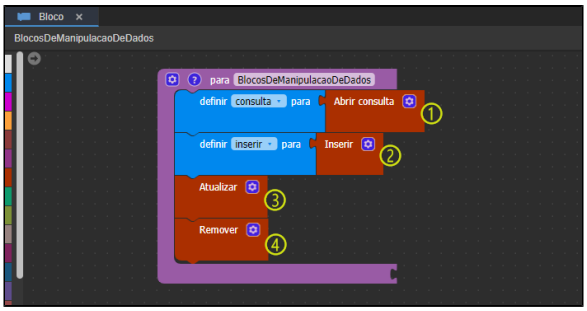

Figura 1.1 - Blocos de acesso ao Assistente de consulta visual

Blocos do tipo servidor para acesso ao assistente visual:

1. **Abrir consulta:** cria uma consulta no banco de dados por meio do assistente visual.
2. **Inserir**: adiciona novos registros ao banco de dados por meio do assistente visual.
3. **Atualizar:** atualiza registros no banco de dados por meio do assistente visual.
4. **Remover**: remove registros do banco de dados por meio do assistente visual.

Fonte de Dados

O Assistente de consulta também pode ser acessado através de uma Fonte de Dados do tipo **Entidade** ou SQL Nativo. Na imagem abaixo, vemos como configurar uma Fonte de dados e acessar o Assistente de consulta por meio dela.

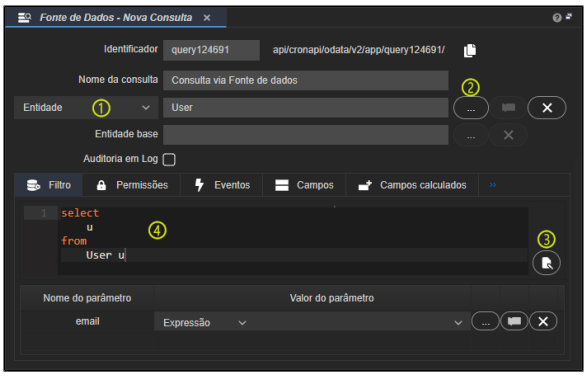

Figura 1.2 - Fonte de dados vinculada a uma entidade e sua consulta

Destaques da Figura 1.2:

1. **Tipo da fonte**: define o tipo da origem dos dados, podendo ser Entidade, SQL Nativo, Bloco de programação ou Web Services. Como informado anteriormente, apenas os tipos **Entidade** e **SQL Nativo** permitem o acesso ao Assistente de consulta, escolha uma dessas duas opções.
2. " **...** " : seleciona a origem dos dados. Essa seleção irá variar a depender do tipo da fonte escolhida. Se for do tipo **Entidade**, será possível selecionar uma das classes de qualquer Diagrama de dados (*namespace*) do projeto. Por outro lado, se for do tipo **SQL Nativo**, poderá selecionar um dos *namespaces* do projeto.
3. **Editar**: abre o Assistente de consulta e permite personalizá-la. Esse botão estará desabilitado em Fontes de dados que não possuem o Assistente.
4. **Consulta**: exibe a consulta JPQL para a fonte do tipo **Entidade**, ou SQL para Fonte do tipo **SQL Nativo**.

A imagem a seguir apresenta o Assistente de consulta aberto após clicar no botão "Editar" (destaque 3 da Figura 1.2). Nele é possível personalizar a consulta realizada.

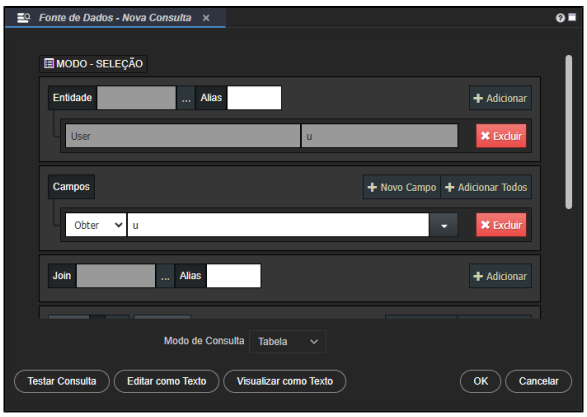

Figura 1.3 - Assistente de consulta da Fonte de dados

Testador de Consultas

Outra forma de acessar o Assistente de consulta é utilizando o **Testador de Consultas JPQL**, acessível a partir do menu do sistema em **Ferramentas** > **Testador de Consultas JPQL** (Figura 1.4).

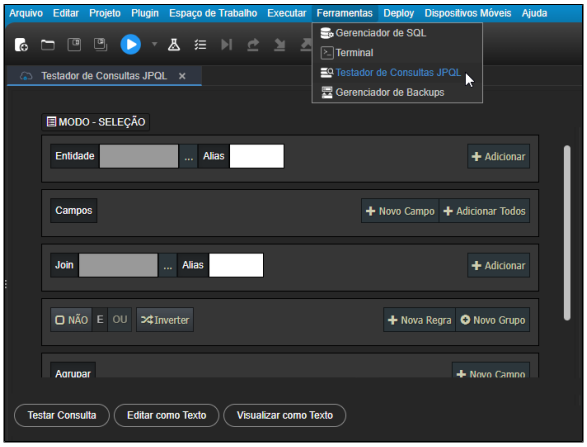

Figura 1.4 - Testador de Consultas JPQL

Estrutura da Consulta

O Assistente de consulta visual gera consultas em: JPQL (*Jakarta Persistence Query Language,* conhecido antigamente como *Java Persistence Query Language*), uma linguagem de consulta ORM que atua sobre classes e objetos; SQL (*Structured Query Language*), linguagem que trabalha com banco de dados relacionais e realiza as consultas sobre tabelas; e em requisições do tipo REST (*Representational State Transfer*), quando o assistente é utilizado em um bloco de programação ou no Testador de Consultas JPQL para gerar uma consulta a partir de uma Fonte de Dados, acesse o tópico REST para mais detalhes.

