Disponibilizando Web Service REST

O Cronapp permite disponibilizar serviços REST por Bloco de programação ou Fonte de dados. Veremos a seguir como configurar nos dois casos. E, ao final, mostraremos como obter esses recursos usando um programa externo ao Cronapp.

Pré-requisitos

1. Projeto do tipo mobile ou web criado, caso haja dúvidas de como criar esse tipo de projeto acesse o link criar projeto;
2. Saber utilizar as funcionalidades do diagrama, como criar classes e gerar persistência, caso haja dúvidas acesse o link diagrama;
3. Saber criar um bloco de programação, caso haja dúvidas acesse o link bloco de programação;
4. Saber criar uma fonte de dados, caso haja dúvidas acesse o link fonte de dados.

Configurando o diagrama

Nesse tutorial vamos criar uma agenda simples utilizando um relacionamento de 1 para N, onde um cliente pode possuir vários telefones. Vamos começar criando a classe **Cliente** com os atributos `id` (inteiro) e `nome` (Texto). Em seguida, vamos adicionar a classe **Telefone** com os atributos `id` (inteiro), `numero` (Texto) e `ehFixo` (Lógico). Por fim, vamos adicionar o relacionamento 1 to N (destaque 1 da figura 1) entre a classe **Cliente** e a classe **Telefone**.

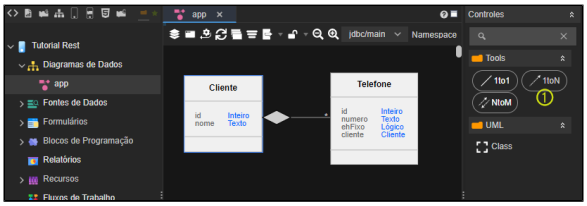

Figura 1 - Configurando o relacionamento entre classes

Após configurar as classes e o relacionamento, vamos gerar a camada de persistência e as páginas CRUD.

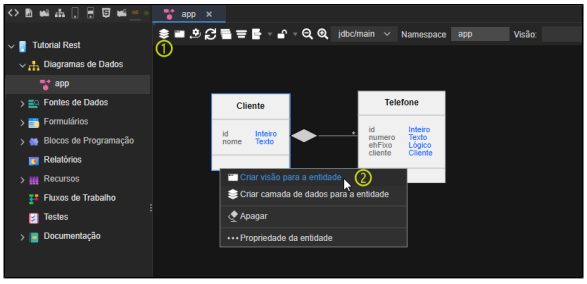

Figura 1.1 - Gerando o CRUD e a camada de persistência

1. Parar gerar a camada de persistência, vamos clicar no **ícone** (1 da Figura 1.1) e, na nova janela, clicar no botão **Gerar**;
2. Para gerar as páginas CRUD, vamos clicar com o botão direito do mouse sobre a entidade `Cliente`, selecionar **Criar visão para a entidade** (2), escolher o modelo de formulário CRUD Web na nova janela e avançar até finalizar.

Repita o procedimento 2 na classe `Telefone`.

Veremos a seguir como disponibilizar serviço REST via Bloco de programação e depois via Fonte de dados.

Nesta página

- Pré-requisitos
- Configurando o diagrama
- Via Bloco de programação
  - Configuração
    - Parâmetro REST
    - Propriedades do bloco
  - Teste do serviço via Bloco
  - Informação extra
    - Parâmetro Query String
    - Tipo do conteúdo da requisição
      - Texto Simples
      - JSON
      - Formulários
      - Formulário com envio de arquivos
- Via Fonte de dados
  - Configuração
  - Teste do serviço via Fonte de dados
    - Formato da requisição (XML / JSON)
- Recursos privados
  - Restrição via bloco
    - Restrições diferentes para os verbos HTTP
  - Restrição via Fonte de dados
  - Teste dos recursos privados
    - Obter e atualizar o token
    - Requisição via Bloco privado
    - Requisição via Fonte de dados privado

Outras informações:

Web Services

Via Bloco de programação

Para configurar o serviço REST via bloco, vamos iniciar criando um bloco de programação do lado Servidor (Figura 2). Siga os passos abaixo.

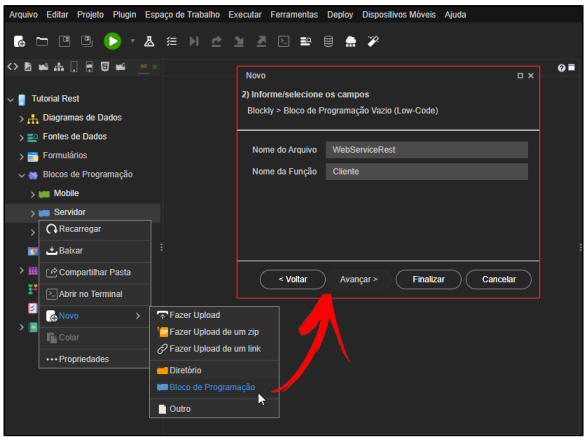

Figura 2 - Nova função de bloco de programação

- Clique com o botão direito na pasta **Servidor** e no menu de contexto escolha **Novo > Bloco de Programação**;
- Na janela **Novo** vamos selecionar "Bloco de Programação Vazio (Low-code)", clicar em **Avançar** , preencher os campos abaixo e clicar em **Finalizar**.
    - **Nome do Arquivo:** WebServiceRest;
    - **Nome da Função:** cliente.

Configuração

Parâmetro REST

Vamos começar a configuração criando o parâmetro `idCliente` que passará como argumento para a função o identificador do cliente. Clique na engrenagem da função Cliente (destaque 1 da Figura 2.1), arraste o bloco **nome de entrada** para dentro do bloco **entradas** e altere o nome da entrada para **idClie nte** (destaque 2). O parâmetro criado ficará disponível na categoria **Variáveis** (destaque 3). Para mais detalhes de configuração dos parâmetros de uma função, acesse o tópico "Parâmetros da função" da documentação Variável.

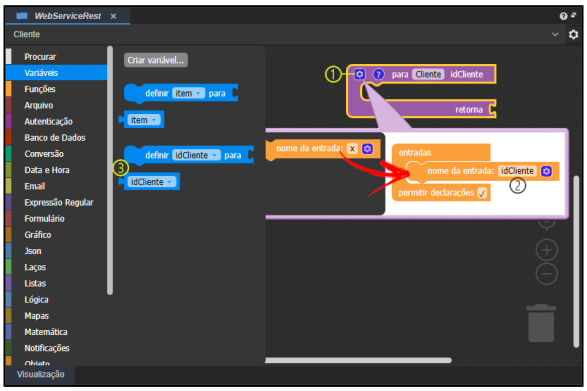

Figura 2.1 - Adicionando parâmetro de entrada à função

Agora arraste o bloco Abrir consulta (categoria Banco de dados) para que os dados sejam retornados, encaixe-o no retorno da função Cliente (Figura 2.2) e configure-o conforme os passos abaixo.

