Entendendo o Components json

O **components.json** é um arquivo que possui uma estrutura do tipo JSON e usado pelo editor de *views* para exibir informações e configurações de suas propriedades, eventos e responsividades de cada componente (Figura 1). É através dele que se consegue adicionar mais recursos a um componente, permitindo sua customização sem a necessidade de configurá-lo via código.

Dentre alguns exemplos da necessidade do **components.json** no Cronapp, estão:

- Alteração do nome e ícone exibido no componente usando o editor de *views*;
- Personalização de elementos e atributos no *template* de forma visual;
- Internacionalização do controle;
- Inclusão de eventos a elementos do controle e outros.

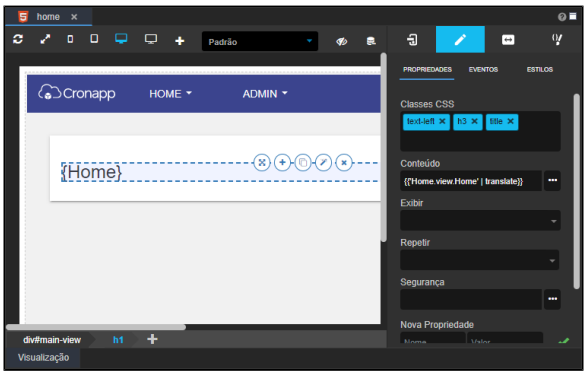

Figura 1 - Editor de *views* mostrando as propriedades do componente selecionado

Nesta página

- Estrutura do arquivo
- Propriedades do arquivo
  - Principais propriedades
  - Outras propriedades
  - attributesForPreview
    - values
  - childrenProperties
  - handleRules
    - rules
  - properties
    - Características

Conteúdo complementar

- Estilos - CSS
- Eventos dos componentes visuais
- Propriedades dos componentes visuais
- Formatação de máscaras na camada servidor
- Formatação de máscaras na camada cliente

Estrutura do arquivo

A estrutura do **components.json** é baseada em JSON (*JavaScript Object Notation* - Notação de Objetos JavaScript) que possui uma formatação simples e de fácil leitura e escrita, tanto para computadores quanto para humanos. Foi criado para facilitar a comunicação entre sistemas, independente da linguagem usada.

Os componentes padrões do Cronapp possuem seus próprios arquivos que terminam com a extensão **components.json** e seus diretórios mudam conforme o tipo de componente - ou seja, se o tipo é *web* ou *mobile* (Figura 2.1).

Os arquivos **components.json** só serão mostrados quando o modo avançado estiver habilitado.

- Componente *mobile*: **src/main/mobileapp/www/node_modules/cronapp-framework-mobile-js/components/**
- Componente *web*: **src/main/webapp/node_modules/cronapp-framework-js/components/**

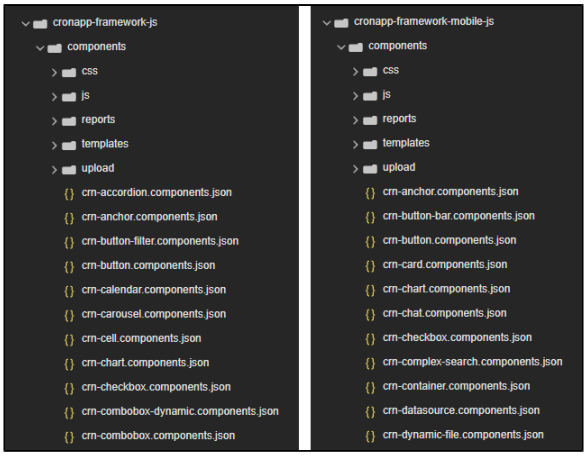

Figura 2.1 - Arquivos JSON de cada componente padrão do Cronapp, sendo *web* à direita e *mobile* à esquerda

O Cronapp permite criar componentes visuais e para cada novo componente, são criados os arquivos ***.template.html** e ***.components.json**. Ambos arquivos devem ficar no mesmo diretório (destaque na Figura 2.2).