É importante destacar que o Assistente de consulta pode apresentar diferenças, dependendo da forma de acesso e o tipo de consulta realizada. As funcionalidades apresentadas neste tópico se referem ao Assistente acessado por meio do bloco de programação Abrir consulta, visto que possui mais campos para configurar.

Para obter dados que necessitam de consultas mais complexas, é possível desenvolver consultas de forma *high-code*, utilizando o JPQL com base no EclipseLInk do JPA.

Figura 2 - Assistente de consulta visual (Bloco de programação)

1. **Entidade**: adiciona as entidades, tabelas ou Fonte de dados que farão parte da consulta.
2. **Campos**: define quais campos serão buscados na consulta.
3. **Join**: combina colunas de uma ou mais entendidas ou tabelas, usando valores comuns a cada uma delas.
4. **Regras**: regra que será considerada ao buscar os registros.
5. **Agrupar**: agrupa linhas que possuem valores iguais nos mesmos campos.
6. **Having**: cria regras para serem usadas junto com os campos agrupados.
7. **Ordenar**: define a ordem em que os dados serão retornados.
8. **SQL Nativo**: permite o uso de consultas do tipo SQL nativo.
9. **Modo de Consulta**: altera a exibição dos dados em modo de teste da consulta, que pode ser por tabela ou objetos.
10. **Aplicar dados da requisição:** aplica as requisições (paginação, ordenação e outros) diretamente na consulta.
11. **Habilitar Paginação:** realiza a paginação dos dados em telas de forma automática.
12. **Testar Consulta:** exibe o resultado da consulta criada em uma nova janela, altere o campo Modo de Consulta para modificar a forma de exibição.
13. **Editar como Texto:** muda a janela para editar a consulta manualmente (*high-code*). Após alterar uma consulta manualmente, não será possível editar novamente em modo visual.
14. **Visualizar como Texto:** exibe a consulta gerada através da configuração visual.
15. **OK e Cancelar:** salva ou cancela a consulta.

Entidade

Adiciona as entidades, tabelas ou Fonte de dados que farão parte da consulta. Equivale a seleção de tabelas após a sentença "`FROM`" em uma consulta SQL. Dependendo do tipo de consulta selecionada no campo **Entidade**, alguns campos serão ou não exibidos. Veja o tópico Diferença entre consultas.

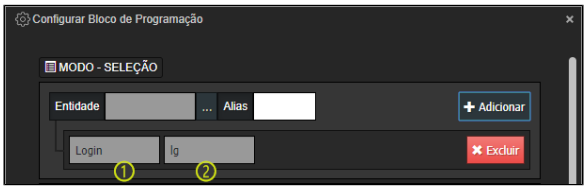

Figura 2.1 - Área de seleção das entidades

Campos e botões:

- **Entidade**: exibe a entidade, tabela ou Fonte de dados recém selecionada.
- " **...** " : abre a janela para seleção.
- **Alias**: permite adicionar um apelido.
- **+ Adicionar**: confirma a selecionada na consulta.
- **Excluir**: remove a entidade, tabela ou Fonte de dados selecionada.

Destaques da Figura 2.1:

1. Exibe a entidade, tabela ou Fonte de dados selecionada.
2. Exibe o alias configurado para a entidade, tabela ou Fonte de dados selecionada.

Para adicionar à consulta, clique no botão **"..."** do campo **Entidade**, selecione a classe, tabela ou a Fonte de dados desejada e finalize clicando em "**+ Adicionar**".

JPQL e SQL Nativo

Para realizar consultas do tipo **SQL Nativo** no Assistente de consulta dos bloco de programação, será necessário selecionar a *checkbox* SQL Nativo. Consulte o tópico correspondente nesta documentação para mais informações. A consulta definida na Figura 2.1, em **JPQL** e **SQL Nativo**, ficará da seguinte forma:

```
FROM Login lg
```

Fonte de dados

É importante destacar que as consultas do tipo Fonte de dados gera uma requisição REST. Consulte o tópico correspondente nesta documentação para mais informações. A consulta definida na Figura 2.1 ficará da seguinte forma:

```
GET app/login
```

Campos

Define quais campos serão buscados na consulta. Equivale a seleção dos campos das tabelas após a sentença "`SELECT`" em uma consulta SQL.