Caso tenha dúvidas de como configurar a consulta, acesse a documentação do Assistente de consulta.

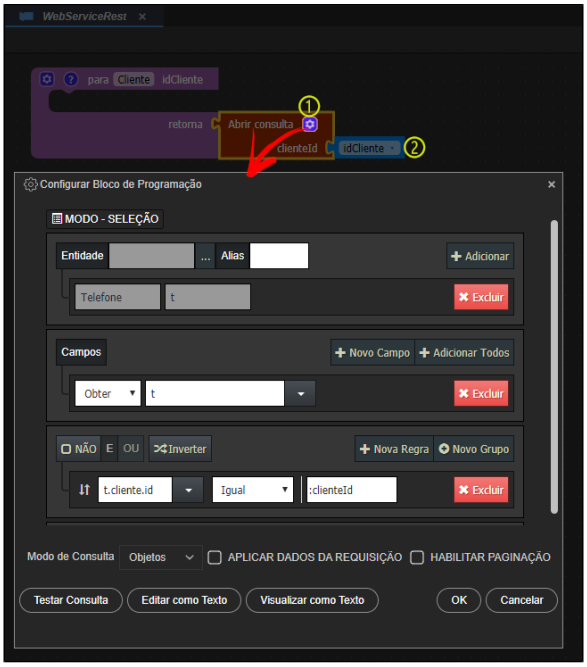

Figura 2.2 - Configurando o bloco de programação e consulta

1. Clique na engrenagem do bloco **Abrir consulta** e na janela configure seguindo os passos abaixo:
    - Selecione a entidade **Telefone** em "..." e clique em **Adicionar**;
    - Adicione um **Novo Campo** e escolha a opção **Obter t**;
    - Adicione uma **Nova Regra** e selecione **t.cliente.id**, o restante da expressão será preenchido de forma automática;
    - Clique em **OK** ao final.
2. Por fim, adicione a variável **idCliente** (categoria Variáveis) no parâmetro **clienteId** que apareceu no bloco Abrir Consulta.

Propriedades do bloco

Agora precisamos alterar as propriedades do bloco de programação, clique na engrenagem **Configuração dos tipos de regras** (lado direito/superior) para abrir a janela de **Propriedades do bloco de programação** (Figura 2.3). Nessa janela, é importante entender os campos que estão descritos abaixo:

1. **Tempo limite**: limite em segundos para a execução. Após o limite, a ação é finalizada.
2. **Tipo**: define o tipo de permissão para acesso ao bloco, sendo:
    - **Interno**: o bloco só é consumido dentro do projeto;
    - **Externo**: permite que o bloco seja também consumido fora do projeto.
3. **Segurança**: Abre outra janela que nos permite dar autorização de CRUD e Execução aos permissionáveis do sistema.
4. **Endereço REST**: exibe e permite configurar as URIs do endereço do serviço REST de cada função do bloco de programação, que será concatenado com o domínio do sistema, bem como a configuração dos tipos de operações HTTP utilizadas, o tipo dos dados enviados e recebidos na requisição e o tipo de segurança. Para mais informações, acesse o tópico "Propriedades do Blockly" da documentação Bloco de programação.

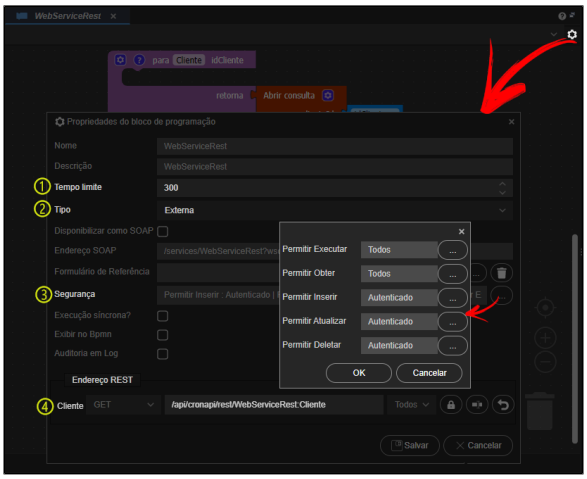

Figura 2.3 - Configuração da chamada REST

1. Mantenha o tempo limite configurado com o valor padrão, 300 segundos.
2. Selecione a opção **Externa** no campo **Tipo**.
3. No campo **Segurança**, clique no botão "**...**" e altere as permissões **Permitir Executar** e **Permitir Obter** para a opção **Todos**;
4. Copie o endereço do campo **Cliente**, ele será utilizado para testar o serviço.

Por fim, vamos salvar tudo, executar a aplicação, pois a requisição REST via bloco de programação já estará disponível.

Teste do serviço via Bloco

Para testar e demostrar os resultados desse tutorial usaremos a aplicação Insomnia, mas existem diversas opções como: Postman, ReqBin e outras.

Antes de partir para os próximos passos, é necessário que os dados estejam cadastrados. Por isso, rode a aplicação, abra no navegador Web e cadastre alguns clientes e telefones.

Dica

Para acessar diretamente as páginas Cliente e Telefone, configure a URL do projeto concatenando a pasta (nome no modo avançado) e o nome do formulário.

Exemplo: para acessar o formulário **`cliente`** que está dentro da pasta **`logged`** (Autenticado), deve-se adicionar **`/logged/cliente`**. O endereço deverá ficar assim: `https://app-30-205-11680.ide.cronapp.io/#/home/logged/cliente`

Para esse tutorial vamos usar o domínio temporário gerado no debug do Cronapp (ex.: `https://app-30-205-11680.ide.cronapp.io/`). Quando o seu sistema estiver em produção e com domínio próprio (ex.: `https://www.cronapp.io`), utilize-o para concatenar com o endereço (URN) do serviço REST apontado no campo **Endereço REST** (destaque 4 da Figura 2.3).

Considere que endereço REST é formado pelos principais elementos abaixo:

```
<Domínio>/api/cronapi/rest/blockly.<Nome do Blockly>:<Nome da Função>[
/<Parâmetro 1>[/Parâmetro N]]
```

- **Domínio:** é o domínio da aplicação. Nesse tutorial estamos utilizando o domínio temporário. Ex.: `https://app-11-213-49431.ide.cronapp.io`;
- **Nome do Blockly:** é o arquivo Blockly ou bloco de programação. Ex.: WebServiceRest;
- **Nome da Função:** é o nome da função que se deseja acessar. Ex.: Cliente;
- **Parâmetro:** é o valor do parâmetro passado na função, nesse tutorial estamos utilizando o ID do cliente como parâmetro. Ex.: 1.

O endereço REST completo ficará:

```
https://app-11-213-49431.ide.cronapp.io/api/cronapi/rest/WebServiceRest
:Cliente/1
```

Assim, abra o seu cliente de teste Rest, informe o verbo GET (destaque 1 da figura 2.4), informe o endereço da requisição (2) e clique no botão **Send**. Requisições REST de um bloco de programação pode retornar qualquer formato, porém, como estamos obtendo o conteúdo do bloco Abrir consulta, o conteúdo será exibido através de uma lista de objetos em formato JSON (3).

