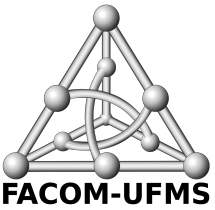
Universidade Federal de Mato Grosso do Sul (UFMS)
Faculdade de Computação (FACOM)



Documentação Técnica das

Regras JSONForms (Rule)

Yan G. Santana

Campo Grande/MS
2024

Introdução

Este documento visa fornecer uma documentação técnica clara e concisa sobre as regras e funcionalidades avançadas do JSONForms, com foco em exemplos práticos e cenários de teste. A documentação segue as diretrizes de Technical Writing do Google, buscando clareza, precisão e facilidade de compreensão para desenvolvedores e usuários.

O JSONForms é uma ferramenta poderosa para a criação de formulários dinâmicos a partir de JSON Schemas. Compreender suas regras é essencial para construir interfaces de usuário robustas e eficientes. Esta documentação abordará as seguintes regras e conceitos:

- Visibilidade Condicional (***rule*** com ***effect: HIDE/SHOW***)
- Múltiplas Condições (***allOf*** , ***anyOf***)
- Habilitação Condicional Explícita (***rule*** com ***effect: ENABLE***)

Cada seção apresentará a regra, sua aplicação em um JSON Schema e UI Schema, e um cenário de teste detalhado para demonstrar seu comportamento. O objetivo é fornecer um recurso abrangente que complemente a documentação oficial do JSONForms, preenchendo lacunas e oferecendo exemplos práticos para facilitar o aprendizado e a implementação.

1. Visibilidade Condicional Simples (***rule*** com ***effect: HIDE/SHOW***)

A regra de visibilidade condicional no JSONForms permite controlar a exibição de elementos do formulário (campos, grupos, layouts) com base em condições definidas no JSON Schema. Utilizando a propriedade *rule* dentro do *uischema* , é possível especificar um *effect* (como *HIDE* para ocultar ou *SHOW* para exibir) e uma *condition* que, quando verdadeira, aplica o efeito. Essa funcionalidade é crucial para criar formulários dinâmicos e responsivos, que se adaptam às entradas do usuário, melhorando a experiência e a relevância dos campos apresentados.

1.1 Cenário de Teste: Visibilidade Condicional Simples

Este cenário demonstra como um campo de telefone pode ser exibido ou ocultado dinamicamente com base na resposta do usuário a uma pergunta simples: "Deseja receber ligações?". O objetivo é garantir que o campo de telefone só seja visível e, portanto, preenchível, quando o usuário explicitamente indicar que deseja receber ligações.

JSON Schema

```
{
  "type": "object",
  "properties": {
    "nome": {
      "type": "string",
      "title": "Nome Completo",
      "minLength": 3
    },
    "receberLigacoes": {
      "type": "string",
      "title": "Deseja receber ligacoes?",
      "enum": [
        "Sim",
        "Nao"
      ],
      "default": "Nao"
    },
    "telefone": {
      "type": "string",
      "title": "Telefone",
      "pattern": "^\\([0-9]{2}\\)
[0-9]{4,5}-[0-9]{4}$"
    }
  },
  "required": [
    "nome",
    "receberLigacoes"

}
```

UI Schema

```
{
  "type": "VerticalLayout",
  "elements": [
    {
      "type": "Control",
      "scope": "#/properties/nome",
      "label": "Nome Completo"
    },
    {
      "type": "Control",
      "scope": "#/properties/receberLigacoes",
      "label": "Deseja receber ligacoees?"
    },
    {
      "type": "Control",
      "scope": "#/properties/telefone",
      "label": "Telefone",
      "rule": {
        "effect": "HIDE",
        "condition": {
          "scope":
"#/properties/receberLigacoes",
          "schema": {
            "not": {
              "const": "Sim"
            }
          }
        }
      }
    }
  ]
}
```

1.2 Explicação da Solução

No *uischema* , a propriedade *rule* é aplicada ao controle do campo *telefone* . Esta regra define que o campo deve ser *HIDE* (ocultado) quando a *condition* for verdadeira. A condição, por sua vez,, verifica o *scope* *#/properties/receberLigacoes* e utiliza *schema: { "not": { "const": "Sim" } }* . Isso significa que o campo telefone será ocultado se o valor de *receberLigacoes* não for "Sim". Consequentemente, o campo *telefone* só será exibido quando *receberLigacoes* for exatamente "Sim".