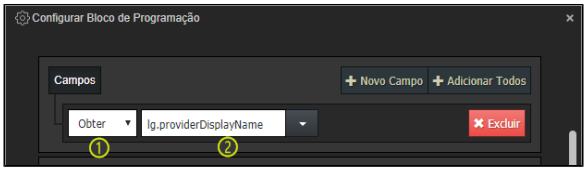

Figura 2.2 - Área de seleção dos campos

É possível adicionar os campos desejados através do botão "**+ Novo Campo**" ou adicionar todos os campos da entidade de uma só vez, clicando em "**+ Adicionar Todos**".

Botões:

- **+ Novo Campo**: adiciona um campo das entidades selecionadas ou de entidades relacionadas.
- **+ Adicionar Todos**: insere todos os campos das entidades selecionadas.
- **Excluir**: exclui o campo.

Destaques da Figura 2.2:

1. **Tipo de retorno:** define como o valor do campo será retornado. Os tipos são:
    - **Obter:** retorna o valor do campo como está no banco.
    - **MIN:** retorna o menor valor do campo selecionado.
    - **MAX:** retorna o maior valor do campo selecionado.
    - **AVG:** retorna a média aritmética dos valores do campo selecionado.
    - **SUM:** retorna a soma dos valores do campo selecionado.
    - **COUNT:** retorna a quantidade de registros do campo selecionado.
2. **Campo selecionado:** seleciona qual o campo será consultado.

Para mais detalhes sobre como utilizar o Tipo de retorno, acesse o tópico **Aggregation functions** (Funções de agregação) na documentação oficial.

JPQL

Em consultas JPQL, a área de seleção dos campos já mapeia todos os campos das entidades relacionadas (terceiro "obter" da imagem abaixo), por isso não é necessário usar o `JOIN`. Por exemplo, imagine que uma entidade `A` tem relação com `B`, a entidade `B` tem relação com `C` e `C` possui um campo chamado "`nome`", dessa forma, basta selecionar a entidade `A` e na área **Campos** selecionar a opção "`A.B.C.nome`", criando um `JOIN` automaticamente.

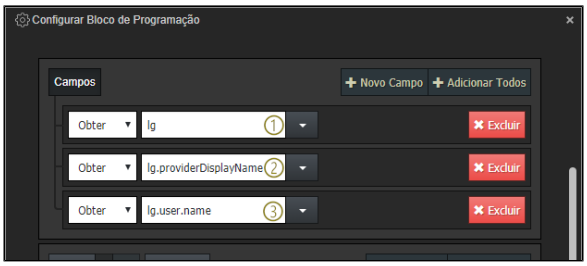

Figura 2.2.1 - Área de seleção dos campos de uma consulta JPQL

Destaques da Figura 2.2.1:

1. Para retornar todos os campos de uma entidade, basta selecionar o *alias* da entidade desejada.
2. Para retornar um campo específico, é necessário selecionar o alias da entidade com o campo desejado.
3. Caso queira obter um campo de uma entidade relacionada (`JOIN`), selecione o alias da entidade, junto com o nome da entidade relacionada e o nome do campo.

Por padrão, em consultas JQPL a caixa de seleção dos campos só exibe relacionamentos até o terceiro nível. Para relacionamentos maiores, informe manualmente. Exemplo: `EntidadeA.EntidadeB.EntidadeC.EntidadeD.EntidadeE.atributo`.

A consulta definida na Figura 2.2.1 ficará da seguinte forma:

```
SELECT lg, lg.providerDisplayName, lg.user.name FROM Login lg, User usr
```

SQL Nativo

Em consultas do tipo SQL Nativo, diferente de uma consulta JPQL, a área de seleção dos campos mapeia apenas em primeiro nível os campos das tabelas.

Figura 2.2.2 - Área de seleção dos campos de uma consulta SQL Nativo

A consulta definida na Figura 2.2.2 ficará da seguinte forma:

```
SELECT LG.ID, LG.PROVIDER_DISPLAY_NAME FROM LOGIN LG
```

Além disso, não é possível retornar todos os campos de uma tabela selecionando o alias, para lidar com vários níveis de relacionamento, será necessário utilizar o `JOIN`. Consulte o tópico correspondente para mais informações.

Fonte de dados

Em consultas do tipo Fonte de dados, a área de seleção dos campos mapeia apenas em primeiro nível os campos da Fonte de dados. Além disso, não é possível retornar todos os campos de uma tabela selecionando o *alias*.

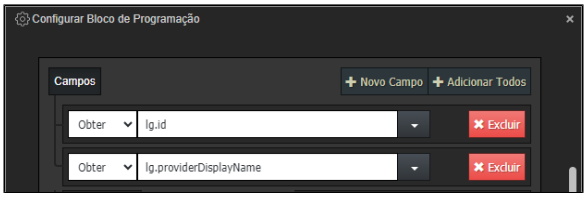

Figura 2.2.3 - Área de seleção dos campos de uma consulta Fonte de dados

A consulta REST definida na Figura 2.2.3 ficará da seguinte forma:

```
GET app/login?$select=id%2CproviderDisplayName
```

Join

Essa opção é exibida apenas em consultas do tipo **JPQL** e **SQL Nativo**. É uma operação de junção que combina colunas de uma ou mais tabelas, usando valores comuns a cada uma delas. Equivale a sentença "`JOIN`" em uma consulta SQL.