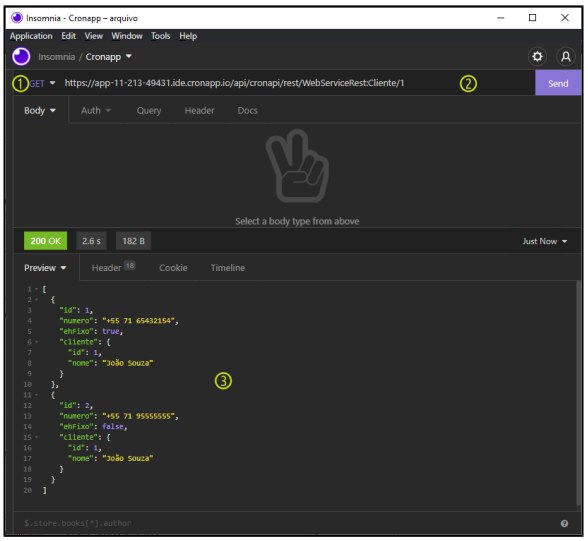

Figura 2.4 - Retorno do serviço via Bloco de programação

Informação extra

Neste tópico vamos apresentar algumas informações importantes sobre o **recurso REST via bloco de programação** que podem ser úteis em seu projeto. São eles:

- Parâmetro Query String
- Content Type

Parâmetro Query String

Também é possível obter parâmetros por *query string*. Explicaremos sobre esse recurso nesse tópico extra.

Os parâmetros via *query string* também pode ser passados junto com os parâmetros de rotas, dessa forma, é possível efetuar uma requisição como no trecho da URI abaixo, em que após o parâmetro "3123" (identificador do cliente), usamos o caractere interrogação "?" para informar os parâmetros query string e seus valores.

Diferentemente dos parâmetros por Rotas, não é necessário se preocupar com a ordem dos parâmetros query string definidos na URI. Para obter seus valores, basta utilizar o bloco Obter parâmetro da query string, conforme as informações abaixo e a Figura 2.5.

Rode o projeto, abra a aplicação no navegador Web e cole o endereço abaixo, alterando as informações conforme seu projeto:

```
<Domínio>/api/cronapi/rest/WebServiceREST:Cliente/3123?param1=Cronapp&p
aram2=low-code
```

- **Domínio:** é o domínio da aplicação. Nesse tutorial estamos utilizando o domínio temporário. Ex.: `https://app-30-205-11680.ide.cronapp.io`;

- **Endereço REST:** é encontrado nas Propriedades do bloco (item 4 da Figura 2.3), contendo o nome do bloco, o nome da função e o nome de entrada definidos. Observação: Note que o **`<id Cliente>`** foi passado na URI como `3123`;
- **?**: separa o endereço URI dos parâmetros query string;
- **Parâmetro=valor**: nome do parâmetro e seu valor. Ex.: `param1=Cronapp`;
- **& (ampersand):** permite adicionar outro parâmetro=valor.

O endereço REST completo ficará: `https://app-30-205-11680.ide.cronapp.io/api/cronapi/REST/WebServiceREST:Cliente/`**`3123`**`?param1=`**`Cronapp`**`&param2=`**`low-code`**

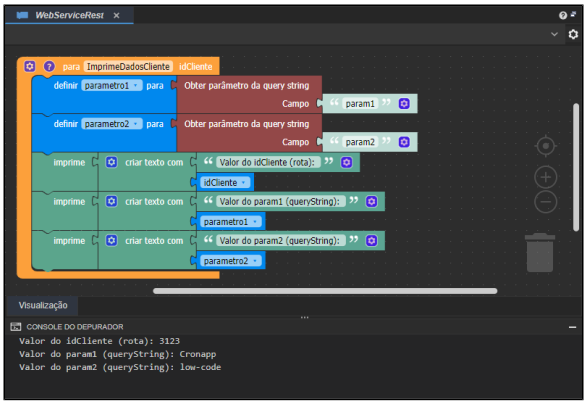

Figura 2.5 - Obtendo parâmetros via Rotas e Query string

Tipo do conteúdo da requisição

O content-type é uma propriedade do cabeçalho da requisição e tem o objetivo de informar para quem irá receber a requisição (servidor do recurso) qual o tipo de conteúdo está sendo passado no corpo da requisição. Essa propriedade não é utilizada em requisições Para Obter (verbo GET), porém, é necessária em requisições POST, PUT e DELETE.

Quem define o tipo do conteúdo e a forma como a API irá funcionar é o servidor do recurso. Dessa forma, é importante que exista uma documentação básica mostrando como o cliente pode gerar as requisições. Mostraremos nos subtópicos abaixo como podemos obter os diferentes tipos de conteúdo no corpo de uma requisição POST, usada como exemplo.

Para exemplificar os tipos, usaremos a mesma aplicação cliente que usamos nos testes dos blocos de programação, o Insomnia. Para alterar entre os tipos passados na requisição, basta clicar na primeira aba e selecionar uma das opções apresentadas (Figura 2.6)

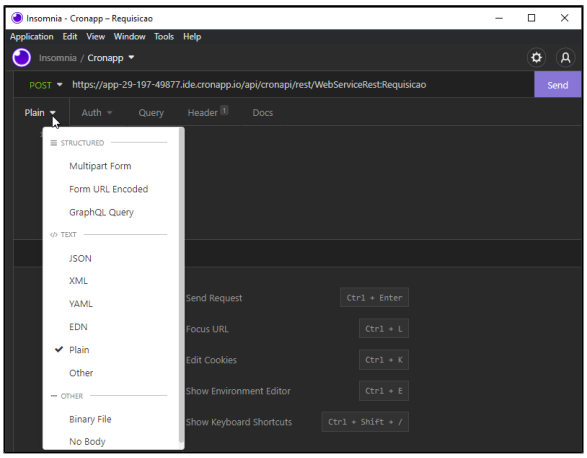

Figura 2.6 - Selecionando o formato que será passado no corpo da requisição

Após selecionar a opção (Figura 2.6), o valor do parâmetro Content-Type, no cabeçalho da requisição, será ajustado automaticamente para o tipo selecionado (Figura 2.6.1).

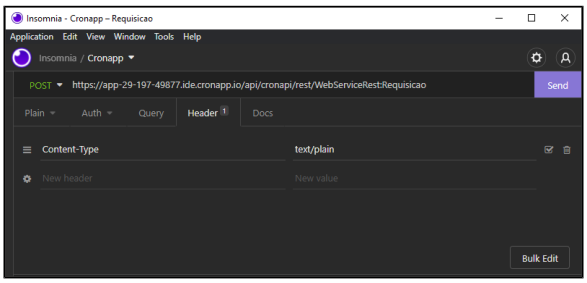

Figura 2.6.1 - O formato do corpo da requisição deve ser informado no parâmetro Content-type do cabeçalho

Texto Simples

O formato **text/plain** informa que o corpo da requisição possui um texto simples. No exemplo abaixo (Figura 2.6.2) informamos na rota do *endpoint* um parâmetro (**param**), esse valor será recebido como argumento pela função **RequisicaoTextPain**.