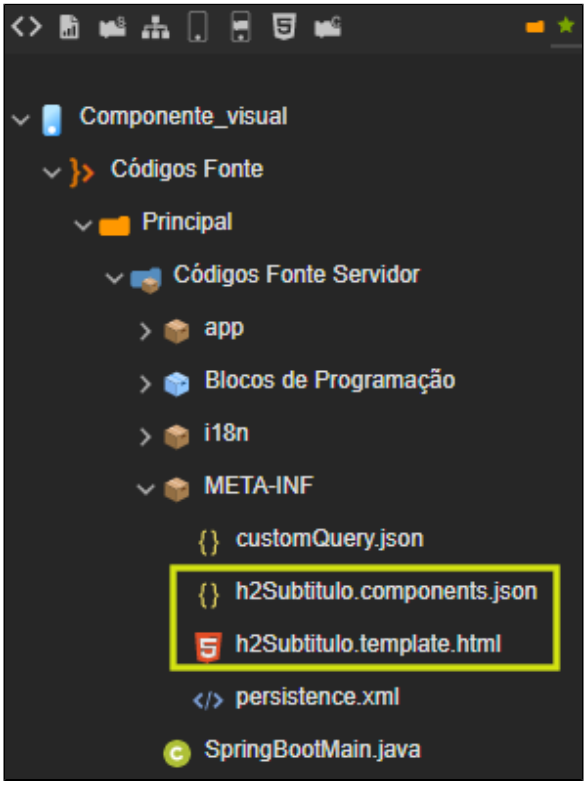

Figura 2.2 - Arquivos gerados ao criar componente visual

Propriedades do arquivo

As propriedades nesses arquivos mudam conforme o componente.

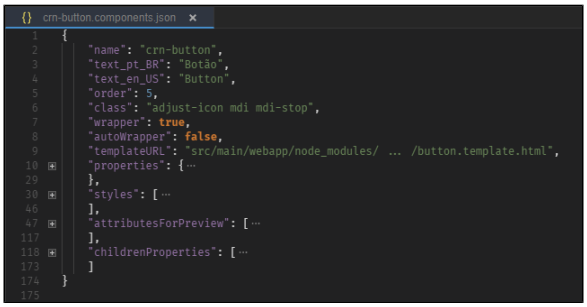

Figura 3 - Arquivo do componente visual Botão

Principais propriedades

São aquelas encontradas na maioria dos componentes.

Propriedade	Objetivo	Possíveis valores
name	Atributo id do componente HTML.	**Ex.**: "name": "crn-button",
text_pt_BR	Nome do componente que aparecerá na aba controle do editor de *views* quando a IDE estiver no idioma português (Brasil).	**Ex.**: "text_pt_BR": "Botão",
text_en_US	Nome do componente que aparecerá na aba controle do editor de *views* quando a IDE estiver no idioma inglês (EUA).	**Ex.**: "text_en_US": "Button",
class	Define o ícone do controle que aparecerá no editor de *views*. **OBS**.: Diferente da aplicação que permite o uso dos ícones Glyphicons, font awesome icon e Material Design Icons, os botões dos componentes no editor de *views* permite apenas o Material Design Icons.	**Ex.**: "class": "adjust-icon mdi mdi-stop",
templateURL	Endereço do template do componente na IDE.	**Ex.**: "templateURL": "src/main/webapp /node_modules /cronapp-framework-js /dist/components /templates/button. template.html",
properties	Lista as diretivas Angular, elementos HTML ou propriedades do Cronapp para permitir sua edição nas propriedades do componente no editor de *views*.	Consulte o tópico **properties**.
styles	Lista os elementos HTML que compõem o template, possibilitando selecioná-los e personaliza-los no editor de *views*. Essa seleção pode ser feita utilizando as subabas dentro das abas **Propriedades** no editor de *views* ou selecionando o componente na tela, abrindo o ícone do cadeado e selecionando novamente os elementos internos do componente.	Para cada elemento informado na lista: • **selector**: informa qual elemento HTML será selecionado. Se o template possuir dois elementos HTML iguais, é possível utilizar o **selector** do CSS para referenciá-lo; • **text_pt_BR**: internacionalização em português do Brasil para o rótulo da subaba que aparecerá em **propriedades** e **eventos** no editor de views referente ao elemento HTML selecionado; • **text_en_US**: internacionalização em inglês do EUA para o rótulo da subaba que aparecerá em **propriedades** e **eventos** no editor de *views* referente ao elemento HTML selecionado. **Ex.**: "styles": [{ "selector": "div# {id} button", "text_pt_BR": "Plano de fundo do botão", "text_en_US": "Button background" }],

