Fonte de Dados

A Fonte de Dados é um recurso disponível no lado servidor dos projetos Cronapp e possibilita a comunicação bidirecional entre uma origem de dados, como um banco de dados, e diversas ferramentas no Cronapp, como páginas ou relatórios. Essa comunicação é realizada através do protocolo OData e possui diversos recursos como filtros, tratamento de dados, eventos, tratamento de segurança, geração de recurso REST e a possibilidade de reutilização da mesma Fonte de Dados em várias ferramentas do Cronapp. Esses aspectos garantem a interoperabilidade e integração entre as diferentes camadas do sistema.

Figura 1 - Fluxo de comunicação da Fonte de dados

A **Fonte de dados** e o componente visual fonte de dados, utilizado no Editor de views, são ferramentas diferentes.

A **Fonte de dados** é um recurso criado no lado servidor do projeto e serve a todos os recursos que necessitam de uma Fonte de dados, já o **componente visual fonte de dados** é utilizado apenas nas páginas web e telas mobile (lado cliente) e tem o objetivo de referenciar uma Fonte de dados, fazendo a comunicação entre os demais componentes visuais (exemplo Grade) e a Fonte de dados referenciada.

Tipos de Fontes de dados

O Cronapp possui algumas variações (tipos) de Fonte de dados, essas variações fazem referências a origem dos dados e a possibilidade de personalização dos dados obtidos.

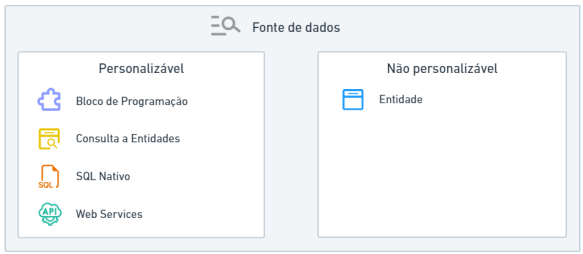

Figura 2 - Tipos de Fonte de dados

Personalizável

Para utilizar um dos tipos abaixo, primeiro será necessário criar e configurar uma Fonte de dados.

- **Bloco de Programação (servidor)**: configure um bloco de programação servidor para retornar uma lista de objetos.
- **Consulta a Entidades**: selecione uma classe do sistema e faça filtros a partir de uma consulta JPQL com o Assistente de consulta.
- **SQL Nativo**: selecione um diagrama de dados (namespace) do sistema e faça filtros a partir de consultas SQL nativa, utilizando o Assistente de consulta ou declarações SQL.
- **Web Services**: utilize um *endpoint* REST ou um endereço SOAP (`*.wsdl`).

Não personalizável

Para utilizar o tipo abaixo, basta selecioná-la a partir de alguma ferramenta como Editor de view ou Relatório.

Nesta página

Assista sobre o tema no Cronapp Academy

Caso seja seu primeiro acesso ao Cronapp Academy, crie antes uma conta gratuita e matricule-se no curso abaixo.

- Aula: Criar fonte de dados

- **Entidade**: retornará todos os registros de uma entidade, sem a possibilidade de realizar filtros ou personalização no lado servidor, apenas na requisição do lado cliente. Veja mais detalhes no tópico "Filtros e Parâmetros" da documentação Componente visual fonte de dados.

Acesso à lista

É possível acessar a lista das Fontes de dados do projeto a partir da árvore de recursos, no diretório **Fontes de Dados** (Localização: `/Fontes de Dados/`) (destaque 2 na árvore de recursos), ou através da janela **Buscar Fonte de Dados**. Para abrir a janela, acesse no menu do sistema: **Projeto > Fonte de dados**.

O diretório **Fontes de Dados** (destaque 2 na árvore de recursos) permite criar subdiretórios para organizar as Fontes de dados do sistema (3). Ao abrir a janela **Buscar Fonte de Dados**, a lista estará agrupada pelo tipo da Fonte de dados (Bloco de programação, Consulta a Entidades, SQL Nativo ou Web Services) e por diretório (3).

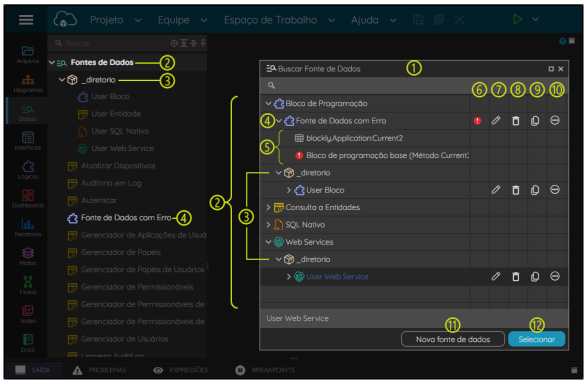

Figura 3 - Acesso a lista de Fontes de dados

1. **Janela Buscar Fonte de Dados**: lista as Fontes de dados do projeto e permite criar, editar, duplicar ou excluir.
2. **Diretório Fontes de Dados**: exibe a lista das Fontes de dados do sistema.
    - A **árvore de recurso** lista as Fontes de dados de forma hierárquica, exibindo diretórios e subdiretórios.
    - A **janela de seleção** exibe as Fontes de dados de forma agrupada, separando pelo tipo Fonte de dados (Bloco de programação, Consulta a Entidades, SQL Nativo ou Web Services) e depois por diretórios.
3. **Subdiretório**: utilize subdiretórios para organizar as Fontes de dados de acordo as necessidades do projeto.
4. **Fonte de dados**: dê um clique duplo para abrir a janela de configuração da Fonte de dados.
    - A **árvore de recurso** mostra a Fonte de dados com o ícone do seu tipo e nome.
    - A **janela de seleção** mostra o nome e permite expandir a Fonte de dados para exibir mais informações (5).
5. **Informações**: ao expandir a Fonte de dados na janela, é possível visualizar o que alimenta a fonte de dados, filtros e informações de erros na configuração da Fonte de dados.
6. **Erro**: quando uma Fonte de dados possui algum erro de configuração, um ícone de alerta é exibido. Posicione o cursor do mouse sobre o ícone para exibir um *tooltip* com informações do problema.
7. **Editar**: abre a janela de edição da Fonte de dados.
8. **Excluir**: apaga a Fonte de dados selecionada.
9. **Duplicar**: faz uma cópia da Fonte de dados selecionada.
10. **Opções/Ações**: abre um menu com opções para gerar recursos a partir da Fonte de dados. Essas opções também estão disponíveis no item "Ações" no menu de contexto da Fonte de dados, veja mais detalhes no tópico Drag and Drop a ações da Fonte de dados.
11. **Botão Nova Fonte de dados**: abre a janela para configurar uma nova Fonte de dados.
12. **Botão Selecionar**: abre a janela de configuração da Fonte de dados selecionada.

Acesse o tópico Drag and Drop a ações da Fonte de dados para conhecer outras funções geradas pelos arquivos da Fonte de dados.