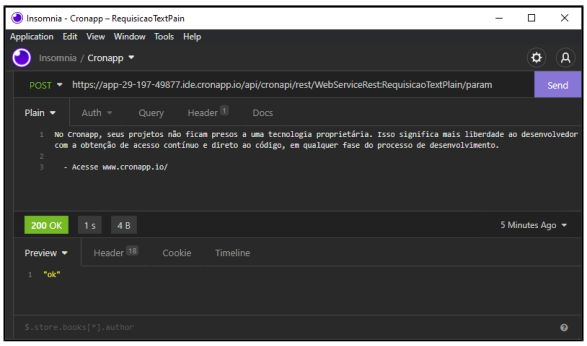

Figura 2.6.2 - Requisição e o seu retorno

Apesar de informamos um parâmetro na rota da requisição (param), ele não é obrigatório. O bloco de programação reconhece no primeiro argumento o valor do corpo da requisição. (ver exemplo no tópico **Formato Json**)

Após receber o conteúdo do corpo da requisição como argumento (**parametro**), a função **RequisicaoTextPain** imprime seu conteúdo e retorna o valor "ok" (Figura 2.6.3).

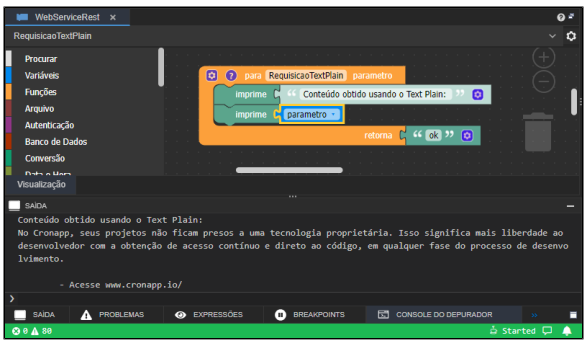

Figura 2.6.3 - Função obtendo o conteúdo no formato text/plain

JSON

O formato **application/json** informa que o corpo da requisição possui um JSON. No exemplo abaixo incluímos no corpo da requisição uma lista com 2 objetos.

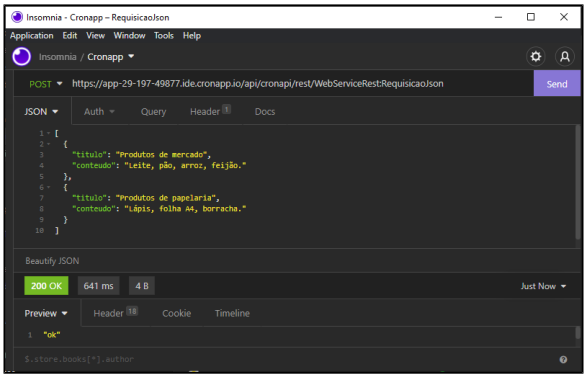

Figura 2.6.4 - Requisição e o seu retorno

Após receber o conteúdo do corpo da requisição como argumento (**json**), a função **RequisicaoJson** entra em loop para imprimir os atributos de seus objetos, em seguida retorna o valor "ok" (Figura 2.6.5).

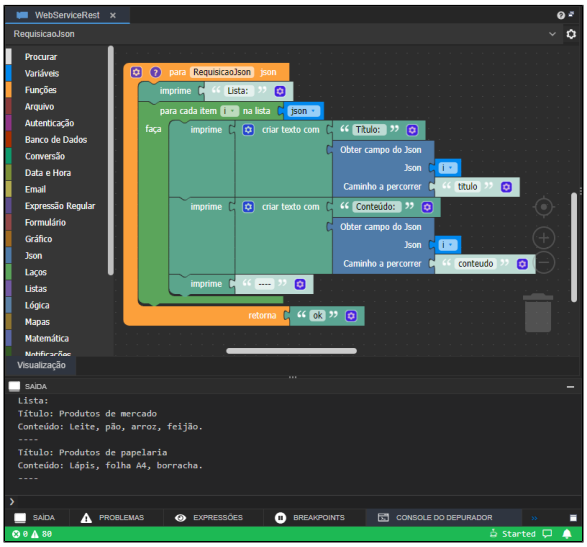

Figura 2.6.5 - Função obtendo o conteúdo no formato application/json

Formulários

O formato **application/x-www-form-urlencoded** normalmente é utilizado em formulários web, em seu corpo são enviados diversos campos contendo uma chave e o seu respectivo valor. Na requisição de exemplo (Figura 2.6.6) são passados 2 campos em seu corpo: **titulo** e **conteudo**.

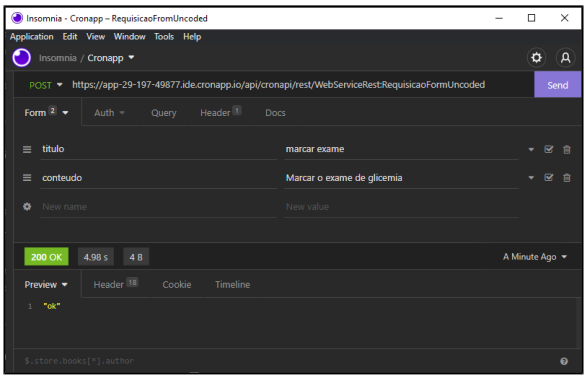

Figura 2.6.6 - Requisição e o seu retorno

Para obter os campos do formato Form, podemos usar o bloco de programação Obter parâmetro da query string e informar o nome do campo (Figura 2.6.7).

Figura 2.6.7 - Função obtendo os campos via parâmetro da requisição

Outra opção é receber os campos a partir de um mapa no parâmetro da função. Nesse caso, podemos usar o bloco Obter campo do mapa por chave e informar o parâmetro da função e o campo desejado (Figura 2.6.8).

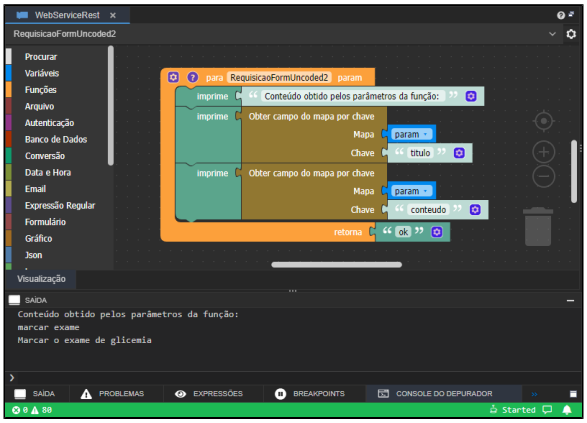

Figura 2.6.8 - Função obtendo os campos via parâmetro da função

Formulário com envio de arquivos

O formato **multipart/form-data** também é utilizado em formulários web, em seu corpo são enviados diversos campos contendo uma chave e o seu respectivo valor. A diferença desse formato em relação ao **application/x-www-form-urlencoded**, é que o **multipart/form-data** suporta em seus campos não só texto, mas também arquivos, convertidos em *array de bytes*.