description **description_pt_BR** **description_en_US**	Define o *tooltip* que será exibido quando o cursor do mouse estiver posicionado sobre o botão do componente visual na paleta de controles do Editor de views. É possível definir um texto para cada uma das línguas disponíveis (português ou inglês) ou usar um valor genérico que será exibido independente da língua.	É possível utilizar uma opção ou todas. • **description**: valor padrão, será utilizado para definir o valor para o idioma português (pt_BR) e inglês (en_US), caso não possua; • **description_en_US**: valor usado quando a IDE estiver com o idioma em inglês; • **description_pt_BR**: valor usado quando a IDE estiver com o idioma em português. Essa opção é desnecessária ao utilizar a opção **description**. **Ex.**: "description": "Exibição de gráfico com valores da fonte de dados", "description_en_US": "Graph display with data source values",
category **category_pt_BR** **category_en_US**	Define quais categorias da paleta de controle serão exibidos o botão do componente visual. Não é obrigado que um componente esteja associado a uma categoria, mas caso isso ocorra, ele será listado apenas na categoria "Todos". Existem algumas categorias padronizadas do Cronapp (lista abaixo), porém é possível criar uma nova categoria apenas informando o novo nome na lista. Valores padrão: 1. **INPUTS**: Controles de entrada de dados, • en_US: Inputs; • pt_BR: Entradas 2. **BUTTONS**: Controles com botões, • en_US: Buttons; • pt_BR: Botões 3. **PLUGINS**: Controles proveniente de plugins, • en_US: Plugins • pt_BR: Plugins 4. **LAYOUTS**: Controles que constam layout. • en_US: Layouts • pt_BR: Layouts 5. **FORMS**: Controles que formam formulários. • en_US: Forms • pt_BR: Formulários 6. **GRIDS**: Grades, listas e Tabelas. • en_US: Grids and Tables • pt_BR: Grades e Tabelas 7. **LISTS**: Controles com listas. • en_US: Lists • pt_BR: Listas 8. **COMBOS**: Comboxes fixos e dinâmicos. • en_US: Combos • pt_BR: Combos 9. **TEXTS**: Textos. • en_US: Texts • pt_BR: Textos	É possível utilizar uma opção ou todas. • **category**: lista padrão, será utilizado para informar a categoria nos idiomas português (pt_BR) e inglês (en_US), caso não possua; • **category_en_US**: lista usada quando a IDE estiver com o idioma em inglês; • **category_pt_BR**: lista usada quando a IDE estiver com o idioma em português. Essa opção é desnecessária ao utilizar a opção **description**. **Ex.**: `"category": [ "BUTTONS", "FORMS", "Minha nova categoria" ], "category_en_US": [ "BUTTONS", "FORMS", "My new category" ],`

Outras propriedades

A utilização dessas propriedades varia de acordo com o componente.