Criar e Configurar

É possível criar uma Fonte de dados por vários caminhos:

1. Na estrutura de arquivos, posicione o mouse sobre o diretório **Fontes de Dados** (destaque 2 da figura 3) (Localização: `/Fontes de Dados/`), clique no ícone **(+)** ao lado e selecione **Font e de dados**.
2. Na estrutura de arquivos, abra o menu de contexto do diretório **Fontes de Dados** (destaque 2 da figura 3) (Localização: `/Fontes de Dados/`) e acesse **Novo > Fonte de dados**.
3. Na janela **Buscar Fonte de Dados**, clique no botão **Nova Fonte de dados** (10 da figura 3).
4. Nas ferramentas que utilizam a Fonte de dados, como Editor de views ou Dashboard, ao abrir a janela de seleção da Fonte de dados, é possível encontrar o botão **Nova Fonte de dados**.

A janela de edição possui diversos recursos para personalizar a Fonte de dados. A configuração mais básica necessita apenas dos campos: **Nome da consulta** (destaque 5 da Figura 4), **Tipo da Fonte de dados** (6) e **Origem dos dados** (7).

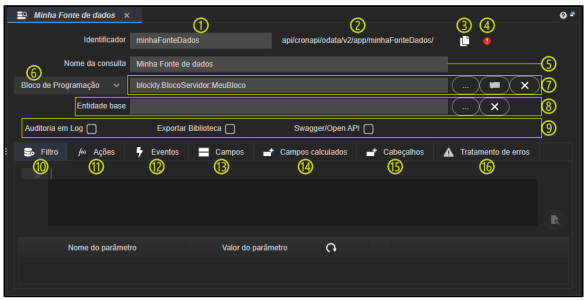

Figura 4 - Janela de edição da Fonte de dados

1. **Identificador**: identificador único da Fonte de dados no projeto, não aceita espaços ou caracteres especiais.
    - A Fonte de dados sempre é criada com um identificador padrão único e imutável (ex.: "query123456"), ao modificar esse campo, um segundo identificador é criado e a Fonte de dados passará a atender ambas requisições, isso possibilita gerar endereços REST (2) personalizados. Internamente, o Cronapp sempre utilizará o identificador padrão.
2. **Endereço REST**: independentemente do tipo da Fonte de dados, ela sempre irá gerar um endereço REST (URN). Esse recurso atenderá aos principais verbos HTTP (aba Ações) e pode ser utilizado dentro do projeto ou fora, com um token de autenticação (X-AUTH-TOKEN) ou habilitando as permissões da Fonte de dados e seus atributos (abas Ações e Campos).
3. **Copiar URL**: copia o URN da Fonte de dados.
4. **Sinalização de Erro**: ao detectar erro de configuração, um ícone de alerta é exibido. Posicione o cursor do mouse sobre o ícone para exibir um *tooltip* com informações do problema.
5. **Nome da consulta**: nome amigável da consulta exibido na janela de seleção das Fontes de dados, aceita espaços e alguns caracteres especiais.
6. **Tipo da fonte**: define o tipo da Origem dos dados (7), podendo ser Entidade (classe do namespace), SQL Nativo (consulta ao Banco de dados), Bloco de programação (função Java) ou Web services (REST ou SOAP).
7. **Origem dos dados**: o campo sempre exibirá o que foi selecionado, já os botões, vão variar a depender do que for escolhido no campo "Tipo da fonte" (6).
    - **Consulta a Entidades**: exibe um botão para selecionar uma das classes de qualquer Diagrama de dados (namespace) do projeto.
    - **Bloco de programação**: exibirá botões para selecionar uma das funções de bloco servidor ou criar um novo bloco de programação. A função deve retornar uma lista de objeto JSON.
    - **Web Service**: abre uma janela que permite configurar um endereço REST ou SOAP. Veja mais detalhes em Fonte de dados tipo Web Service (REST / SOAP).
    - **SQL Nativo**: permite selecionar um dos namespaces (Diagrama de dados) do projeto.
8. **Entidade base**: esse campo só é habilitado para as Fontes do tipo Bloco de programação ou *web service* e é usado para definir os campos do objeto com base em uma das classes do Diagrama de dados (namespace).
9. **Opções:** conjunto de caixas de seleção, acesse a documentação oficial de cada uma para mais detalhes.
    - **Auditoria em Log**: habilita o sistema de auditoria para essa Fonte de dados.
    - **Exportar biblioteca**: inclui a Fonte de dados selecionada como um recurso ao exportar uma biblioteca do Cronapp.
    - **Swagger/Open API**: inclui o *endpoint* gerado pela Fonte de dados selecionada na lista de recursos expostos no Swagger/Open API.
10. **Aba Filtro**: define uma consulta JPQL ou SQL e permite tratar os filtros e parâmetros criados nas configurações da Fonte de dados.
11. **Aba Ações**: define permissões para os verbos HTTP.
12. **Aba Eventos**: chama ações em diversos momentos da execução da Fonte de dados.

13. **Aba Campos**: configura os atributos (campos) dos objetos e dá permissões diferenciadas a partir dos permissionáveis.
14. **Aba Campos calculados**: adiciona novos campos ao mapa da Fonte de dados.
15. **Aba Cabeçalhos**: adiciona cabeçalho em Fonte de dados via Web Services.
16. **Aba Tratamento de erros**: adiciona mensagens de erros personalizadas, permite internacionalização dessas mensagens.

Aba Filtro

Essa aba permite definir as consultas JPQL e SQL para as Fontes de dados Consulta a Entidades e SQL Nativo, respectivamente. Já os filtros/parâmetros (destaque 3 da figura 4.1) possuem diferenças a depender do tipo de Fonte de dados:

- **Consulta a Entidades**: os filtros são gerados a partir da cláusula `where` da consulta JPQL (destaque 1 da figura 4.1) ou ao utilizar a expressão "**`:<parâmetro>`**" (dois pontos + nome do parâmetro) em outras partes da consulta.
- **Bloco de programação**: os filtros são criados a partir dos parâmetros da função de bloco de programação vinculada.
- **Web Services**:
  - **REST**: os filtros são definidos através das configurações dos parâmetros na aba Ações . Cada verbo da aba Ações (Para obter, Para inserir...) permite configurar novos parâmetros e escolher como serão alimentados, caso escolha a opção "Parâmetro" na coluna **Valor do campo** e informe um nome, esse parâmetro será exibido na lista de parâmetros da aba Filtro (3). Veja mais detalhes da configuração de parâmetros no tópico Parâmetros e Constantes.
    Também é possível definir um parâmetro no endPoint do recurso e gerar rotas diferentes com base em algum atributo. Ao utilizar a expressão "**`:<parâmetro>`**" (dois pontos + nome do parâmetro) no endereço do endPoint (ex: `http://www.ws.com/usuario/`**`:idusuario`**`/dados`), um filtro ("`idusuario`") será criado na lista de parâmetros da aba Filtro (3).
  - **SOAP**: os filtros são definidos através dos parâmetros dos métodos selecionados na aba Ações. Cada verbo dessa aba (Para obter, Para inserir...) permite selecionar um dos métodos presentes no recurso SOAP, caso o método escolhido possua parâmetro e na configuração desse parâmetro seja escolhido a opção "Parâmetro" na coluna **Valor do campo** e definido um nome, esse nome será exibido na lista de parâmetros da aba Filtro (3). Veja mais detalhes da configuração de parâmetros no tópico Parâmetros e Constantes.
- **SQL Nativo**: os filtros são gerados a partir da cláusula `where` da consulta SQL (destaque 1 da figura 4.1) ou ao utilizar a expressão "**`:<parâmetro>`**" (dois pontos + nome do parâmetro) em outras partes da consulta.