1.3 Passos para Teste

1. **Inicie o formulário:** Observe que o campo "Telefone" não está visível inicialmente, pois o valor padrão de "Deseja receber ligações?" é "Não".
2. **Altere a opção:** Selecione "Sim" para "Desejar receber ligações?". O campo "Telefone" deve aparecer imediatamente.
3. **Preencha o telefone:** Insira um número de telefone válido no formato *(XX) XXXX-XXXX* ou *(XX) XXXXX-XXXX* .
4. **Altere de volta:** Selecione "Não" para "Desejar receber ligações?". O campo "Telefone" deve desaparecer.
5. **Verifique a validação:** Tente enviar o formulário com o campo "Telefone" visível, mas vazio ou com formato inválido, e observe as mensagens de erro.

Este teste validaa a capacidade do JSONForms de controlar a visibilidade de campos de forma condicional, garantindo que apenas informações relevantes sejam solicitadas ao usuário em um determinado momento.

2. Múltiplas Condições (***allOf*** *,* ***anyOf***)

No JSON Schema, é possível combinar múltiplas condições lógicas para controlar a visibilidade ou o comportamento de elementos do formulário. As palavras-chave *allOf* e *anyOf* são usadas para expressar essas combinações:

- ***allOf*** : Significa que todas as subcondições especificadas devem ser verdadeiras para que a condição principal seja satisfeita. É o equivalente lógico de um operador AND.

- ***anyOf*** : Significa que pelo menos uma das sub-condições especificadas deve ser verdadeira para que a condição principal seja satisfeita. É o equivalente lógico de um operador OR.

Essas combinações são poderosas para criar lógicas de formulário complexas, onde a exibição de um campo depende de várias seleções ou estados de outros campos.

2.1 Cenário de Teste: Teste 3 - Múltiplas Condições

Este teste explora como um campo pode ser exibido apenas quando múltiplas condições são satisfeitas simultaneamente. O objetivo é mostrar o campo "Detalhes do Evento" somente se o usuário desejar participar de eventos E o tipo de evento selecionado for "Presencial".

JSON Schema

```
{
  "type": "object",
  "properties": {
    "nome": {
      "type": "string",
      "title": "Nome Completo",
      "minLength": 3
    },
    "participarEventos": {
      "type": "string",
      "title": "Deseja participar de
eventos?",
      "enum": [
        "Sim",
        "Nao"
      ],
      "default": "Nao"
    },
    "tipoEvento": {
      "type": "string",
      "title": "Tipo de Evento",
      "enum": [
        "Presencial",
        "Online",
        "Hibrido"
      ],
      "default": "Online"
    },
```

UI Schema

```
{
  "type": "VerticalLayout",
  "elements": [
    {
      "type": "Control",
      "scope": "#/properties/nome",
      "label": "Nome Completo"
    },
    {
      "type": "Control",
      "scope":
"#/properties/participarEventos",
      "label": "Deseja participar de
eventos?"
    },
    {
      "type": "Control",
      "scope": "#/properties/tipoEvento",
      "label": "Tipo de Evento"
    },
    {
      "type": "Control",
      "scope":
"#/properties/detalhesEvento",
      "label": "Detalhes do Evento",
      "rule": {
        "effect": "SHOW",
        "condition": {
```

```
    "detalhesEvento": {
      "type": "string",
      "title": "Detalhes do Evento",
      "description": "Informacoes sobre
local, horario e programacao",
      "minLength": 10
    },
    "endereco": {
      "type": "string",
      "title": "Endereco para Eventos
Presenciais",
      "minLength": 10
    }
  },
  "required": [
    "nome",
    "participarEventos",
    "tipoEvento"
  ]
}
```

```
        "schema": {
          "allOf": [
            {
              "properties": {
                "participarEventos": { "const":
"Sim" }
              },
              "required": ["participarEventos"]
            },
            {
              "properties": {
                "tipoEvento": { "const":
"Presencial" }
              },
              "required": ["tipoEvento"]
            }
          ]
        }
      }
    }
    },
    {
      "type": "Control",
      "scope": "#/properties/endereco",
      "label": "Endereco para Eventos
Presenciais"
    }
  ]
}
```

2.2 Explicação da Solução

Para o campo *detalhesEvento* , a *rule* no *uischema* utiliza *effect: SHOW* combinado com *allOf* na *condition* . Isso significa que o campo só será visível se o valor de *participarEventos* for "Sim" E o valor de *tipoEvento* for "Presencial". SSe qualquer uma dessas condições não for atendida, o campo permanecerá oculto.