Propriedade	Objetivo	Possíveis valores
attributesForPreview	Lista de propriedades customizadas nos atributos CSS dos componentes.	Consulte o tópico **attributesForPreview**.
autoWrapper	Adiciona uma tag em torno do template do componente, facilitando a modificação da diagramação.	• **true**: Permite tratamento diferenciado entre o componente e a . • **false**: Não permite tratamento diferenciado entre o componente e a e é o valor *default* quando a propriedade não é informada. **Ex.:** `"autoWrapper": false,`
childrenProperties	Lista as diretivas Angular, elementos HTML ou exclusivas do Cronapp para permitir sua edição nas propriedades dos subcomponentes (filhos) do componente no editor de *views*.	Consulte o tópico **childrenProperties** .
dependences	Possui uma lista (editor) com os endereços de arquivos CSS específica para esse controle.	**Ex.:** `"dependences": { "editor": [ "node_modules /cronapp-framework-js /components/css/multiselect-editor.css" ]},`
designTimeHTMLURL	Sinaliza qual o caminho do html que será exibido no *designtime*.	**Ex.:** `"designTimeHTMLURL": "src /main/webapp/node_modules /cronapp-framework-js/dist /components/templates/cron-scheduler.designtime.html",`
designTimeSelector	Sinaliza qual diretiva ele irá usar.	**Ex.:** `"designTimeSelector": "cron-scheduler",`
forcedGroup	Força o agrupamento dos componentes HTML no template, permitindo a seleção de todos os subcomponentes de uma só vez.	• **true**: Agrupa todos os subcomponentes, necessário realizar o desbloqueio para poder selecionar cada subcomponente. • **false**: Permite a seleção de cada subcomponente sem a necessidade do desbloqueio e é o valor *default* quando a propriedade não é informada. **Ex.:** `"forcedGroup": true,`
handleRules	Define características para algumas propriedades e regras para elementos do componente.	Consulte o tópico **handleRules**.
mask-placeholder	Informa qual componente HTML receberá o texto do campo Sugestão no editor de *views*.	Possui em seu interior o valor "selector" onde será informado o componente HTML. **Ex.:** `"name": "mask-placeholder"`
onDrop	Abre a janela de configuração especificada quando o componente é arrastado para a área de edição da view.	Nome do atributo. **Ex.:** `"onDrop": "openEditor",`
onDoubleClick	Abre a janela de configuração especificada quando o componente recebe um duplo click na área de edição da view.	Nome do atributo. **Ex.:** `"onDoubleClick": "openEditor",`
pallete	Oculta ou exibe o controle no editor de *views*.	• **true**: exibe o controle no editor de *views* e é o valor *default* quando a propriedade não é informada. • **false**: oculta o controle no editor de *views*. **Ex.:** `"pallete": false,`

rows	Define o tamanho do campo em número de linhas.	**Ex.**: `"rows": { "displayName": "rows" },`
template	Template HTML do componente. Por este ser um arquivo JSON, será necessário utilizar caracteres de escape para utilizar alguns caracteres. **Exemplo**: “\n”e “\””	**Ex.**: `<i class=\"fa fa-star\"/>`
text	Nome do componente que aparecerá no editor de *views* quando não houver internacionalização (propriedades: text_en_US e text_pt_BR).	**Ex.**:`"text": "File Upload Button",`
wrapper	Adiciona uma tag em torno do template do componente.	• **true**: adiciona uma tag e é o valor *default* quando a propriedade não é informada. • **false**: não adiciona uma tag . **Ex.**: `"wrapper": false,`

attributesForPreview

Lista de propriedades customizadas para os atributos CSS dos componentes. Veja a tabela de opções dessa propriedade após o código de exemplo.

crn-text.components.json

```
"attributesForPreview": [
  {
    "name": "xattr-position",
    "type": "btngroup",
    "target": "class",
    "values": [
      {
        "key": "text-left",
        "value": "Standard",
        "icon": "mdi mdi-ray-start"
      }
    ]
  }
]
```