No exemplo da Figura 4.1, a Fonte de dados retornará todos os usuários que possuem e-mails com algum valor definido no **Valor do parâmetro** (4), obtido a partir da função "ServidorEmail" do bloco de programação.

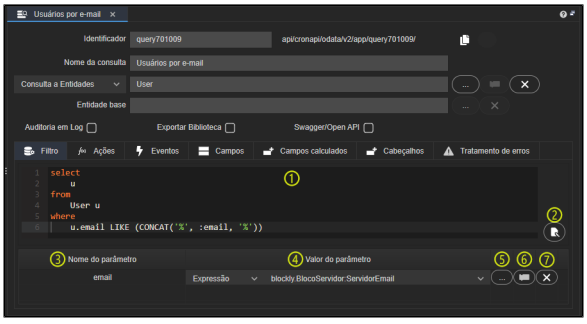

Figura 4.1 - Exemplo de filtro para uma entidade

1. **Consulta**: exibe a consulta JPQL ou SQL, esse campo estará desabilitado para as Fontes de dados Bloco de programação e Web Service.
2. **Editar**: abre a janela do Assistente JPQL e SQL, esse botão estará desabilitado em Fontes de dados Bloco de programação e Web Service.
3. **Nome do parâmetro**: nome do filtro/parâmetro.
4. **Valor do Parâmetro**: define um valor estático ou dinâmico para o filtro. Acesse o tópico Parâmetros e Constantes para mais detalhes sobre valores dinâmicos.
5. **"..."**: seleciona uma função de bloco de programação servidor para alimentar o filtro.
6. **Novo bloco**: cria função de bloco de programação servidor para alimentar o filtro.

7. **Limpar**: apaga a configuração do filtro.

Aba Ações

Na aba Ações é possível definir o que será executado dos verbos HTTP (get, post, put, delete) e das ações filtrar e contar, além dos permissionáveis que terão acesso. A forma como a ação será executada vai depender do tipo da Fonte de dados.

Paginação

O verbo **Para Contar** é usado internamente para paginar os registros retornados, porém, para funcionar, as ações **Para Obter** e **Para Contar** devem possuir as mesmas configurações.

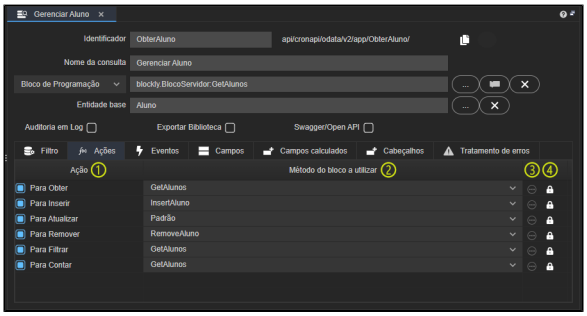

Figura 4.2 - Aba Ações da Fonte de dados

1. **Coluna Ações**: nome da ação com um checkbox para ativar ou desativar cada ação (verbos HTTP).
2. **Coluna opção da ação**: essa coluna terá um nome e um comportamento diferente a depender do tipo da Fonte de dados:
    - **Consulta a Entidades**: (coluna desabilitada) possui um padrão próprio para executar os verbos e não é possível personalizá-los.
    - **Bloco de Programação**: ("Método do bloco a utilizar") permite selecionar funções existentes no arquivo do bloco de programação (blockly) que alimenta a Fonte de dados, essas funções podem ser selecionadas para substituir a execução padrão de cada ação.
      Com exceção das ações "Para Obter" e "Para Contar", as demais ações não permitem configurar o que será passado via parâmetro para as funções selecionadas, a lógica das funções devem se adaptar a esse formato:
        - **Para Obter**: essa ação é a que alimenta a Fonte de dados e é a única que permite tratar os parâmetros, ao selecionar uma função com parâmetros, esses devem ser configurados na aba Filtro.
          Essa função também será responsável por tratar ordenação e paginação, para isso, utilize o bloco Obter parâmetro da query string para obter os parâmetros de query String da Fonte de dados (ex.: "`$orderby`"). A função deve retornar uma lista de objetos.
        - **Para Inserir**: enviará por parâmetro um objeto JSON contendo todos os atributos, inclusive o ID.
        - **Para Atualizar**: enviará por parâmetro um objeto JSON contendo todos os atributos.
        - **Para Remover**: enviará por parâmetro um objeto JSON contendo apenas o atributo ID (ex.: `{"id":"123456"}`).
        - **Para Filtrar**: ação executada ao utilizar o campo de pesquisa da página CRUD ou os campos de filtro nas colunas da Grade. Utilize o bloco Obter parâmetro da query string para obter os parâmetros de query String da Fonte de dados (ex.: "`$filter`"). A função deve retornar uma lista de objetos.
        - **Para Contar**: essa ação é usada internamente para paginar os registros retornados e deve possuir as mesmas configurações da ação "Para Obter".
    - **Web Services**:
        - **REST**: ("Função") é possível definir um *endpoint* específico para cada ação e adicionar novos parâmetros para essa requisição (ver mais detalhes no tópico "Aba Ações" da documentação Fonte de dados tipo Web Service).
        - **SOAP**: ("Caminho") é possível selecionar qualquer um dos métodos do serviço SOAP e definir os parâmetros desse método (ver mais detalhes no tópico "Aba Ações" da documentação Fonte de dados tipo Web Service).
    - **SQL Nativo**: (coluna desabilitada) possui um padrão próprio para executar os verbos e não é possível personalizá-los.

3. "**...**": esse botão ficará habilitado apenas em Fontes de dados do tipo Web Services e abre uma janela para personalizar os parâmetros do método SOAP ou *endpoint* REST em cada ação (ver mais detalhes no tópico "Aba Ações" da documentação Fonte de dados tipo Web Service).
4. **Permissão**: define quais permissionáveis podem executar a ação configurada.

   As permissões aplicadas na aba **Ações** serão herdadas para todos os atributos nas abas **Campos** e **Campos calculados**. Por exemplo, ao selecionar a permissão "Administradores" do verbo **Para Obter** na aba Ações, todos os atributos terão os seus campos **Permitir Obter** (destaques 11 da figura 4.4 e 9 da figura 4.5) configurados automaticamente para "Administradores".

Outras informações sobre o tipo Bloco de programação

Se o campo "Entidade base" (destaque 8 da figura 4) estiver configurado, será possível selecionar a opção "Padrão" na caixa de seleção de cada ação. Essa opção faz com que a ação seja executada da mesma forma como ocorre nas Fontes de dados do tipo Consulta a Entidades. Assim, a lista de dados pode ser obtida via bloco de programação, mas as demais ações (inserir, atualizar e remover) serão executadas no banco de dados através da classe selecionada no campo "Entidade base".