Só é possível obter o campo que contém o arquivo a partir do parâmetro da função, logo não é possível obter o conteúdo do arquivo via bloco Obter parâmetro da query string.

Na requisição de exemplo (Figura 2.6.9) são passados 3 campos em seu corpo:

- **arquivo**: conteúdo do arquivo convertido em *array de bytes*.
- **extensao**: extensão do arquivo, usado para renomear o arquivo no servidor.
- **extra**: o *content-type form* permite adicionar quantos campos forem necessário, incluímos o campo "extra" apenas para demostrar isso. Será passado um valor qualquer e, em seguida, retornaremos na requisição esse mesmo valor.

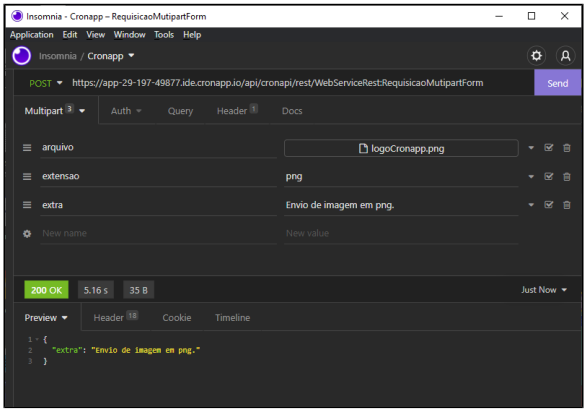

Figura 2.6.9 - Requisição e o seu retorno

A função que recebe a requisição (Figura 2.6.10) irá salvar o arquivo em um diretório no servidor da aplicação.

Vale lembrar que salvar arquivos no servidor da aplicação ou no banco de dados não é considerado uma boa prática, estamos tratando dessa forma apenas para facilitar o exemplo. Para esses casos, o mais recomendado seria utilizar em locais próprios para o arquivamento de dados, como o sistema de Armazenamento do Cloud Services do Cronapp.

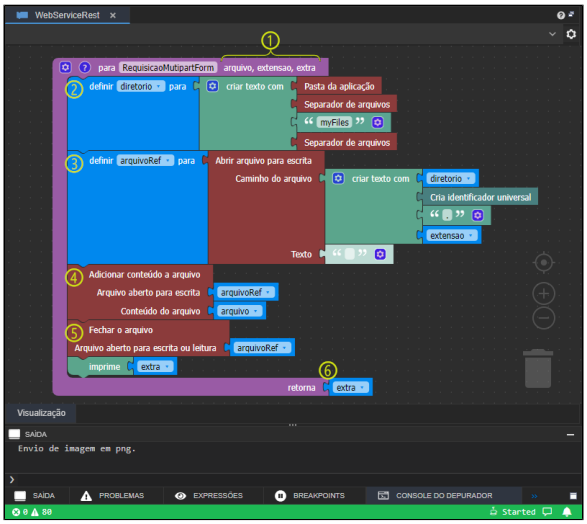

Figura 2.6.10 - Função que converte o array de bytes em arquivo

Nesse exemplo estamos criando duas variáveis (**diretorio** (destaque 2) e **arquivoRef** (destaque 3)) que serão usadas apenas dentro do escopo da função.

1. Os nomes dos parâmetros da função devem ser idênticos aos nomes dos campos da requisição (Figura 2.6.9).
2. A variável **diretorio** recebe o endereço de uma pasta na raiz do projeto em que salvaremos os arquivos recebidos.
3. A variável **arquivoRef** recebe a referência do novo arquivo através do bloco Abrir arquivo para escrita, em que passamos o endereço do diretório e o novo nome do arquivo, composto por um identificador universal e a extensão passada pelo parâmetro da função.
4. No bloco Adicionar conteúdo a arquivo passamos a variável com a referência do arquivo a ser gerado (**arquivoRef**) e o conteúdo do arquivo em *Array de Bytes* vindo do parâmetro da função (**arquivo**).
5. Por fim, usamos o bloco Fechar o arquivo e passamos a variável de referência do novo arquivo.
6. O conteúdo do parâmetro extra é impresso no console e passado de volta no retorno da função.

Após efetuar a requisição (Figura 2.6.9), o arquivo pode ser visualizado no diretório da aplicação do projeto (Figura 2.6.11).

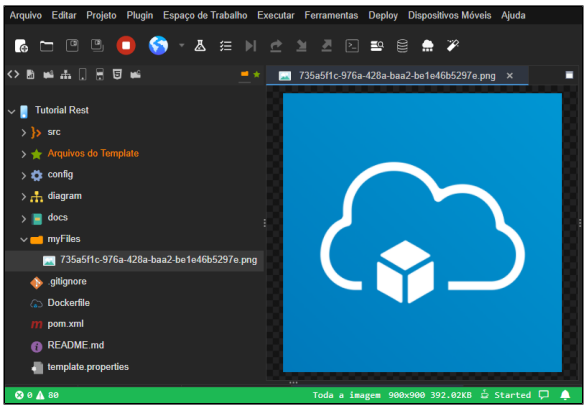

Figura 2.6.11 - Arquivo obtido por uma requisição POST

Via Fonte de dados

Veremos agora como disponibilizar recursos REST via Fonte dados. Diferentemente do bloco de programação, em que podemos retornar a requisição no formato definido no retorno da função (exemplo: JSON, XML, text plain ou outros), a fonte de dados é baseada no OData e, por padrão, sempre retornará um XML (exemplo), porém, como veremos

Vamos iniciar criando uma Fonte de dados, clique no ícone da **Fonte de Dados** e na janela **Buscar Fonte de Dados** clique no botão **Nova fonte de dados** (Figura 3).

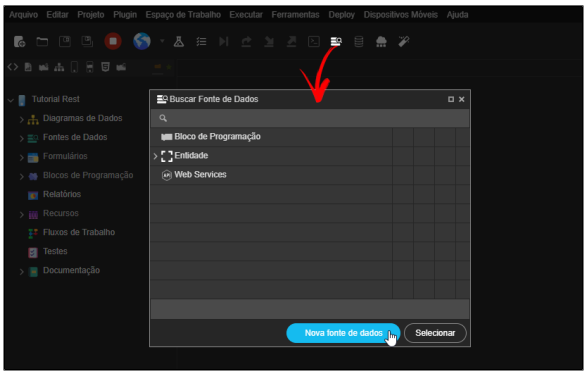

Figura 3 - Criando nova Fonte de dados

Configuração

Com a janela da Fonte de dados aberta, vamos configurar as informações da fonte de dados e a consulta ao banco na aba Filtros.