Propriedade	Objetivo	Possíveis valores
name	Nome da propriedade que terá suas características alteradas.	**Ex.**: `"name": "xattr-theme",`
type	Define a forma que será exibida as opções da propriedade. Normalmente são utilizados o grupo de botões ou seletor de itens.	• **btngroup**: para exibir as propriedades em um grupo de botões alinhados. Permite a utilização de ícones nos botões. • **options**: para exibir as propriedades em um seletor de itens (*dropdown*). Permite a utilização de ícones, cores ou ambos. **Ex.**: `"type": "btngroup",`
target	Informa o atributo ou elemento que será customizado. Normalmente utilizado para adicionar **class** ou **style**.	**Ex.**: `"target": "class",`
values	Lista das opções da propriedade.	Consulte o tópico **values**.

values

Lista das opções da propriedade **`values`** do **`attributesForPreview`**.

Propriedade	Objetivo	Possíveis valores (exemplos)
`key`	Classe CSS que será adicionado no template após o usuário escolher o value.	**`Ex.`**`: "key": "text-left",`
`value`	Nome da opção que será exibida para o usuário.	**`Ex.`**`: "value": "Standard",`
`icon`	Define o ícone que será exibido ao lado do nome da opção.	**`Ex.`**`: "icon": "mdi mdi-ray-start"`
`color`	Exibe um quadrado com a cor especificada ao lado do nome da opção. Observação: Utilizar apenas quando **`type`** é definido como **`options`**.	**`Ex.`**`: "color": "#fff"`

childrenProperties

Responsável por criar propriedades distintas entre subcomponentes dentro de um componente. Veja a tabela de opções dessa propriedade após o código de exemplo.

Trecho de código do arquivo crn-dynamic-menu.components.json

```
"childrenProperties": [
  {
    "name": "options",
    "selector": "cron-dynamic-menu",
    "displayName_pt_BR": "Configuração",
    "displayName_en_US": "Configuration",
    "type": "text",
    "order": 1,
    "mandatory": true
  }
]
```

Propriedade	Objetivo	Possíveis valores
`displayName_en_US`	Nome do campo que aparecerá na aba propriedades do editor de *views* quando a IDE estiver no idioma inglês (EUA).	**`Ex.`**`:"displayName_en_US": "Title",`
`displayName_pt_BR`	Nome do campo que aparecerá na aba propriedades do editor de *views* quando a IDE estiver no idioma português (BR).	**`Ex.`**`: "displayName_pt_BR": "Título",`
`editExpression`	Exibe o ícone **Editar expressão** ao lado do nome do campo e ao clicar é aberta uma caixa de texto para a edição da expressão ou inserção de Fonte de dados.	• **`true`**: ativa a opção; • **`false`**: desativa a opção e é o valor *default* quando a propriedade não é informada. **`Ex.:`**`"editExpression": false,`
`mandatory`	Define uma propriedade como preenchimento obrigatório, adicionando contorno vermelho à propriedade quando vazia.	• **`true`**: define como obrigatória. • **`false`**: define como não obrigatória. **`Ex.:`**`"mandatory": true,`

name	Informa o atributo do elemento HTML que será manipulado em um campo do editor de *views*.	**Ex.**: `"name": "ng-model",`
onDisplay	Executa comando em JavaScript para atualizar o componente na tela sempre que uma alteração for feita.	Com o comando javascript:function() { /* função JavaScript */ } **Ex.**: `"onDisplay": "javascript: function() { return arguments[0]. replace('vars.',''); }"`
onSave	Executa comando em JavaScript para salvar sempre que uma opção for selecionada ou ação tomada no componente.	Com o comando javascript:function() { /* função JavaScript */ } **Ex.**: `"onSave": "javascript: function() { var result = arguments [0]; if (result.indexOf('.') == -1 && result.trim().length > 0 ) { result = 'vars.'+result; } return result; }"`
selector	Informa qual atributo do elemento HTML será selecionado para manipulação.	**Ex.**: `"selector": "ui-select",`
type	Define a forma de como o subcomponente irá se comportar como permitir apenas texto, seletor ou importação de fonte de dados.	• **type**: campo seletor *input* (ex: *text*, *number* , *date*, *email*, *money*, *telephone*, *time*, *url*, etc); • **text**: texto normal; • **event**: direciona para a aba de "eventos"; • **logic**: booleano (sim ou não); • **content**: direciona para a aba de "conteúdo"; • **icon**: abre o modal de adicionar ícone; • **pageDatasource**: campo seletor de fonte de dados (para funcionar é necessário inserir uma fonte de dados); • **allDatasourceFields**: captura as variáveis da fonte de dados (para funcionar é necessário inserir uma fonte de dados); • **projectResource**: abre um modal que busca os arquivos da extensão sinalizado no "*resourceType*", como imagem. **Ex.:** `"type": "event",`

