Debug via código

Debug (depuração, em português) é o processo de encontrar e reduzir defeitos num *software*, por meio de uma análise detalhada do código. No Cronapp é possível depurar e definir os pontos de paradas diretamente nos blocos de programação (Servidor), através do Debug visual, ou da forma tradicional, aplicando diretamente no código gerado pelos blocos de programação (cliente ou Servidor).

Nesta página:

- Pré-requisitos
- Blocos cliente
  - Desminificar código
  - Iniciar a depuração
- Blocos servidor

Debug visual

Acesse o tópico **Debug visual** na documentação Bloco de programação para ver como depurar de forma *low-code*, diretamente pelos blocos de programação.

Pré-requisitos

Antes de começar é necessário ter certeza de que se possui um ambiente preparado para a realização do exemplo. Abaixo estão os principais requisitos necessários.

Requisitos:

1. Projeto do tipo web ou web-mobile criado. Caso haja dúvidas em relação a como criar esse tipo de projeto acesse a documentação Criar projeto.
2. Conhecimento na criação de bloco de programação. Caso tenha dúvidas acesse a documentação Bloco de programação.
3. É necessário que o botão Modo Avançado esteja habilitado.

Blocos cliente

Os blocos clientes são blocos front-end, que geram código JavaScript, em projetos mobile e web.

Desminificar código

Por ser uma boa prática e melhorar a performance, os códigos cliente são minificados, dessa forma precisamos exibir o código de forma legível para permite que o *breakpoint* seja adicionado no código e facilita a visualização. Assim, verifique se nos arquivos **index.html** dos projetos mobile ou web (endereços listados abaixo em visualização de pasta) estão apontando para o arquivo `cronapi.js` e não para sua versão minificada (`cronapi.min.js`). Para isso, abra o arquivo **index.html** (endereço list ado abaixo) e em seguida localize a tag script (`<script>`) que possui `node_modules/cronapi-js /dist/cronapi.min.js`, altere de **`cronapi.min.js`** para **`cronapi.js`** e remova **`/dist`**, retornando um nível na hierarquia de diretórios. Por fim, salve o index.html (Figura 1).

Localização do arquivo **index.html**:

- **mobile**: `src/main/mobileapp/www/index.html`
- **web**: `src/main/webapp/index.html`

Linha original: `<script src="node_modules/cronapi-js/dist/cronapi.min.js"></script>`
Linha alterada: `<script src="node-modules/cronapi-js/cronapi.js"></script>`

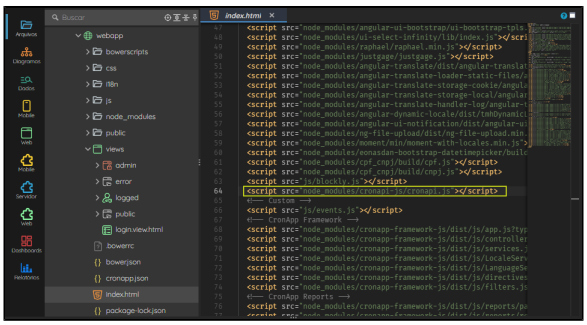

Figura 1 - Arquivo index.html alterado

Iniciar a depuração

Agora já podemos depurar os blocos cliente em nosso navegador e, para isso, vamos precisar criar um bloco de programação. Em nosso exemplo foi utilizado um bloco que recebe dois parâmetros (a, b), realiza a operação de multiplicação entre esses parâmetros, guardando o valor na variável **resultado** e, em seguida, altera o valor de um campo no formulário (Figura 1.1). Esses parâmetros provêm dos campos **Valor de a** e **Valor de b**, obtidos através do bloco Obter valor do Campo, por fim, altera o valor do campo **Resultado** com o resultado da operação no bloco através da ação no botão **Calcular** (Figura 1.2).