Acesse a documentação Tratar recurso REST de terceiros no Cronapp e veja exemplos de como tratar recursos de paginação, ordenação e filtro com Fonte de dados do tipo Bloco de programação.

Aba Eventos

A aba Eventos permite rodar alguma função antes ou após a Fonte de dados executar uma ação. Os eventos "Antes" costumam ser muito úteis, pois, caso uma validação não seja atendida, uma exceção pode ser lançada.

Existem diferenças entre os eventos da Fonte de dados (Figura 4.3) e os eventos do componente visual fonte de dados. O componente visual trata os dados do lado cliente (JavaScript) e seus eventos não podem ser interrompidos, já os Eventos da Fonte de dados (Java) podem ser paralisados a partir de exceções em seu processo. Essas exceções são eventos que quebram o fluxo normal das instruções, assim, ao lançar uma exceção, uma mensagem será exibida na saída do console e uma notificação de erro na aplicação (veja mais detalhes sobre exceções em Validação da fonte de dados usando Eventos).

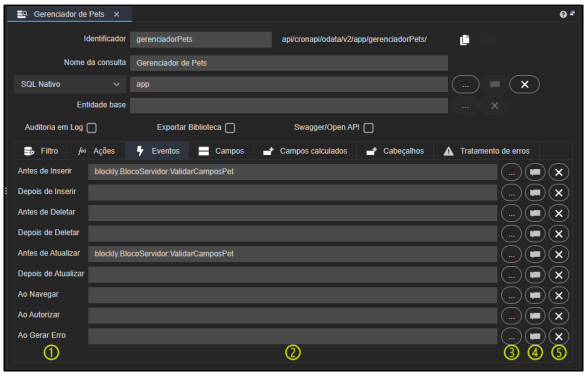

Figura 4.3 - Aba Eventos da Fonte de dados

Colunas

1. **Eventos**: lista de eventos.
2. **Campo**: exibe a função de bloco de programação selecionada.
3. **"..."**: abre a janela para seleção da função.
4. **Novo bloco**: cria função de bloco de programação servidor para alimentar o evento.
5. **Limpar**: exclui os dados do campo.

Eventos

- **Antes de Inserir**: realiza o evento antes de **inserir** algum registro.
- **Depois de Inserir**: realiza o evento após **inserir** algum registro.
- **Antes de Deletar**: realiza o evento antes de **deletar** algum registro.
- **Depois de Deletar**: realiza o evento após **deletar** algum registro.
- **Antes de Atualizar**: realiza o evento antes de **atualizar** algum registro.
- **Depois de Atualizar**: realiza o evento após **atualizar** algum registro.
- **Ao navegar**: executa a ação ao abrir e ao navegar entre os elementos da Fonte de dados.
- **Ao Autorizar**: faz uma validação, se o evento lançar uma exceção, ele será bloqueado e o erro será exibido. Isso evitará, por exemplo, que um usuário possa pegar o id para montar e acessar uma requisição sem permissão. O evento será executado para todas as chamadas GET, PUT,

DELETE, POST da Fonte de dados. Esse evento costuma ser muito útil em sistemas multitenant.
- **Ao Gerar Erro:** tratamento de erros de qualquer natureza durante o processo de manipulação de dados de uma Fonte de Dados.

Aba Campos

A aba Campos é responsável por mapear os objetos manipulados pela Fonte de dados, permitindo configurar cada atributo individualmente. Algumas características dessa aba mudam a depender do tipo de Fonte de dados:

- **Consulta a Entidades**: apenas o tipo Consulta a Entidades não permite adicionar ou remover campos, pois, como é baseada em classes, exibirá todos os atributos contidos na classe selecionada. Pelo mesmo motivo não estão disponíveis ou habilitados os campos de configuração 6, 7, 8, 10 e 13 da Figura 4.4, já que essas características foram definidas nas classes do Diagrama de dados.
- **Bloco de programação**: é possível selecionar uma classe no campo "Entidade base" (destaque 8 da figura 4) para preencher todos os seus campos automaticamente. O campo de configuração 10 da figura 4.4 não estará habilitado.
- **Web Services**: após vincular um *endpoint* à Fonte de dados, todos os campos do objeto da API serão automaticamente incluídos nessa aba. No entanto, é possível selecionar uma classe no campo "Entidade base" (destaque 8 da figura 4) e substituir todos os atributos pelos da classe selecionada. Isso pode ser útil caso o objeto da API não represente a classe do projeto, utilize a opção "Caminho" (destaque 10 da figura 4.4) para informar o *JSON Path* de cada atributo do objeto da API.
- **SQL Nativo**: após definir a consulta SQL na aba Filtro, todos os campos da cláusula `select` serão automaticamente incluídos, mas também é possível selecionar uma classe no campo "Entidade base" (destaque 8 da figura 4) para adicionar todos os seus atributos automaticamente.

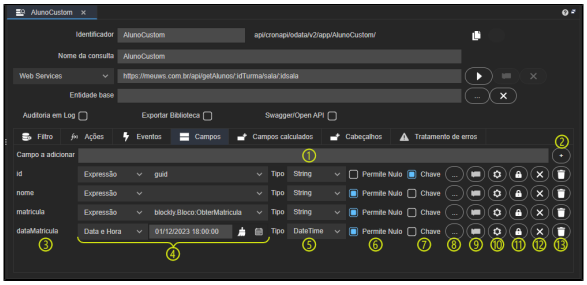

Figura 4.4 - Aba campos e as várias configurações de seus atributos

1. **Campo a adicionar**: nome do novo atributo.
2. **Adicionar campo**: insere o atributo informado (1).
3. **Nome dos campos**: coluna com o nome dos atributos.
4. **Valor do campo**: define um valor estático ou dinâmico para alimentar o atributo. Acesse o tópico Parâmetros e Constantes para mais detalhes das configurações dos valores dinâmicos (Expressão).
5. **Tipo do campo**: define o tipo do atributo.
6. **Permite nulo**: define se o campo aceitará valores nulos.
7. **Chave**: define se o campo é uma das chaves primárias da entidade.
8. **"..."**: selecione uma função bloco de programação servidor existente para alimentar o campo. Para passar um atributo pelo parâmetro da função, utilize a expressão `data.<nome_atributo>` ou `data` para enviar o objeto completo.
9. **Bloco**: cria uma função de bloco de programação para alimentar o campo.
10. **Caminho**: disponível apenas para Fontes de dados do tipo Web Services e SQL Nativo.
    - **Web Service**: permite informa o caminho (*JSON Path*) do atributo que irá alimentar o campo (ex,: `$.data[0].id`).
    - **SQL Nativo**: permite acessar as expressões usadas nos campos, como *subqueries* ou "`CASE WHEN`", caso o campo não possua expressão, exibirá apenas o nome do campo no formato: `<tabela ou alias>.<nome_campo>`. (Veja mais detalhes na documentação Fonte de dados tipo SQL Nativo)
11. **Permissões**: define restrições para cada verbo aos permissionáveis cadastrados no sistema.