handleRules

Define características para algumas propriedades e regras para elementos do componente. Veja abaixo a tabela de opções dessa propriedade.

Trecho de código do arquivo crn-grid.components.json

```
"handleRules": {
    "canOrder": true,
    "canActivate": true,
    "canIncrement": true,
    "rules": [
      {
        "activeSelector": ".table-header th.active",
        "activeClass": "active",
        "sourceHTML": "<th class=\"k-header\" data-container=\"true\"
rowspan=\"\" colspan=\"\"><div data-container=\"true\" data-replace=\"
true\">New Title</div></th>",
        "targetSelector": ".table-header"
      }
    ]
  }
```

Propriedade	Objetivo	Possíveis valores / exemplos
`canActivate`	Ativa ou desativa as regras.	• **`true`**: ativa o handleRules do componente. • **`false`**: desativa o handleRules do componente. **`Ex.:`** `"canActivate": true,`
`canIncrement`	Exibe o ícone **adicionar** ao lado do nome do campo e ao clicar é inserido outro item na lista. Exemplo: outra coluna em uma tabela ou outro item em uma lista de opções.	• **`true`**: insere o ícone **adicionar**; • **`false`**: oculta o ícone **adicionar** e é o valor *default* quando a propriedade não é informada. **`Ex.:`** `"canIncrement": true,`
`canOrder`	Define a possibilidade de ordenação dos itens.	• **`true`**: possibilita ordenação dos itens e é o valor *def ault* a propriedade não é informada; • **`false`**: impede a ordenação dos itens. **`Ex.:`** `"canOrder": true,`
`canToggle`	Mostra o ícone **exibir** ao lado do nome do campo e esse ícone recolhe ou expande os itens ou submenus desse controle no editor de views, permitindo visualizar e selecionar seu conteúdo para personalização. Exemplo: caixa de seleção ou menu.	• **`true`**: insere o ícone **exibir**; • **`false`**: oculta o ícone **exibir** e é o valor *default* a propriedade não é informada. **`Ex.:`** `"canToggle": true,`
`rules`	Define algumas regras para alteração do template do componente a partir de determinadas ações.	Consulte o tópico `rules` .

rules

Lista de as regras de controle do **`handleRules`**.

Propriedade	Objetivo	Possíveis valores / exemplos
`accordionTabClicked`	Específico do componente acordeão, usado para acionar a ação de expandir ou recolher seus itens.	**`Ex.:`** `"accordionTabClicked": "panel-collapse. collapse",`
`activeClass`	Ativa uma classe específica apontada anteriormente.	**`Ex.:`** `"activeClass": "in",`
`activeSelector`	Seleciona o atributo que sofrerá a alteração.	**`Ex.:`** `"activeSelector": "div.panel-collapse.collapse. in",`

sourceHTML	Define trecho do HTML que será acrescido ou retirado do *template*, caso a regra seja atendida.	**Ex.**: "sourceHTML": " ",
targetSelector	Seleciona um elemento HTML ou classe CSS para sofrer a alteração,	**Ex.**: "targetSelector": "div.panel-group",
toggleClass	Adicionar ou retirar uma classe CSS.	**Ex.**: "toggleClass": "open",
toggleSelector	Adicionar ou retirar um elemento seletor.	**Ex.**: "toggleSelector": ".dropdown",

properties

Responsável por criar propriedades e eventos que afetam todo o componente (incluindo os subcomponentes). Veja abaixo a tabela de opções dessa propriedade.