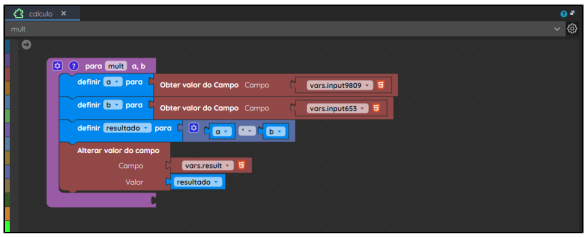

Figura 1.1 - Estrutura do Bloco Cliente

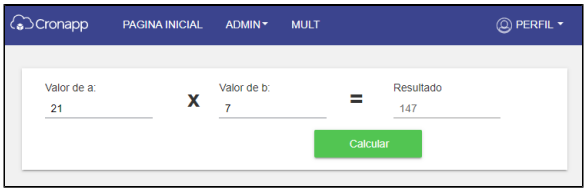

Figura 1.2 - Formulário que chama o bloco 'mult' da Figura 1.1

Para depurar o código, é necessário rodar o projeto e executá-lo em um navegador, como o Chrome. Aperte a tecla F12 para abrir a ferramenta de desenvolvedor do navegador, o DevTools (Figura 1.3).

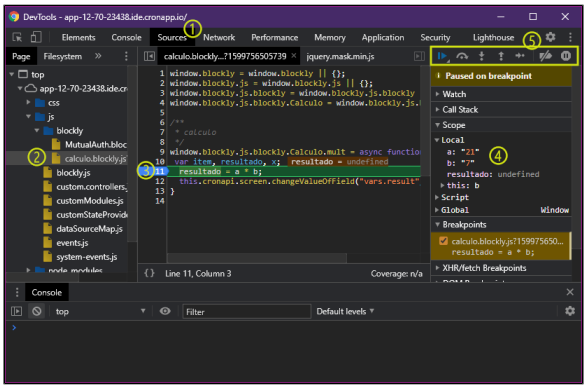

Figura 1.3 - *Debug* na ferramenta do desenvolvedor do Google Chrome (DevTools)

Para acessar a área de depuração, siga os passos abaixo:

1. Acesse a aba ***Sources***.
2. Clique em **Page > aplicação > js > blockly** e selecione o arquivo com a função (`<nome_do_arquivo>.blockly.js`).
3. Clique no número da linha que deseja adicionar o breakpoint.
4. Na janela da aplicação, execute a ação que chama a função selecionada (Figura 1.2). Você poderá visualizar o conteúdo das variáveis na aba lateral **Local**.
5. Use os botões de execução para avançar os passos do código:
    - **Executa o script**: executa todo o código restante sem a necessidade de depuração;
    - **Pular a próxima função**: ignora uma função e pula para o próximo passo;
    - **Pular para a próxima função**: ignora os comandos e pula para a próxima chamada de função;
    - **Sair da função atual**: sai da função corrente e avança para o próximo comando;

- **Próximo passo**: executa o próximo comando;
- **Desativar breakpoints**: desabilita todos os breakpoints da aplicação, e ignora-os quando as funções são chamadas;
- **Pausar em exceções**: deixa o código em modo *debug* caso alguma exceção seja encontrada.

Blocos servidor

Debug visual

Acesse o tópico **Debug visual** na documentação Bloco de programação para ver como depurar de forma *low-code*, diretamente pelos blocos de programação.

Nesse segundo exemplo vamos aprender a depurar blocos servidor. Esses blocos são depurados dentro do Cronapp, logo, o processo com esse bloco é mais simples. Em nosso exemplo foi utilizado um bloco que recebe dois parâmetros (a, b), realiza a operação de adição entre esses parâmetros, guardando o valor na variável **resultado** e, em seguida, altera o valor de um campo no formulário (Figura 2). Esses parâmetros provêm dos campos **Valor de a** e **Valor de b**, obtidos através do bloco Obter valor do Campo, por fim, altera o valor do campo **Resultado** com o resultado da operação no bloco através da ação no botão **Calcular** (Figura 2.1).