Como já informado, em consultas JPQL não é necessário usar o JOIN, pois a área de seleção dos campos já mapeia todos os campos das entidades relacionadas. No entanto, você pode utilizá-lo em consultas do tipo SQL Nativo, como no exemplo da imagem abaixo.

As cláusulas JOIN disponíveis são apenas as suportadas pelo EclipseLink, "`JOIN`" e "`LEFT JOIN`". Para mais detalhes, acesse os subtópicos do tópico **FROM CLAUSE** (Cláusula FROM) na documentação oficial.

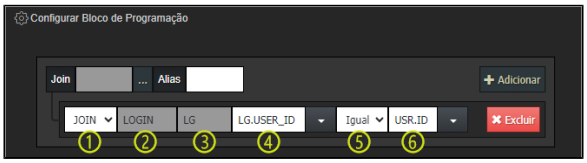

Figura 2.3 - Área de seleção do Join

Campos e botões:

- **Join**: exibe a entidade recém selecionada.
- " **...** " : abre a janela para seleção da entidade.
- **Alias**: permite adicionar um apelido.
- **+ Adicionar**: adiciona a entidade selecionada à consulta.
- **Excluir**: exclui entidade adicionada.

Destaques da Figura 2.3:

1. **Cláusula**: tipo da cláusula JOIN utilizada. São elas:
    - **JOIN**: retorna apenas os registros que têm correspondências em ambas as entidades envolvidas na junção.
    - **LEFT JOIN**: retorna todos os registros da entidade à esquerda e os registros correspondentes da entidade à direita.
2. **Entidade**: exibe a entidade selecionada.
3. **Alias**: exibe o alias configurado para a entidade selecionada.
4. **Campo da entidade**: seleciona qual o campo da entidade será usado para consulta.
5. **Condição**: condição que será utilizada para comparar os valores das entidades. São elas:
    - Igual.
    - Diferente.
    - Menor.
    - Menor ou igual.
    - Maior.
    - Maior ou igual.
6. **Parâmetro**: campo que será usado como comparação.

A consulta definida na Figura 2.3, em **JPQL** e **SQL Nativo**, ficará da seguinte forma:

```
JOIN LOGIN LG ON LG.USER_ID = USR.ID;
```

Regras

Essa opção é exibida apenas em consultas do tipo **JPQL** e **SQL Nativo**. Nessa propriedade é definida as regras que serão consideradas ao buscar os dados no banco. Por meio do botão "**+ Nova Regra**" é possível adicionar regras e selecionar um condicional único entre as regras, ou adicionar vários grupos de regras e selecionar condicionais para cada grupo individualmente através do botão "**+ Novo Grupo**". As regras equivalem ao filtro após a sentença "WHERE" em uma consulta SQL. Caso tenha alguma dúvida sobre a regra WHERE, acesse o tópico **WHERE Clause** (Cláusula WHERE) na documentação oficial.

Figura 2.4 - Área de seleção das regras da consulta

Botões:

- **+ Nova Regra**: adiciona uma regra vinculada ao campo selecionado.
- **+ Novo Grupo**: insere um grupo com uma ou mais regras.
- **Excluir**: exclui a regra ou grupo selecionado.

Destaques da Figura 2.4:

1. **Negação**: nega a sentença vinculada, adicionando um "not" antes.
2. **Operador lógico**: define se o operador lógico entre as sentenças será "e" ou "ou".
3. **Inverter**: alterna o operador lógico e os campos condicionais das sentenças vinculadas, por exemplo: se estiver selecionado "ou", será substituído por "e", isso também ocorrerá no campo condicional (campo 5 da Figura 2.4), que alternará de "igual" para "diferente".
4. **Campo da entidade**: campo da entidade que será utilizado na criação da regra.
5. **Condição:** define a condição que deve existir entre o campo e o parâmetro, são elas:
    - Igual.
    - Diferente.
    - Contido.
    - Não contido.
    - Iniciando com.
    - Não iniciando com.
    - Contém.
    - Não contém.
    - Terminando com.
    - Terminado sem.
    - É vazio.
    - Não é vazio.
    - É nulo.
    - Não é nulo.
6. **Tipo de parâmetro:** tipo do parâmetro que será utilizado na consulta, são eles:
    - **Parâmetro**: ao selecionar esse tipo, no campo "Parâmetro" é possível definir uma variável adicionando ":" (dois pontos) no início dela (destaque 7). Ao inserir um conteúdo sem os ":" (dois pontos) no início, este conteúdo ficará entre aspas (') e será tratado como uma *string*.
    - **Personalizado**: ao selecionar esse tipo, no campo "Parâmetro" também é possível definir uma variável adicionando ":" (dois pontos) no início dela (destaque 7). No entanto, deve ser utilizado com o objetivo de passar um conteúdo estático personalizado, como uma *string* ou uma lista de elementos. Nesse caso, as aspas (') não serão inseridas automaticamente, o próprio usuário deverá realizar esse tratamento (destaque 9).
7. **Parâmetro:** define o nome do parâmetro que será comparado com o campo da entidade. O conteúdo informado neste campo deve estar de acordo com o "Tipo de parâmetro" selecionado. Como informado anteriormente, neste campo é possível definir uma variável adicionando ":" (dois pontos) no início dela (destaque 7) ou um conteúdo estático personalizado (destaque 9).
8. : clique e arraste o ícone para modificar a posição de uma regra ou grupo.