Para mais informações, acesse a documentação da Fonte de dados.

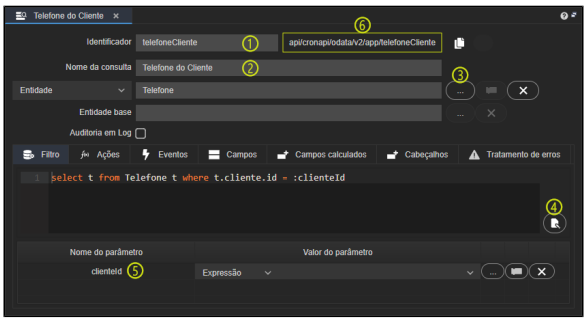

Figura 3.1 - Configuração das informações da fonte e sua consulta

1. **Identificador**: fonte de dados que será usado na URI do serviço REST. Estamos utilizando `telefonesCliente`.
2. **Nome da consulta**: nome que será exibido na lista de fonte de dados do projeto. Estamos utilizando `Telefone do Cliente`.
3. **Entidade**: clique no botão "`...`" da entidade e selecione a entidade `Telefone.`
4. **Configura consulta**: clique no botão para personalizar a consulta da fonte de dados. Na nova janela, clique em **Nova Regra** e configure os campos com a regra (conforme a Figura 2.2): **`t.cliente.id = :clienteId`**
5. **Parâmetros**: após salvar a consulta (passo 4), o parâmetro aparecerá. Não é necessário definir nenhum valor. Ele será usado na consulta e passado na *query string* do endereço.
6. **Endereço REST**: trecho final (URN) do endereço do serviço REST, será concatenado após o domínio do sistema.

Agora vamos configurar as permissões de acesso. Vá até a aba **Ações**, clique no cadeado do verbo **Permitir Obter** para abrir a janela Permissão de Segurança, deixe apenas a opção **Todos** selecionada. Repita em **Permitir Filtrar** e **Permitir Contar**.

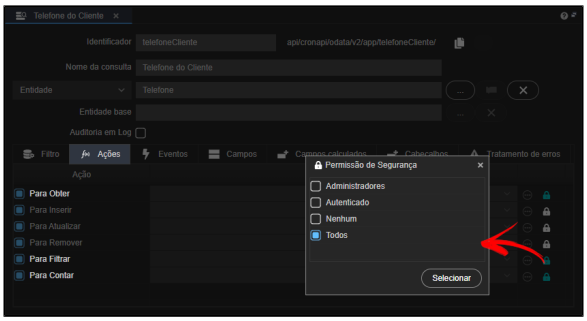

Figura 3.2 - Configuração das permissões da fonte de dados

Por fim, vamos salvar tudo, executar a aplicação e a requisição REST via fonte de dados já estará disponível.

Teste do serviço via Fonte de dados

Para testar e demostrar os resultados desse tutorial usaremos a aplicação Insomnia, mas existem diversas opções como: Postman, ReqBin e outras.

Antes de partir para os próximos passos, é necessário que os dados estejam cadastrados. Por isso, rode a aplicação, abra no navegador Web e cadastre alguns clientes e telefones.

Dica

Para acessar diretamente as páginas Cliente e Telefone, configure a URL do projeto concatenando a pasta (nome no modo avançado) e o nome do formulário.

Exemplo: para acessar o formulário **`cliente`** que está dentro da pasta **`logged`** (Autenticado), deve-se adicionar **`/logged/cliente`**. O endereço deverá ficar assim: `https://app-30-205-11680.ide.cronapp.io/#/home`**`/logged/cliente`**

Para esse tutorial vamos usar o domínio temporário gerado no debug do Cronapp (ex.: `https://app-30-205-11680.ide.cronapp.io/`). Quando o seu sistema estiver em produção e com domínio próprio (ex.: `https://www.cronapp.io`), utilize-o para concatenar com o endereço (URN) do serviço REST apontado no campo **Identificador** (destaque 6 da Figura 3.1).

Considere que endereço REST é formado pelos principais elementos abaixo:

```
<Domínio>/api/cronapi/odata/v2/app/<Identificador da fonte>[?<Parâmetro 1>=<Valor do Parâmetro 1>[&<Parâmetro n>=<Valor do Parâmetro n>]]
```

Considerando os elementos abaixo:

- **Domínio:** é o domínio da aplicação. Nesse tutorial estamos utilizando o domínio temporário. Ex.: https://app-30-205-11680.ide.cronapp.io/;
- **Identificador da fonte:** é o identificador da Fonte de Dados (item 1 da Figura 3.1). Ex.: TelefonesCliente;
- **Parâmetro:** clienteId (obtido através da lista de parâmetros, destaque 5 da Figura 3.1);
- **Valor do parâmetro:** é o valor do parâmetro passado na função, nesse tutorial estamos utilizando o ID do cliente como parâmetro. Ex.: 1.

O endereço REST completo ficará: `https://app-30-205-11680.ide.cronapp.io`**`/api/cronapi/odata/v2/app/TelefonesCliente?clienteId=2`**

Assim, abra o seu cliente de teste Rest, informe o verbo GET (destaque 1 da figura 3.4), informe o endereço da requisição (2) e clique no botão **Send**. Fontes de dados seguem o padrão OData, dessa forma, o resultado da requisição vem no formato XML (3). Veja o próximo tópico para saber como alternar para o formato JSON.

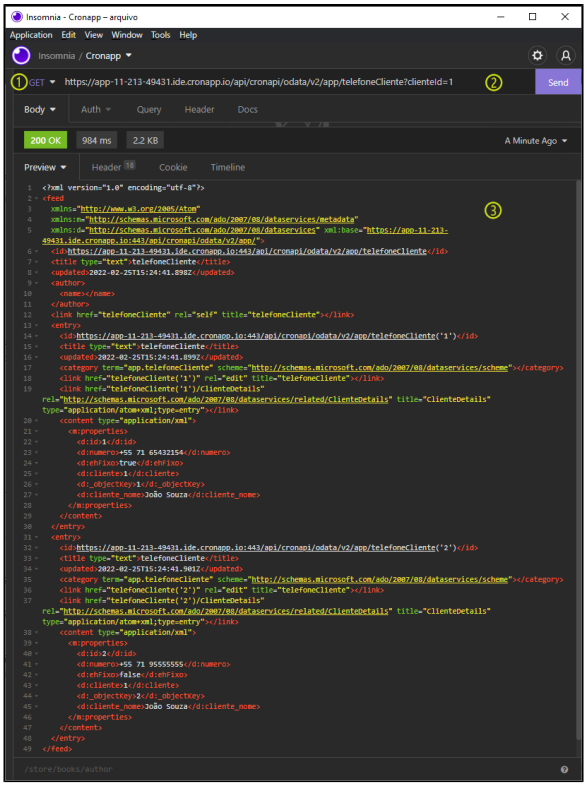

Figura 3.4 - Retorno do serviço via Fonte de dados

Formato da requisição (XML / JSON)

Como informado acima, a Fonte de dados é baseada na tecnologia OData e no Cronapp sempre retornará o formato XML quando requisitada. Porém, o OData também suporta o formato JSON, e assim podemos alterar o formato passado pela fonte de dados apenas incluindo o parâmetro "`$format=json`" no endereço da requisição.