    As configurações das permissões feitas na aba Ações serão herdadas automaticamente para cada atributo. Após a herança de permissões, ainda será possível ajustar as permissões individualmente em cada atributo.

12. **Limpar**: apaga as configurações feitas no atributo.

13. **Excluir**: exclui o atributo.

Aba Campos calculados

A aba Campos calculados permite incluir campos temporários nos objetos manipulados pela Fonte de dados. Diferentemente da aba Campos, aqui os atributos e seus valores não são persistidos, existem apenas durante o tempo de execução, na memória na aplicação.

Os campos calculados não suportam filtros, pois as requisições de filtros são encaminhadas ao banco de dados ou a outro recurso que alimenta a Fonte de dados e lá, os campos calculados não existem.

Os campos calculados podem não funcionar em recursos que rodem internamente no Cronapp (como é o caso da aba **Pré-visualizar** do Relatório e Dashboard), sendo exibidos somente ao executar a aplicação.

Exemplo de uso: Em uma Fonte de dados vinculada a entidade usuários, podemos criar um campo calculado `idadeAtual`, ela chamará uma função Servidor que obtém o campo `dataNascimento` do usuário e retornará a sua idade baseado na data atual. Dessa forma, sempre que precisar exibir a idade atual dos usuários, basta chamar o campo calculado `idadeAtual` da Fonte de dados.

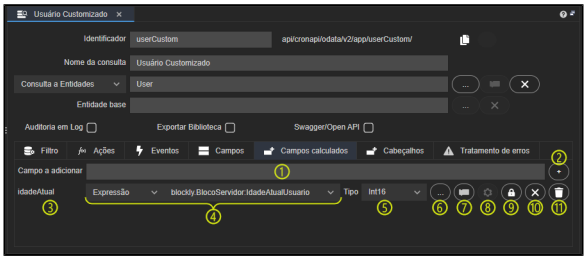

Figura 4.5 - Aba de campos calculados

1. **Campo a adicionar**: nome do novo atributo.
2. **Adicionar campo**: insere o atributo informado (1).
3. **Nome dos campos**: coluna com o nome dos campos calculados.
4. **Valor do campo**: define um valor estático ou dinâmico para alimentar o campo. Acesse o tópico Parâmetros e Constantes para mais detalhes sobre valores dinâmicos (Expressão).
5. **Tipo**: define o tipo do atributo.
6. **"..."**: selecione uma função bloco de programação servidor existente para alimentar o campo. Para passar um atributo pelo parâmetro da função, utilize a expressão `data.<nome_atributo>` ou `data` para enviar o objeto completo.
7. **Bloco**: cria uma função de bloco de programação para alimentar o campo.
8. **Caminho**: opção não disponível para Campos calculados.
9. **Permissões**: define restrições para cada verbo aos permissionáveis cadastrados no sistema.

   As configurações das permissões feitas na aba Ações serão herdadas automaticamente para cada atributo de campo calculado. Após a herança de permissões, ainda será possível ajustar as permissões individualmente em cada campo.
10. **Limpar**: apaga as configurações feitas no atributo.
11. **Excluir**: exclui o atributo.

Entidades relacionadas

Os campos calculados permitem referenciar atributos da própria entidade ou os atributos de outras entidades que possuam relacionamento 1 para N. Podemos utilizar esse recurso em diversas situações, por exemplo, em um relacionamento entre as entidades Cidade, Estado e País, no qual a Fonte de dados da entidade Cidade pode ter acesso a qualquer atributo das entidades Estado ou País. Na Figura 4.5.1, a Fonte de dados tem acesso ao atributo "região" da entidade Estado e ao atributo "nome" da entidade País. O termo `this` representa a entidade atual.

Como não existem limites de relacionamentos que podem ser feitos em um banco de dados, o Cronapp limita a lista de atributos (via caixa de seleção) até o terceiro relacionamento. Porém, é possível inserir manualmente relacionamentos maiores - Exemplo: `this.estado.pais.continente.planeta.distanciaSolemKm` .

Além da Fonte de dados do tipo Consulta a Entidades, esse recurso também estará disponível para os tipos Bloco de programação e Web Services, desde que tenha uma Entidade vinculada no campo **Entidade base** (destaque 8 da figura 4).

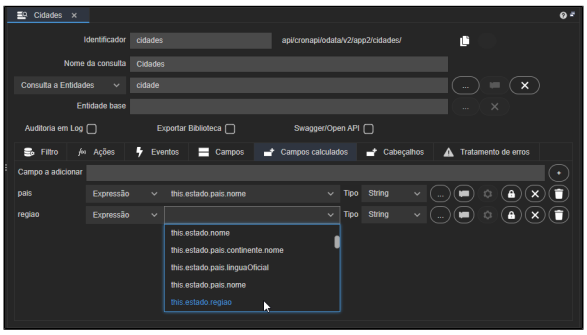

Figura 4.5.1 - Campos calculados referenciando atributos de outras entidades relacionadas

Aba Cabeçalhos

Essa aba é utilizada apenas para as Fontes de dados do tipo Web services e permite configurar os cabeçalhos das requisições HTTP do endPoint configurado. Por padrão, os campos Accept e Content-Type já vem definidos ao criar uma Fonte do tipo Web Services, porém é possível alterar e adicionar novos.

Os cabeçalhos da janela "Obter campos do serviço" (destaque a da figura 4.6) não aceitam valores dinâmicos (função de blocos de programação ou expressão), dessa forma, ao configurar um cabeçalho com valor dinâmico na aba Cabeçalhos e depois acessar a janela "Obter campos do serviço" (destaque a), o cabeçalho estará vazio e será necessário informar o valor estático. Veja mais detalhes em Fonte de dados tipo Web Service (REST / SOAP).

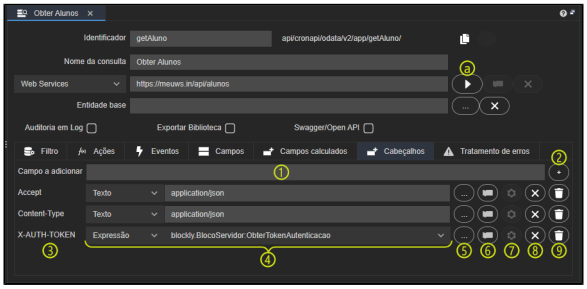

Figura 4.6 - Aba Cabeçalhos de uma Fonte de dados Web Services

1. **Campo a adicionar**: nome do novo cabeçalho.
2. **Adicionar campo**: insere o cabeçalho informado (1).
3. **Nome**: coluna com o nome dos cabeçalhos.
4. **Valor do cabeçalho**: define um valor estático ou dinâmico para alimentar o cabeçalho. Acesse o tópico Parâmetros e Constantes para mais detalhes sobre valores dinâmicos (Expressão).
5. **"..."**: selecione uma função bloco de programação servidor existente para alimentar o cabeçalho.
6. **Bloco**: crie uma função de bloco de programação para alimentar o cabeçalho.
7. **Caminho**: opção não disponível para cabeçalhos.
8. **Limpar**: apaga as configurações feitas no cabeçalho.
9. **Excluir**: exclui o cabeçalho.