A regra definida na Figura 2.4, em **JPQL** ou **SQL Nativo**, ficará da seguinte forma:

```
WHERE usr.phoneNumber = :phoneNumber AND (NOT (usr.name = ('Maria') ) )
```

Filtro

Essa opção é exibida apenas em consultas do tipo **Fonte de dados**. Este campo permite filtrar ou restringir os registros de uma consulta, semelhante ao campo Regras. No entanto, possui algumas limitações, já que sua consulta será convertida em uma requisição REST. A imagem abaixo detalha os recursos dessa funcionalidade.

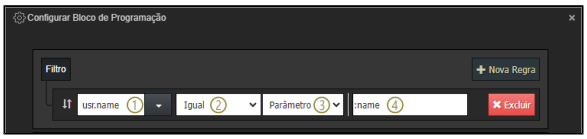

Figura 2.5 - Área de seleção dos filtros da consulta

Botões:

- **+ Nova Regra**: adiciona um filtro vinculado ao campo selecionado.
- **Excluir**: exclui o filtro selecionado.

Destaques da Figura 2.5:

1. **Campo da Fonte de dados**: campo da Fonte de dados que será utilizado na criação do filtro.
2. **Condição:** define a condição que deve existir entre o campo e o parâmetro, são elas:
    - Igual.
    - Diferente.
    - Contido.
    - Não contido.
    - Iniciando com.
    - Não iniciando com.
    - Contém.
    - Não contém.
    - Terminando com.
    - Terminado sem.
    - É vazio.
    - Não é vazio.
    - É nulo.
    - Não é nulo.
3. **Tipo de parâmetro:** tipo do parâmetro que será utilizado na consulta, são eles:
    - **Parâmetro**: ao selecionar esse tipo, no campo "Parâmetro" é possível definir uma variável adicionando ":" (dois pontos) no início dela (destaque 4). Ao inserir um conteúdo sem os ":" (dois pontos) no início, este conteúdo ficará entre aspas (") e será tratado como uma *string*.
    - **Personalizado**: ao selecionar esse tipo, no campo "Parâmetro" também é possível definir uma variável adicionando ":" (dois pontos) no início dela (destaque 7). No entanto, deve ser utilizado com o objetivo de passar um conteúdo estático personalizado, como uma *string* ou uma lista de elementos.
4. **Parâmetro:** define o nome do parâmetro que será comparado com o campo da entidade. Neste campo é possível definir uma variável adicionando ":" (dois pontos) no início dela (destaque 4) ou um conteúdo estático personalizado.

A regra definida na Figura 2.5 ficará da seguinte forma na requisição REST:

```
filter=%28name%20eq%20%3Aname%29
```

Agrupar

Essa opção é exibida apenas em consultas do tipo **JPQL** e **SQL Nativo**. Possui a função de agrupar linhas que têm os mesmos valores em um ou mais campos específicos. Equivale a sentença "`GROUP BY`" em uma consulta SQL. Para mais detalhes sobre o agrupar, acesse o tópico **GROUP BY Clause** (Cláusula GROUP BY) na documentação oficial.

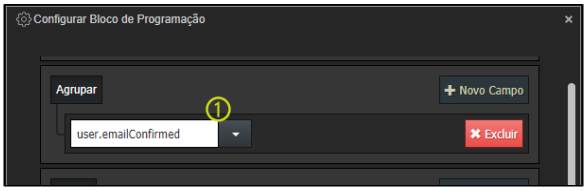

Figura 2.6 - Área de seleção do Agrupar

Botões:

- **+ Novo Campo**: adiciona outro campo para agrupar.
- **Excluir**: exclui o campo selecionado.

Destaques da Figura 2.6:

1. **Campo a ser agrupado**: seleciona qual o campo será agrupado.

A regra definida na Figura 2.6, em **JPQL** e **SQL Nativo**, ficará da seguinte forma:

```
GROUP BY user.emailConfirmed
```

Having

Essa opção é exibida apenas em consultas do tipo **JPQL** e **SQL Nativo**. A propriedade **Having** funciona de forma semelhante a propriedade Regras, entretanto ela cria uma regra para os grupos criados com a propriedade Agrupar. Por isso, o **Having** sempre deve ser usado junto com o **Agrupar**. Possui a mesma função do "`HAVING`" em consultas SQL. Para mais detalhes sobre a condição having, acesse o tópico **H AVING Clause** (Cláusula HAVING) na documentação oficial.

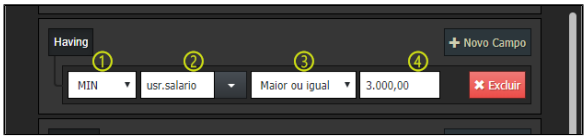

Figura 2.7 - Área de seleção das regras do Having

Botões:

- **+ Novo Campo**: adiciona outro campo para criar uma regra having.
- **Excluir**: exclui o campo selecionado.