Trecho de código do arquivo crn-title.components.json

```
    "properties": {
        "xattr-fullsize": {
            "displayName_en_US": "Width",
            "displayName_pt_BR": "Largura"
        },
        "xattr-theme": {
            "displayName_en_US": "Theme",
            "displayName_pt_BR": "Tema"
        },
        "disabled": {
            "displayName_en_US": "Disabled",
            "displayName_pt_BR": "Desabilitado"
        },
        "class": {
            "order": 9999
        },
        "id": {
            "order": 5
        }
    },
```

Propriedade	Campo no editor de *views*	Objetivo
auto-refresh	Atualizar Automaticamente (ms)	Define o tempo, em milissegundos, que o datasource atualizará automaticamente.
class	Classes CSS	Adiciona classes CSS já criadas.
checkrequired	Validar Campos Requeridos	Insere ou atualiza os dados somente se todos os campos marcados como requeridos estiverem preenchidos.
crn-datasource	Fonte de dados	Define a fonte de dados do componente.
cronapp-filter-autopost	Postagem Automática	Realiza consulta ou ação a cada letra digitada no campo.
cronapp-filter-caseinsensitive	Case Insensitive	Define se o campo diferenciará letras minúsculas e maiúsculas.

cronapp-filter-operator	Operador	Define operadores relacionais e de igualdade. (=, <, >, <= e >=)
dependent-by	Depende de	Cria uma dependência do componente para a exibição de conteúdo. **Ex.**: As informações sobre um usuário só irão aparecer na tela quando um dos usuários for selecionado.
dependent-lazy-post	Dependente de postagem sob demanda	Cria uma dependência do componente com a fonte de dados informada para a inserção de conteúdo. Nesse campo deve ser informado o objeto independente em relação ao campo "dependente de postagem de campo sob demanda". **Ex.**: As regras para um usuário só serão submetidas se existir um usuário.
dependent-lazy-post-field	Dependente de postagem de campo sob demanda	Cria uma dependência do componente com a fonte de dados informada para a inserção de conteúdo. Nesse campo deve ser informado o objeto ou id que ficará dependente. **Ex.**: As regras para um usuário só serão submetidas se existir um usuário.
entity	Origem de dados	Define qual a fonte de dados que irá alimentar o componente.
href	Referência	Define uma URL para gerar um *link*.
id	Identificador	Atributo que especifica um ID exclusivo para o componente. Por padrão um id é gerado automaticamente e atribuído ao componente.
keys	Chaves	Define quais parâmetros serão usados como chave na fonte de dados.
lazy	Postergar Carga	Alimenta a fonte de dados somente após uma ação do usuário.
load-data-strategy	Estratégia para Carregar Dados	Define as condições de carregamento dos dados
mask	Máscara	Lista pré-definida de máscaras. **Ex.**: CEP, CPF, data, telefone etc.
mask-placeholder	Sugestão de máscara	Possibilita adição de novas máscaras customizadas.
ng-click	Ao clicar	Executa evento ao clicar (AngularJS).
ng-dblclick	Ao duplo clicar	Executa evento ao duplo clicar (AngularJS).
ng-init	Ao inciar	Executa evento ao iniciar (AngularJS).
ng-model	Valor	Propriedade AngularJS que pode ser usada para obter e alimentar o conteúdo pelo bloco de programação.
ng-mouseenter	Ao entrar com o mouse	Executa evento ao entrar com o mouse (AngularJS).
ng-mouseleave	Ao sair com o mouse	Executa evento ao sair com o mouse (AngularJS).
ng-mousemove	Ao mover mouse	Executa evento ao mover mouse (AngularJS).
on-after-create	Após criar	Executa um evento após a criação algo.
on-after-delete	on-after-delete	Executa evento após ação de deletar algo.
on-after-fill	Após preencher	Executa evento após preenchimento de um campo.