Aba Tratamento de erros

Essa aba é utilizada para exibir uma mensagem amigável para o usuário caso ocorra algum erro ou a Fonte de dados não encontre chaves primárias e estrangeiras dos objetos manipulados. Essas mensagens podem ser internacionalizadas.

Esse recurso ainda não está disponível para as Fontes de dados do tipo SQL Nativo.

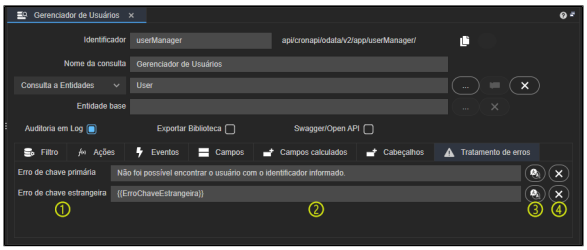

Figura 4.7- Aba tratamento de erro

Colunas

1. **Campos**: lista de opções para tratamento de erros.
2. **Valor**: aceita textos estáticos ou chave de internacionalização.
3. **Internacionalização**: Abre a janela de internacionalização da mensagem.
4. **Limpar**: exclui o conteúdo preenchido no campo.

Opções de tratamento

- **Erro de chave primária**: mensagem informada ao usuário quando não for possível encontrar algum item (*primary key*).
- **Erro de chave estrangeira**: mensagem informada ao usuário quando não for possível encontrar algum item de uma entidade relacionada (*foreign key*).

Outras informações

Neste tópico veremos mais detalhes da Fonte de dados.

Drag and Drop a ações da Fonte de dados

Os arquivos da **Fonte de dados** (Localização: `/Fonte de Dados/`) na árvore de recursos podem ser arrastados (drag and drop) até o Editor de views para gerar componentes visuais já vinculados com o co mponente visual fonte de dados. Ao arrastar, o menu de contexto **Criar Componente** exibirá as opções abaixo, variando se o Editor selecionado for web ou mobile:

- **Caixa de seleção dinâmica**:
  - **Web**: abre a janela de configuração da Caixa de seleção dinâmica (web).
  - **Mobile**: abre a janela de configuração da Caixa de seleção dinâmica (mobile).
- **Crud**:
  - **Web**: abre a janela do Assistente de view para entidade com as opções dos formulários Web.
  - **Mobile**: abre a janela do Assistente de view para entidade com as opções dos Formulários mobile.
- **Grade/Lista avançada**:
  - **Web**: abre a janela de configuração do componente Grade (web).
  - **Mobile**: abre a janela de configuração do componente Lista Avançada (mobile).

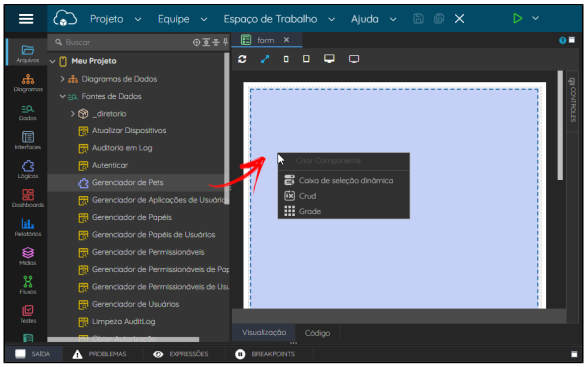

Figura 5 - Adicionando componentes visuais no Editor de Formulários a partir da Fonte de dados

Outro recurso interessante gerado a partir dos arquivos da Fonte de dados é o item **Ação** do seu menu de contexto, ele exibe opções para gerar páginas CRUDs com o Assistente de CRUD ou gerar um relatório, com os atributos da Fonte de dados (veja mais detalhes no tópico "Criar relatórios" da documentação Relatório).

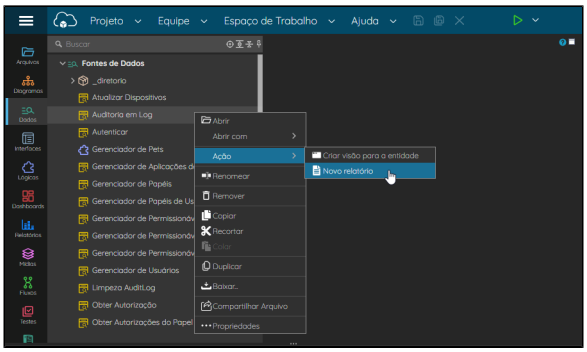

Figura 5.1 - Item Ação do menu de contexto dos arquivos da Fonte de dados

Parâmetros e Constantes

A janela de configurações de parâmetros (Figura 6) está disponível em diversos locais dentro do Cronapp e também nas abas Filtro, Ações, Eventos, Campos, Campos calculados e Cabeçalhos da Fonte de dados. Ela permite definir os parâmetros que serão passados e seus valores.

A depender do local em que essa janela for aberta, algumas características podem mudar. Por exemplo, a possibilidade de incluir novos parâmetros a partir dessa janela ou selecionar as opções "Bloco" ou "Parâmetro" na coluna **Valor do campo**. Abaixo vemos a janela de seleção de Parâmetros da Aba Ações da Fonte de dados do tipo Web Services, em que é possível personalizar os parâmetros (QueryString) passados por cada ação.

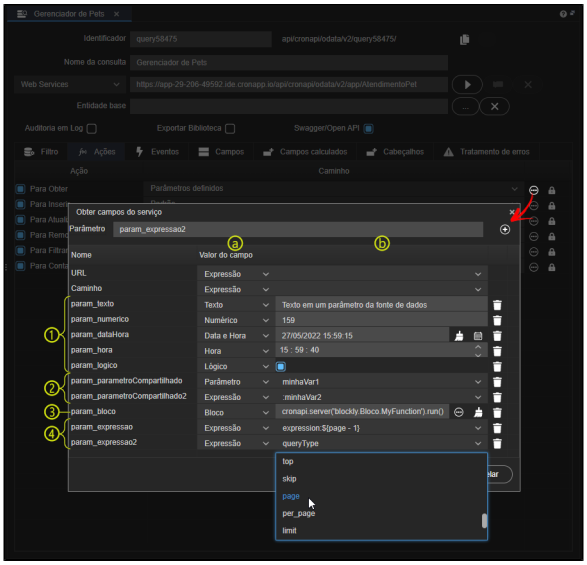

Figura 6 - Exemplos de tipos de parâmetros da janela de configuração das ações

Os 4 itens abaixo representam as opções da coluna **Valor do campo** (destaque a da figura 6) e o que muda nos campos da coluna **Conteúdo** (essa coluna não possui um nome, mas está destaca pela letra **b** na figura 6):