Destaques da Figura 2.7:

1. **Tipo de dados da condição:** é definido qual o tipo será retornado do banco para ser comparado na regra criada pelo having. Os tipos são:
    - **MIN:** retorna o menor valor do campo selecionado.
    - **MAX:** retorna o maior valor do campo selecionado.
    - **AVG:** retorna a média aritmética dos valores do campo selecionado.
    - **SUM:** retorna a soma dos valores do campo selecionado.
    - **COUNT:** retorna a quantidade de registros do campo selecionado.
2. **Campo:** campo usado para a comparação com o parâmetro.
3. **Condição:** condição que será utilizada para comparar o valor obtido no campo e o parâmetro. São eles:
    - Igual.
    - Diferente.
    - Menor.
    - Menor ou igual.
    - Maior.
    - Maior ou igual.

4. **Parâmetro:** valor que será comparado com o tipo de dado retornado. Neste campo é possível definir uma variável adicionando ":" (dois pontos) no início dela (destaque 4) ou um conteúdo estático personalizado.

A regra definida na Figura 2.7, em **JPQL** e **SQL Nativo**, ficará da seguinte forma:

```
HAVING MIN(usr.salario) >= 3.000,00
```

Ordenar

Define a ordem de retorno dos itens. Por exemplo, é possível trazer a lista de funcionários ordenada pelos maiores salários. Possui a mesma função do "`ORDER BY`" em consultas SQL. Para mais detalhes sobre a condição order by, acesse o tópico **ORDER BY Clause** (Cláusula ORDER BY) na documentação oficial.

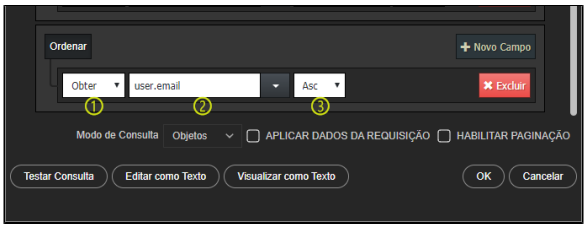

Figura 2.8 - Área de ordenação da consulta

Botões:

- **+ Novo Campo**: adiciona outro campo para ordenar.
- **Excluir**: exclui o campo selecionado.

Destaques da Figura 2.8:

1. **Tipo de dado**: define qual o tipo de dado que será utilizado para ordenar os dados.
    - **Obter:** retorna o valor do campo como está no banco.
    - **MIN:** retorna o menor valor do campo selecionado.
    - **MAX:** retorna o maior valor do campo selecionado.
    - **AVG:** retorna a média aritmética dos valores do campo selecionado.
    - **SUM:** retorna a soma dos valores do campo selecionado.
    - **COUNT:** retorna a quantidade de registros do campo selecionado.
2. **Campo**: campo que será utilizado para a ordenação.
3. **Tipo de ordenação**: define o tipo de ordenação que será considerada na propriedade.
    - **Asc**: ascendente.
    - **Desc**: descendente.

JPQL e SQL Nativo

A regra definida na Figura 2.8, em **JPQL** e **SQL Nativo**, ficará da seguinte forma:

```
ORDER BY user.email ASC
```

Fonte de dados

A regra definida na Figura 2.8, a partir de uma Fonte de dados, ficará da seguinte forma:

```
$orderby=email%20ASC
```

SQL Nativo

Funcionalidade presente apenas no Assistente de consulta dos blocos de programação. Por padrão, o Cronapp utiliza o JPQL para realizar consultas ao banco de dados. Porém ao ativar a opção **SQL Nativo** (destaque 8 da Figura 2) é possível utilizar consultas SQL nativas, possibilitando o uso de diversos recursos não suportados pelo JPQL. Ao selecionar a *checkbox* dessa funcionalidade, será possível escolher um diagrama de dados (*namespace*) da aplicação e as classes desse diagrama serão exibidas no campo Entidade para gerar uma consulta (Figura 2.9).

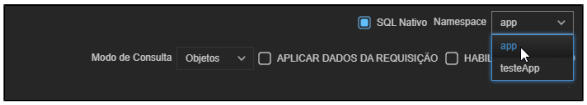

Figura 2.9 - Opção SQL Nativo selecionada

Modo de Consulta

Essa funcionalidade é exibida apenas ao acessar o Assistente visual por bloco de programação e pela Fonte de dados do tipo **Entidade** e **SQL nativo**. A opção Modo de Consulta (destaque 9 da Figura 2) altera a visualização dos dados ao testar a consulta, que pode ser por objetos (Figura 2.10) ou tabelas (Figura 2.11).