Abaixo utilizamos a mesma requisição da figura 3.4, porém incluindo o parâmetro Query String "`$format=json`" (destaque 1 da figura 3.5) para alterar o retorno para o formato JSON.

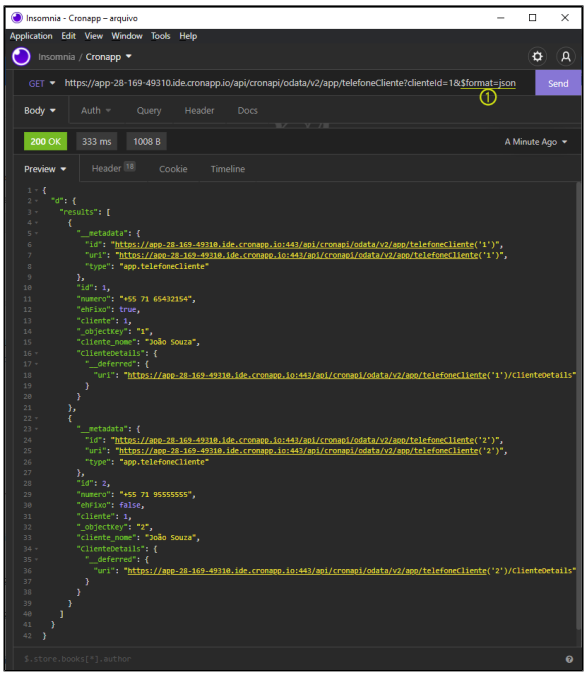

Figura 3.5 - Retorno do serviço via Fonte de dados em formato JSON

Recursos privados

Nos exemplos acima apresentamos como disponibilizar recursos públicos, dessa forma, qualquer usuário que possua o endereço poderá obter os dados. Veremos agora como tornar esses mesmos recursos privado, ou seja, para obter os dados o usuário precisar ter acesso ao sistema (login e senha) e possuir a mesma permissão de segurança do recurso.

Restrição via bloco

Para tornar o recurso Rest via Bloco de programação privado, basta acessar as propriedades do Bloco de programação (passo no exemplo acima) e na propriedade **Segurança** (destaque 1 da figura 4), definir o permissionável para cada verbo HTTP. Nesse exemplo deixaremos todos os verbos com autorização para o permissionável "Autenticado" (Padrão).

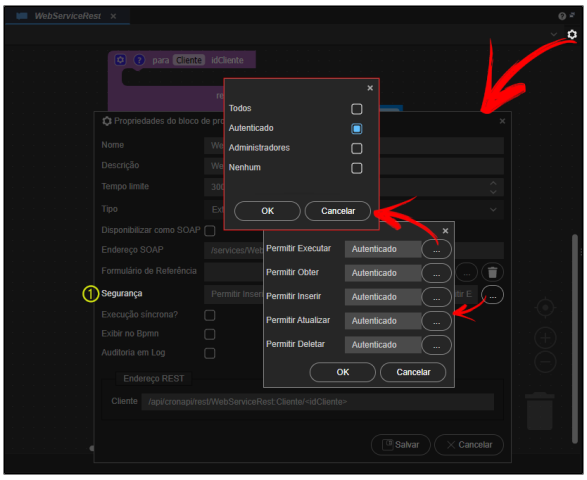

Figura 4 - Restringindo o acesso ao bloco de programação

Restrições diferentes para os verbos HTTP

Os blocos de programação permitem a utilização de todos os verbos (Obter, Inserir, Atualizar e Deletar) e Executar. Porém, as configurações feitas nas propriedades do Bloco de programação serão compartilhadas com todas as funções contidas no bloco de programação (arquivo blockly). Dessa forma, recomendamos criar um arquivo de bloco de programação (blockly) diferente para cada configuração de segurança diferente, exemplo: O recurso "A" terá o verbo **obter** público e o **inserir** autenticado, já os recursos "B" e "C" terão os verbos **obter** público, mas o **inserir** só será permitido para os Administradores (Figura 4.1).

No exemplo da figura 4.1, as funções **RecursoB** e **RecursoC** (destaques 1) possuem as mesmas restrições de acesso (destaques 2).

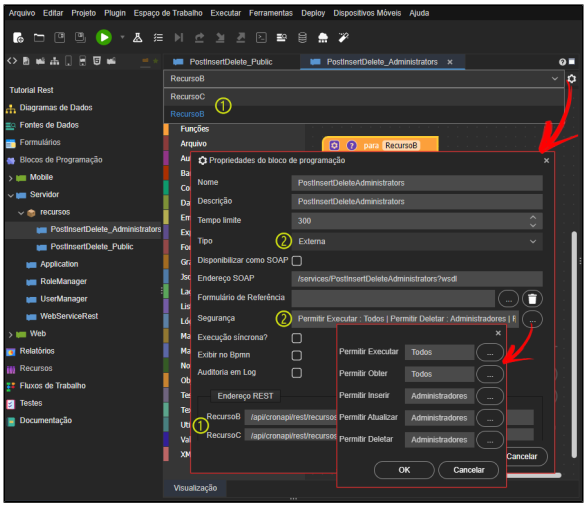

Figura 4.1 - Criando arquivos de blocos diferentes para configurações de segurança diferentes

Restrição via Fonte de dados

Para tornar o recurso Rest via Fonte de dados privado, basta acessar a aba Ações (passo no exemplo acima) e na coluna **Segurança** (destaque 1 da figura 4.2), definir o permissionável para cada Ação. Nesse exemplo deixaremos todos os verbos com autorização para o permissionável "Autenticado".

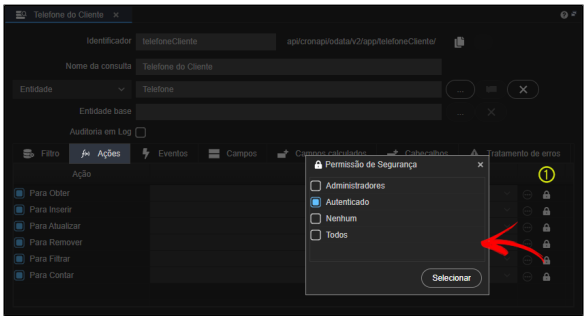

Figura 4.2 - Restringindo o acesso a Fonte de dados

Teste dos recursos privados

Para acessar recursos privados nos projetos Cronapp será necessário obter um token de autorização e, em seguida, fazer a requisição utilizando esse token.