on-after-update	Após atualizar	Executa evento após atualização de algo.
on-before-create	Antes de criar	Executa evento antes de criar algo.
on-before-delete	Antes de deletar	Executa evento antes de deletar algo.
on-before-update	Antes de atualizar	Executa evento antes de atualizar algo.
onclick	Ao clicar	Executa evento ao clicar no componente (JavaScript).
on-error	Ao errar	Executa um evento quando ocorrer um erro no carregamento de algum arquivo ou mídia (JavaScript).
onmouseover	Ao clicar	Executa evento no precionar do clique do botão (JavaScript).
parameters	Mestre Detalhe	Vincula um ou mais atributos de uma fonte de dados com outra fonte de dados.
parameters-null-strategy	Estratégia de Cláusula Nula	Define as condições de carregamento dos dados.
qr	QR Code	Exclusivo para o controle QR Code.
src	Origem	Define campo para carregamento de imagem.
target	Alvo	Seleciona em qual janela será aberta o link.
xattr-fullsize	Largura	Determina se o componente ocupará a linha inteira ou não. Suas opções padrões são **`inline`** ou **`block`** e definidas no `attributesForPreview`.
xattr-position	Posição	Determina em qual posição o componente ficará alinhado. Suas opções padrões são à esquerda (*standard*), centro (*middle*) ou direita (*right*) e definidas no `attributesForPreview`.
xattr-size	Tamanho	Determina o tamanho do componente na tela. Suas opções padrões são *standard*, *larger* (btn-lg) e *small* (btn-sm) e definidas no `attributesForPreview`.
xattr-theme	Tema	Determina o tema do componente. Suas opções padrões são *default* , *primary*, *success*, *info*, *warning*, *danger* e *link* e definidas no `attributesForPreview`.
xattr-type	Style Type	Determina o estilo do botão. Suas opções padrões são *default*, *clear* (button-clear) e *outline* (button-outline) e definidas no `attributesForPreview`.
disabled	Desativado	Determina se o botão será desabilitado ou não.

Acesse a documentação Eventos dos componentes visuais para mais detalhes sobre cada evento.

Características

Define as características das propriedades e evento do controle de qualquer uma das propriedades informadas na tabela de **`properties`**.

Propriedade	Objetivo	Possíveis valores
displayName_en_US	Nome do rótulo que aparecerá na aba propriedades do editor de views quando a IDE estiver no idioma inglês (EUA).	**Ex.**: `"displayName_en_US": "Configuration",`
displayName_pt_BR	Nome do rótulo que aparecerá na aba propriedades do editor de views quando a IDE estiver no idioma português (BRA).	**Ex.**: `"displayName_pt_BR": "Configurações",`

`mandatory`	Define uma propriedade como preenchimento obrigatório, adicionando contorno vermelho à propriedade quando vazia.	• **`true:`** define como obrigatória; • **`false:`** define como não obrigatória. **`Ex.:`** `"mandatory": true,`
`options`	Define as chaves e os valores (tanto em inglês quanto em português) que serão exibidas quando a propriedade for do tipo lista.	**`Ex.:`** `"options": [{ "key": "default", "value_en_US": "Default", "value_pt_BR": "Padrão" }],`
`removable`	Exibe o ícone remover propriedade ao lado do nome do campo e ao clicar, o campo será removido dentre as propriedades ou eventos. **OBS**.: É possível chamar novamente a propriedade através do campo **nova propriedade**.	• **`true`**: insere o ícone; • **`false`**: oculta o ícone e é o valor *default* a propriedade não é informada. **`Ex.:`** `"removable": true,`
`resourceType`	Define qual a extensão de arquivo será utilizada.	Permite imagens do tipo jpeg, gif ou png. **`Ex.:`** `"resourceType": "image/jpeg,image/gif,image/png,image/svg+xml",`