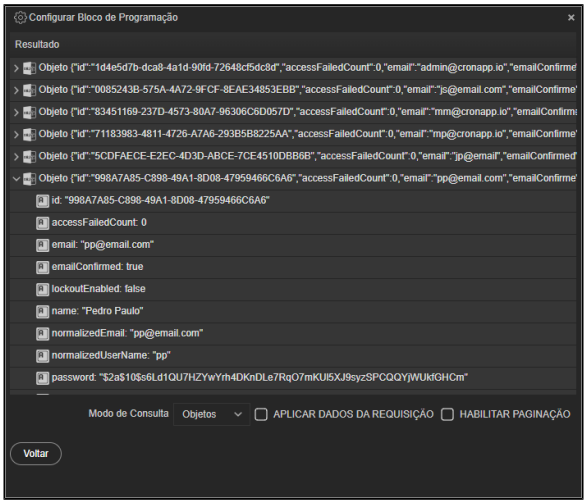

Figura 2.10 - Exibição de consulta em modo objetos

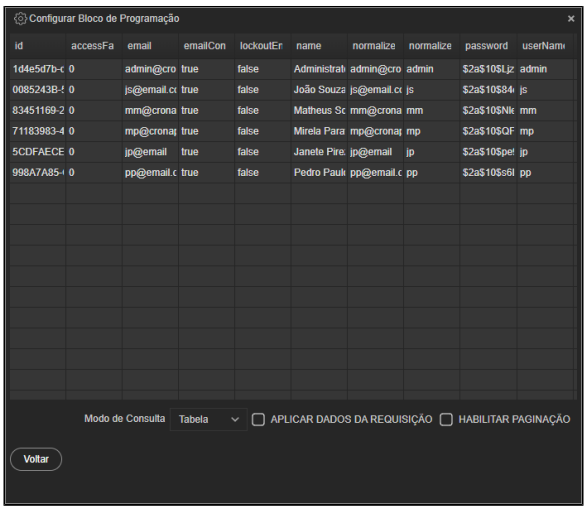

Figura 2.11 - Exibição de consulta em modo tabela

Aplicar dados da requisição

Funcionalidade presente apenas no Assistente de consulta do bloco de programação Abrir consulta. Nas versões anteriores do Cronapp existia uma opção chamada "Paginação Automática". Essa opção aplicava os dados de paginação da requisição da Fonte de dados à consulta. O Cronapp evoluiu essa opção que passou a se chamar **Aplicar Dados da Requisição** (destaque 10 da Figura 2). Essa nova opção aplica paginação, ordenação, filtros e qualquer outro dado da requisição **ODATA** à consulta. Com isso, todas as operações de interface serão enviadas à consulta, fazendo com que a Fonte de dados funcione integralmente com a consulta do bloco de programação.

Habilitar paginação

Assim como a anterior, essa funcionalidade está presente apenas no Assistente de consulta do bloco de programação Abrir consulta. Ela permite paginar os dados exibidos na tela. Para isso, utilize o bloco Abrir consulta e selecione a opção **Habilitar paginação** (destaque 11 da Figura 2), ao fazer isso, o bloco exibirá os parâmetros **limit** e **offset** (Figura 2.12).

- **limit**: define a quantidade de linhas retornadas na consulta.
- **offset**: define a partir de qual linha retornará.

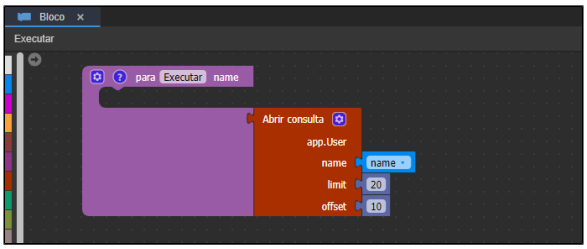

Figura 2.12 - Bloco com a opção Habilitar paginação ativa

Exemplo: Em uma consulta com 100 registros, ao definir os parâmetros **limit** com '20' e o **offset** com '10', o resultado retornará os registros de 10 a 30.

Diferença entre consultas

JPQL e SQL

Como informado anteriormente, o Assistente de consulta visual gera consultas em JPQL, que se parecem muito com consultas SQL.

O Assistente de consulta visual (*low-code*) não dá suporte para os recursos de `UNION`, `INTERSECT` e `EXCEPT`, porém, para consultas mais complexas que necessitam dessas funcionalidades, utilize a opção **Editar como Texto** (modo *high-code*, destaque 13 da Figura 2). Acesse o tópico **UNION** na documentação oficial para mais detalhes.

A Figura 3 mostra um exemplo de consulta JPQL. O Assistente visual foi acessado a partir do bloco de programação Abrir Consulta. Importante destacar que no campo **Entidade** escolhemos uma entidade do tipo classe, em vez de uma Fonte de dados. Por isso, todos os campos da imagem são exibidos. Ao escolher uma **Entidade** do tipo Fonte de dados, uma quantidade menor de campos será exibida. Veja mais detalhes no tópico REST.

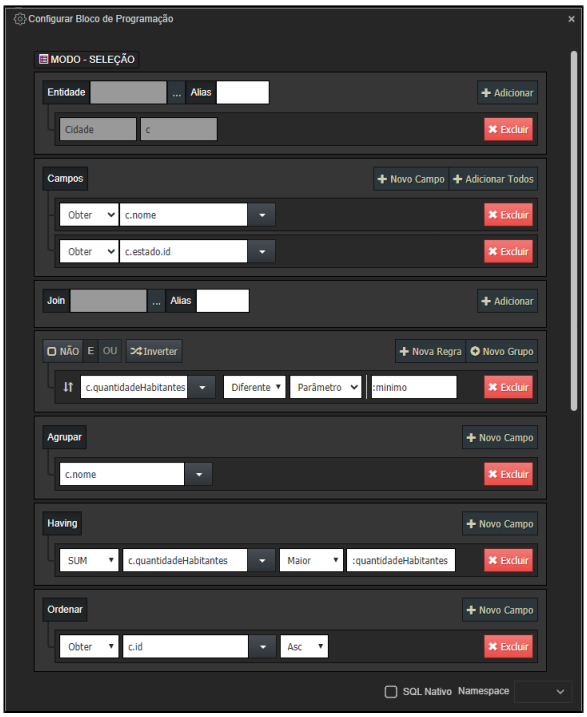

Figura 3 - Exemplo de consulta JPQL

Consulta JPQL gerada pela configuração do assistente visual da Figura 3:

```
SELECT          c.nome,c.estado.id FROM Cidade c
WHERE                c.quantidadeHabitantes <> :minimo
GROUP BY        c.nome
HAVING                  SUM(c.quantidadeHabitantes) > :quantidadeHabitantes
ORDER BY         c.id ASC
```

A Figura 3.1 exibe uma consulta do tipo SQL Nativo que possui o mesmo resultado da consulta JPQL da Figura 3.