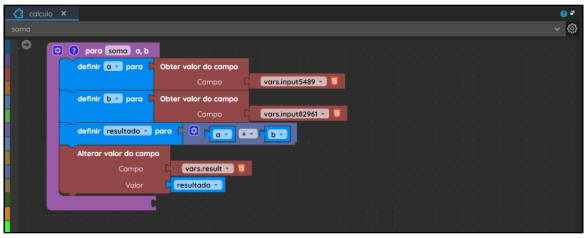

Figura 2 - Estrutura do Bloco Servidor

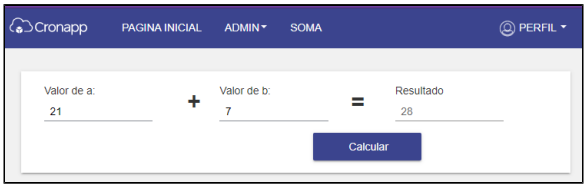

Figura 2.1 - Formulário linkado ao bloco de programação servidor

Depois de criar e salvar o bloco servidor, observe que são criados três arquivos: **`*.blocky`**, **`*.java`** e **`*.map`** (destaque 1 da Figura 2.2). Entretanto, para realizar a depuração, apenas o arquivo **`*.java`** é necessário, pois precisamos acessar diretamente o código fonte Java que é gerado pelo bloco de programação. Por isso, abra o arquivo **`*.java`** através do seu endereço, exemplo: **`src/main/java/blockly/Calculo.java`**.

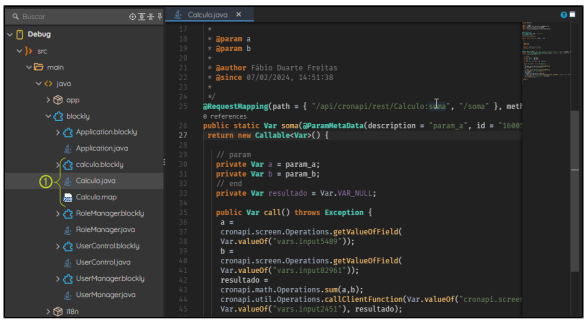

Figura 2.2 - Arquivos gerados após criar um bloco servidor

Após abrir o arquivo,

1. Clique no lado esquerdo do número da linha que deseja iniciar o *debug*, marcando-a com uma bola vermelha.
2. Execute na janela da aplicação a ação que chama a função selecionada (figura 2.1), isso fará com que a linha com breakpoint fique marcada em amarelo.
3. Passe o cursor do mouse sobre de uma variável ou objeto para exibir uma janela detalhando seu conteúdo, alguns elementos são expansíveis (Figura 2.3).
4. Para seguir depurando o código, utilize os botões destacados:
    - **Prosseguir execução**: executa todo o código restante sem a necessidade de depuração;
    - **Próximo passo**: executa o próximo comando;
    - **Entrar**: entra na próxima função;
    - **Sair**: sai da função corrente e avança para o próximo comando.

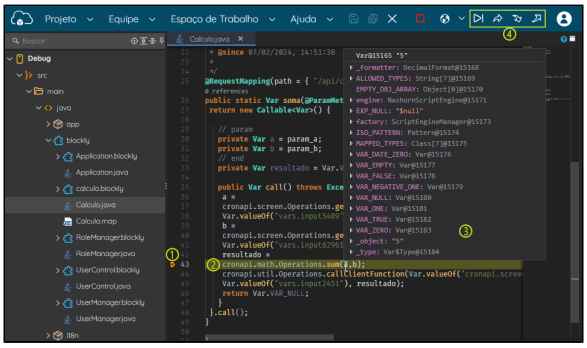

Figura 2.3 - Janela de *debug* detalhando a variável 'resultado' da figura 2