Obter e atualizar o token

Por padrão, o token de acesso é válido por 3600 segundos, mas é possível alterar esse tempo no campo **Expiração do Token (segundos)** na janela de Configurações do projeto (aba **Configurações do projeto**). Enquanto o token estiver válido, é possível realizar diversas requisições utilizando o mesmo token.

Para obter o token, é necessário realizar uma requisição **POST** passando o usuário e senha no corpo da requisição.

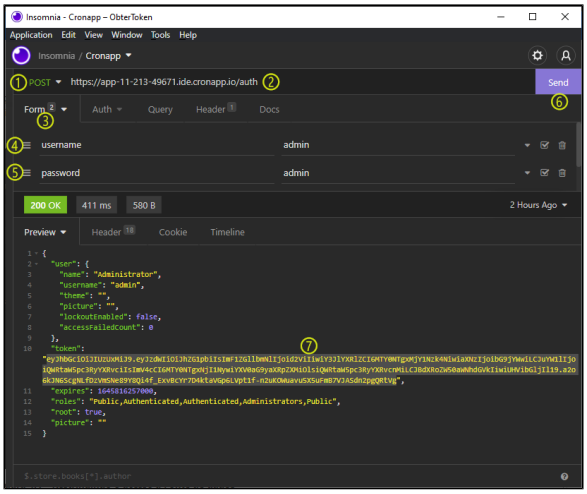

Figura 4.3 - Obtendo o token da requisição

1. Defina a requisição como POST.
2. Informe o *endpoint*: `<Domínio>/auth`
   Ex.: `https://app-11-213-49671.ide.cronapp.io/auth`
3. Configure o corpo da requisição como "Form URL Encoded".
4. No primeiro campo do corpo da requisição informe "username" e o seu valor, nesse exemplo usamos "admin".
5. No segundo campo do corpo da requisição informe "password" e o seu valor, nesse exemplo usamos "admin".
6. Execute a requisição.
7. A requisição retornará diversos dados do usuário, porém o que precisamos é do atributo **token**.

Antes de finalizar o tempo de expiração do token, será necessário fazer uma nova requisição para obter uma atualização do mesmo. A requisição deve ser feita no formato abaixo:

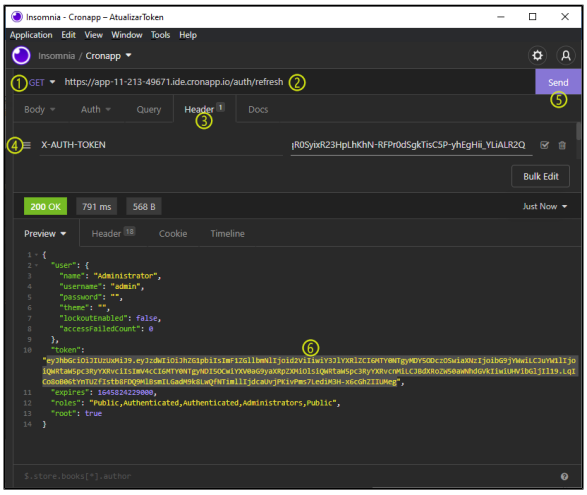

Figura 4.4 - Requisitando a atualização do token de acesso

1. Defina a requisição como GET.
2. Informe o *endpoint*: `<Domínio>/auth/refresh`
   Ex.: `https://app-11-213-49671.ide.cronapp.io/auth/refresh`
3. Configure um campo para o cabeçalho da requisição.

4. No campo do cabeçalho da requisição informe "X-AUTH-TOKEN" e o token obtido no passo anterior (destaque 6 da figura 4.4).
5. Execute a requisição.
6. A requisição retornará diversos dados do usuário, porém o que precisamos é do atributo **token**.

Requisição via Bloco privado

A requisição do Bloco de programação privado deve seguir o padrão abaixo:

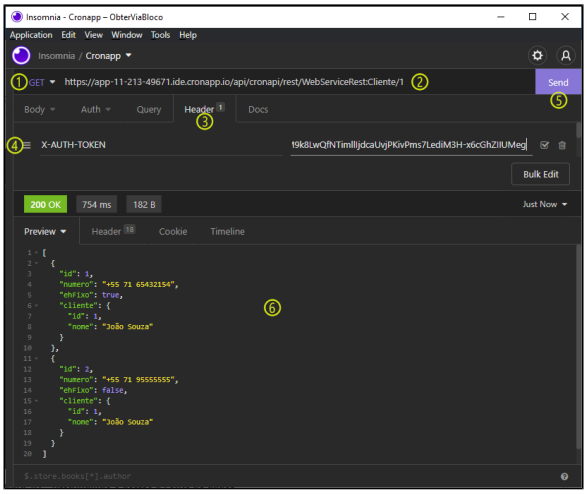

Figura 4.5 - Obtendo dados de uma requisição privada via Bloco de programação

1. Defina o verbo da requisição (exemplo GET).
2. Informe o *endpoint* da requisição.
3. Configure um campo para o cabeçalho da requisição.
4. No campo do cabeçalho da requisição informe "X-AUTH-TOKEN" e o token obtido (destaque 7 da figura 4.3 ou destaque 6 da figura 4.4).
5. Execute a requisição.
6. A requisição retornará os dados solicitados.

Requisição via Fonte de dados privado

A requisição da Fonte de dados privada deve seguir o padrão abaixo:

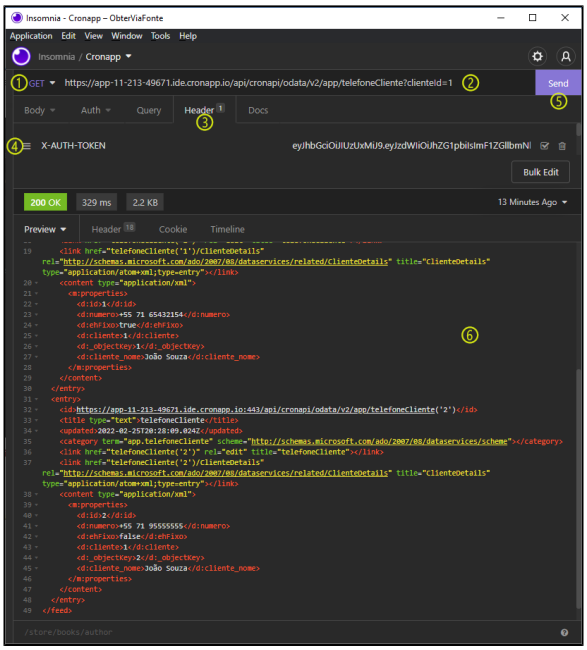

Figura 4.6 - Obtendo dados de uma requisição privada via Bloco de programação

1. Defina o verbo da requisição (exemplo GET).
2. Informe o *endpoint* da requisição.
3. Configure um campo para o cabeçalho da requisição.
4. No campo do cabeçalho da requisição informe "X-AUTH-TOKEN" e o token obtido (destaque 7 da figura 4.3 ou destaque 6 da figura 4.4).
5. Execute a requisição.
6. A requisição retornará os dados solicitados.