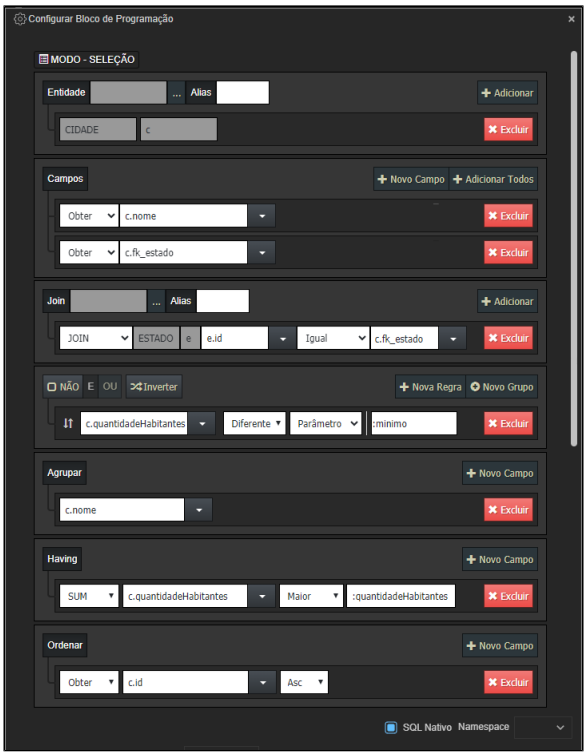

Figura 3.1 - Exemplo de consulta SQL Nativo

Exemplo da consulta em SQL:

```
SELECT      c.nome,c.fk_estado FROM CIDADE c
JOIN                ESTADO e on (e.id= c.fk_estado)
WHERE       c.quantidadehabitantes <> :minimo
GROUP BY    c.nome
HAVING      SUM(c.quantidadehabitantes) > :quantidadeHabitantes
ORDER BY    c.id ASC
```

O JPQL permite executar algumas funções do seu banco de dados com o uso da função `SQL('')`. Porém, esse recurso possui algumas limitações e pode variar a depender do banco de dados utilizado, sendo assim, a sua utilização necessitará de conhecimento do próprio JPQL e do Banco de dados utilizado. Acesse a documentação do JPQL para mais detalhes sobre essa e outras funções do JPQL.

Para incluir na consulta, selecione o modo de edição *High-code* (botão **Editar como texto** (destaque 13 da Figura 2)) no **Assistente de Consulta** do Cronapp.

Por exemplo, a consulta abaixo retorna o horário atual do banco de dados ao executar em um banco **My SQL**.

Exemplo para um banco MySQL

```
select SQL('NOW()') from User u
```

Utilize as consultas em SQL nativo dos blocos de programação da categoria **banco de dados** e na Fonte de dados do tipo SQL Nativo, caso não queira trabalhar com consultas JPQL.

REST

A imagem abaixo exibe o Assistente visual com uma Fonte de dados selecionada no campo Entidade a partir do bloco Abrir Consulta. Observe que apenas quatro campos de configuração são exibidos. E um deles, o campo Filtro, é exibido apenas em consultas do tipo Fonte de dados.

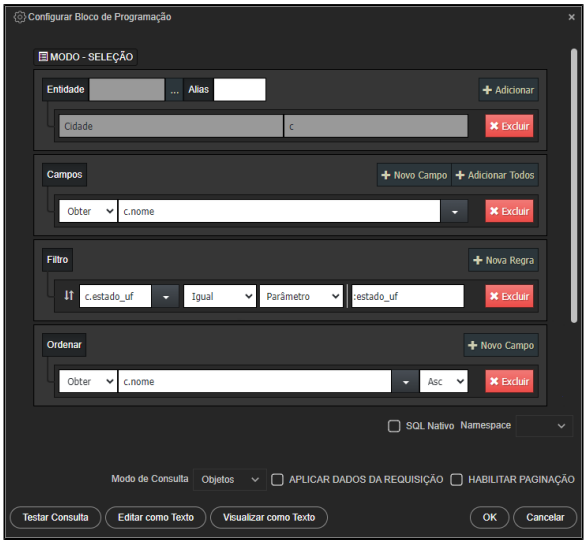

Figura 3.2 - Exemplo de consulta visual com entidade do tipo Fonte de dados

Além da diferença nos campos do Assistente visual, a consulta gerada a partir de uma Fonte de dados gera uma requisição REST:

```
GET app/query187311?$select=nome&$filter=%28estado_uf%20eq%20%3Aestado_uf%
29&$orderby=nome%20ASC
```