1. **Estáticos**: possui as seguintes opções: **Texto**, **Numérico**, **Data e Hora**, **Hora** ou **Lógico**.
    - **Coluna conteúdo**: (destaque b da figura 6) o campo muda de acordo com a opção escolhida. Exemplo, se a opção "Lógico" for selecionada, o campo ao lado exibirá um *c heckbox* para inserir as opções *true* ou *false*.
2. **Parâmetro Compartilhado**: disponível apenas nos campos da aba Ações das fontes de dados Web Service REST, essa opção permite criar uma variável que será alimentada a partir da aba Filtro e poderá ser utilizada nas configurações de todas as ações dessa Fonte de dados.
    - **Coluna conteúdo**: (destaque b da figura 6) o campo conteúdo exibirá um campo de texto para informar o nome da variável. É possível criar uma variável ao selecionar "Parâmetro" na coluna **Valor do campo** (a) e informar um nome no campo conteúdo (b) ou selecionar a opção "Expressão" e no campo conteúdo (b) informar um nome com o caractere dois-pontos na frente (ex,: "`:minhaVar`"). Os parâmetros destacados no item 2 da figura 6 possuirão o mesmo resultado.
3. **Bloco**: permite executar uma função de bloco de programação servidor com retorno. Caso o bloco possua parâmetro, é possível passar como argumento uma lista de constantes, uma expr essões FTL ou criar parâmetro compartilhando (2).
    - **Coluna conteúdo**: (destaque b da figura 6) exibe um campo onde é possível selecionar um bloco de programação Servidor.
4. **Expressão**: permite selecionar um item da lista de constantes ou criar expressões FTL.
    - **Coluna conteúdo**: (destaque b da figura 6) exibe uma caixa de seleção editável onde é possível selecionar/adicionar uma constante Cronapp ou informar uma expressão FTL.

Expressão

A opção **Expressão** na coluna **Valor do campo** (destaque a da figura 6) dará acesso a uma caixa de seleção editável onde é possível selecionar constantes Cronapp ou incluir uma expressões FTL (FreeMarker Template Language).

Com o uso de expressões FTL é possível incluir pequenos cálculos ou gerar pequenos templates diretamente no campo **Conteúdo** (destaque b da figura 6). Esse recurso pode ser bem útil no caso de parâmetros simples, porém, ao necessitar de algo um pouco mais complexo, recomendamos fortemente utilizar uma função de bloco de programação servidor.

Expressões FTL são criadas inserindo o termo "`expression:`" e os caracteres "`${ <conteúdo> }`" em cada expressão (destaque 4 da figura 6).

Exemplo de FTL
`expression: ${page - 1}`

O resultado da expressão contida no bloco acima será o valor da constante "page" subtraído de 1.

Fique atento aos pontos abaixo ao trabalhar com expressões FTL e constantes Cronapp:

- Nem todas as constantes Cronapp são aceitas dentro de expressões FTL;
- É necessário ficar atento ao tipo retornado das constantes Cronapp, com exceção de "page", que retorna um valor numérico, todas as demais retornam o tipo texto (string). Dessa forma, ao tentar fazer um cálculo com o uso de constantes, o valor pode não retornar como esperado. Veja no exemplo abaixo como fazer uma conversão.

Para fazer conversões simples dentro de expressões FTL, como converter tipo string para number, basta incluir "`?number`" e forçar a conversão.

Conversão de tipos no FTL

```
expression: ${skip?number + 2}
```

Se o retorno da constante "skip" no exemplo do bloco acima for "3" e **não for utilizado** o recurso "?number", o resultado final será "32" ao invés de "5", pois a expressão irá concatenar os 2 valores. Já ao utilizar o "?number", o resultado será "5", como esperado.

Clique aqui e veja mais detalhes sobre conversões FTL.

Constantes

As constantes Cronapp são selecionáveis nos campos da coluna **Valor do campo** (destaque a da figura 6) em várias janelas, como as de configuração dos parâmetros query String, filtros e parâmetros de funções, elas retornam o valor do que foi selecionado. Algumas dessas constantes também são aceitas dentro de expressões FTL (veja exemplos de uso no tópico Expressão).

Veja a lista de Constantes:

- **NULL**: retorna nulo.
- **queryType**: retorna o tipo da requisição OData (ex: select, count).
- **queryStatement**: requisição ODATA.
- **queryFilter**: filtro original passado na query OData (ex.: name eq 'José').
- **queryParameters**: mapa com chave/valor dos parâmetros da consulta OData (ex.: ao usar `:<chave>` no jpql ou parâmetros de entrada de bloco).
- **querySort**: campo de ordenação na consulta OData.
- **querySortOrder**: direção de ordenação (desc/asc) da consulta OData.
- **simpleFilter**: obtém o valor passado para um filtro simples OData (ex.: name eq 'José', o simpleFilter retorna 'José').
- **simpleFilterField**: obtém apenas o nome do campo de um filtro simples OData (ex.: name eq 'José' retorna 'name').
- **entityName**: retorna a entidade da consulta OData.
- **data**: objeto contendo os atributos do registro corrente. Ver mais detalhes no tópico abaixo.
- **data.{field}**: valor do atributo informado no registro corrente (ex.: `data.idade` para obter o valor de "idade").
- **formData**: objeto contendo os atributos do registro corrente. Ver mais detalhes no tópico abaixo .
- **formData.{field}**: valor do atributo informado no registro corrente (ex.: `data.idade` para obter o valor de "idade").
- **rawEntry**: mapa contendo todos os atributos da requisição. Ver mais detalhes no tópico abaixo.
- **rawEntry.{field}**: valor do atributo informado no mapa da requisição (ex.: `data.idade` para obter o valor de "idade").
- **primaryKey**: retorna o valor do *primaryKey*.
- **primaryKeys**: retorna uma lista com os *primaryKeys*.
- **token**: retorna o objeto do token de autenticação.
- **token.{key}**: valor do atributo do token (ex.: `token.name` para obter o valor do atributo "name" do token de autenticação).
- **SSOAccessToken**: retorna o token do provedor de acesso SSO utilizado.
- **session.{key}**: valor da variável do tipo de autenticação Sessão. Apesar de existir a constante, não aconselhamos utilizar o tipo de autenticação Sessão no Cronapp.
- **expression:{value}**: retorna uma expressão FTL criada. Exemplos:

    - Na expressão "`expression:${1 + 2}`" o valor retornado será "3".
- **username**: usuário logado.
- **roles**: retorna uma lista com os permissionáveis do sistema.
- **top**: (paginação) parâmetro `top` do OData (máximo de registros).
- **skip**: (paginação) parâmetro `skip` do OData (quantidade de registros ignorados).
- **page**: (paginação, único que retorna um valor inteiro) facilitador que traz o cálculo (top / skip + 1). Alguns serviços RESTs usam `page` ao invés de `top` e `skip`.

  Por padrão, ao requisitar a primeira página, a constante `page` retorna o valor 1. Porém, existem APIs que possuem a página 0 e consideram a página 1 como a segunda página. Nesses casos, o recomendado é usar uma expressão FTL no campo conteúdo (destaque b da figura 6): `expression: ${page - 1}`
- **per_page**: (paginação) igual a top.
- **limit**: (paginação) igual a top.
- **applicationId**: identificador da aplicação em execução, mais detalhes em Multi aplicações.
- **offset**: (paginação) igual a skip.
- **now**: data e hora atual.
- **guid**: novo UUID

Exclusivo nos campos da aba Eventos:

- **eventName**: (exclusivo em parâmetros da aba Eventos) nome do evento que fez a requisição (ex.: onError).

Exclusivos do campo **Ao gerar Erro** (destaque 1 da figura 4.3) da aba Eventos:

- **exception**: objeto da exceção.
- **exceptionMessage**: mensagem da exceção.

Diferenças data, formData e rawEntry

As 3 constantes retornaram o objeto manipulado antes de ser inserido, atualizado ou removido. Após inserir ou modificar, o atributo pode sofrer modificações por eventos da Fonte de dados ou ações no banco de dados (ex.: valores default, *triggers*, auto incrementos etc).

A constante `rawEntry` retorna o mapa completo da requisição enviada pela Fonte de dados (ex.: POST), possui o objeto da entidade vinculada com a Fonte de dados, campos calculados, *metadatas*, URI da requisição e outras informações. Já as constantes `formData` e o `data` funcionam de forma parecida, retornando apenas o objeto da entidade vinculada com a Fonte de dados. A constante `data` é mais antiga e apenas consta na lista de constantes para manter compatibilidade com projetos mais antigos.

Imagine uma Fonte de dados vinculada a classe `Application` do Diagrama de dados que possui apenas 2 atributos (`id` e `name`), essa Fonte de dados teve sua consulta personalizada (bloco de código abaixo) e foi incluído um campo calculado de nome "`campoCalculado`".

```
select
        a.id,
        a.name,
        concat('Techne - ', a.name) as fullName
from
        Application a
```

O retorno das 3 constantes na chamada do evento Antes de Inserir será:

- **rawEntry**: `{"mediaMetadata": {"sourceLink":null, "etag":null, "contentType":null, "editLink":null}, "expandSelectTree": {"properties":[], "links":{}, "explicitlySelected":false, "expanded": false, "allKind":"IMPLICITLYTRUE", "all":true}, "metadata":{"id":null, "etag":null, "uri":null}, "properties":{"campoCalculado":"conteúdo do campo calculado", "_objectKey":"6F01513D-9953-43F7-ADEF-B08CC7763252", "name":"Cronapp", "fullName":null, "id":"6F01513D-9953-43F7-ADEF-B08CC7763252"}}`

- **formData**: `{"id":"6F01513D-9953-43F7-ADEF-B08CC7763252", "name":"Cronapp"}`
- **data**: `{"id":"6F01513D-9953-43F7-ADEF-B08CC7763252", "name":"Cronapp"}`

Parâmetros de Query String

A Fonte de dados trabalha em cima do padrão OData, dessa forma, os recursos dessa tecnologia (filtro, conversão, paginação, ordenação e outros) funcionarão com a Fonte de dados. Veja na documentação oficial mais detalhes sobre os parâmetros utilizados pelo OData.

Os parâmetros apresentados nesse tópico são úteis, por exemplo, quando se quer obter os dados de uma Fonte para alimentar um sistema desenvolvido fora do Cronapp. Os Componentes visuais e demais recursos do Cronapp já são preparados para trabalhar de forma integrada, sem a necessidade desse nível de configuração.

É possível obter os parâmetros passados em uma transação com a Fonte de dados utilizando o bloco de programação Obter parâmetro da query string. Veja um exemplo de uso desse recurso no tutorial Tratar recurso REST de terceiros no Cronapp.

- **$filter**: permite filtrar os registros retornados a partir de regras, para isso, existem diversos operadores lógicos, aritméticos e funções.
  - **Valores aceitos**: acesse a documentação oficial para mais detalhes de como configurar o parâmetro filtro.
  - **Exemplo**: a requisição abaixo retornará apenas registros em que o atributo "price" seja maior ou igual a "20".
    `<URI_da_FonteDados>?$filter=Price gt 20`
  - **Exemplo 2**: a requisição abaixo retornará apenas registros em que o atributo "price" seja maior ou igual a "20" e que o ano do atributo "ReleaseDate" seja igual a "2006".
    `<URI_da_FonteDados>?$filter=Price gt 20 and year(ReleaseDate) eq 2006`
- **$format**: por padrão, a Fonte de dados Cronapp sempre retorna seus dados em formato XML, porém é possível alternar para JSON.
  - **Valores aceitos**: aceita os valores "xml" ou "json".
  - **Exemplo**: a requisição abaixo retornará a lista de objetos em formato JSON.
    `<URI_da_FonteDados>?$format=json`
- **$inlinecount**: inclui o atributo "__count" na raiz do objeto de retorno de requisição, o valor desse atributo corresponde ao número total de registros na base de dados, ignorando qualquer filtro passado. Esse recurso é utilizado para realizar paginação.
  - **Valores aceitos**: aceita os valores "allpages" ou "none", o valor "none" equivale a não informar o atributo "$inlinecount".
  - **Exemplo**: a requisição abaixo retornará apenas os 2 primeiros registros, porém no atributo "__count" exibirá o valor do total de registros na base de dados.
    `<URI_da_FonteDados>?$top=2&$inlinecount=allpages`
- **$orderby**: ordena a lista requisitada a partir de um atributo do objeto.
  - **Valores aceitos**: necessário informar o nome do atributo mais "asc" para ascendente ou "desc" para descendente.
  - **Exemplo**: a requisição abaixo retorna uma lista onde os objetos que possuem o atributo "price" com valores maiores estarão no começo.
    `<URI_da_FonteDados>?$orderby=price desc`
  - **Exemplo 2**: a requisição abaixo retorna uma lista onde os objetos que possuem o atributo "price" com valores menores estarão no começo. Se não informar a ordem (asc ou desc) o OData reconhece como ascendente.
    `<URI_da_FonteDados>?$orderby=price`
- **$select**: especifica quais os atributos retornados na lista de objetos.
  - **Valores aceitos**: aceita o nome dos atributos separados por vírgula ",".
  - **Exemplo**: a requisição abaixo retorna uma lista de objetos contendo apenas os atributos "name", "price".
    `<URI_da_FonteDados>?$select=name,price`
- **$skip**: ignora os primeiros N registros. Ao requisitar $skip=9 em uma coleção de 10 registros, será retornado uma lista com 1 registro, o último.
  - **Valores aceitos**: aceita números inteiros.
  - **Exemplo**: a requisição abaixo retorna uma lista de elementos que começa a partir do elemento 10.
    `<URI_da_FonteDados>?$skip=9`
- **$top**: a quantidade de registros obtidos pela Fonte de dados será igual ao valor passado por esse parâmetro.
  - **Valores aceitos**: Aceita números inteiros.
  - **Exemplo**: independente da quantidade de registros, a requisição abaixo só retornará uma lista com 5 primeiros registros.
    `<URI_da_FonteDados>?$top=5`