O campo *endereco* utiliza uma regra mais simples, sendo visível apenas quando *tipoEvento* for "Presencial".

2.3 Passos para Teste

1. **Inicie o formulário:** Observe que o campo "Detalhes do Evento" e

"Endereço para Eventos Presenciais" não estão visíveis.

2. **Selecione "Sim" para "Deseja participar de eventos?":** O campo "Detalhes do Evento" ainda não deve aparecer, pois a segunda condição (`tipoEvento` = "Presencial") não foi atendida.
3. **Mantenha "Sim" e selecione "Presencial" para "Tipo de Evento":** Agora, o campo "Detalhes do Evento" deve aparecer. O campo "Endereço para Eventos Presenciais" também deve aparecer.
4. **Altere "Tipo de Evento" para "Online":** O campo "Detalhes do Evento" e "Endereço para Eventos Presenciais" devem desaparecer.
5. **Altere "Deseja participar de eventos?" para "Não":** O campo "Detalhes do Evento" deve desaparecer, mesmo que "Tipo de Evento" seja "Presencial".

Este teste valida a capacidade do JSONForms de lidar com múltiplas condições para controlar a visibilidade de campos, o que é fundamental para formulários complexos e lógicas de negócio.

3. Habilitação Condicional Explícita (***effect: ENABLE***)

A regra de habilitação condicional no JSONForms utiliza o `*effect: ENABLE*`, um recurso presente no *core* da ferramenta, mas pouco explorado em exemplos da documentação oficial. Diferente do `DISABLE`, que desabilita o campo quando a condição é verdadeira, o `ENABLE` inverte a lógica: o campo só é explicitamente habilitado quando a condição é satisfeita.

Esse tipo de regra é útil em cenários nos quais a segurança, validação ou o fluxo de preenchimento exigem que o campo permaneça bloqueado por padrão, sendo liberado apenas quando determinado critério for atendido.

3.1 Cenário de Teste: Habilitação Condicional Explícita

Este cenário demonstra como um campo de e-mail pode permanecer desabilitado por padrão e ser habilitado apenas quando o usuário aceita receber comunicações. O objetivo é garantir que os dados de contato só sejam fornecidos se houver consentimento explícito.

JSON Schema

```
{
  "type": "object",
  "properties": {
    "nome": {
      "type": "string",
      "title": "Nome Completo",
      "minLength": 3
    },
    "aceitaComunicacoes": {
      "type": "string",
      "title": "Deseja receber
comunicações?",
      "enum": [
        "Sim",
        "Não"
      ],
      "default": "Não"
    },
    "email": {
      "type": "string",
      "title": "E-mail"
    }
  },
  "required": [
    "nome",
    "aceitaComunicacoes"
  ]
}
```

UI Schema

```
{
  "type": "VerticalLayout",
  "elements": [
    {
      "type": "Control",
      "scope": "#/properties/nome",
      "label": "Nome Completo"
    },
    {
      "type": "Control",
      "scope":
"#/properties/aceitaComunicacoes",
      "label": "Deseja receber
comunicações?"
    },
    {
      "type": "Control",
      "scope": "#/properties/email",
      "label": "E-mail",
      "rule": {
        "effect": "ENABLE",
        "condition": {
          "scope":
"#/properties/aceitaComunicacoes",
          "schema": {
            "const": "Sim"
          }
        }
      }
    }
  ]
}
```

3.2 Explicação da Solução

No **uischema**, a propriedade *rule* é aplicada ao campo *email*.

- O *effect*: ***ENABLE*** garante que o campo permaneça desabilitado inicialmente.
- A condição define que o campo só será habilitado quando o valor de *aceitaComunicacoes* for "Sim".
- Dessa forma, o usuário só pode interagir com o campo de e-mail quando der consentimento explícito para comunicações.

Essa abordagem é particularmente relevante em contextos que envolvem privacidade de dados (como LGPD ou GDPR), onde é necessário consentimento antes de coletar informações sensíveis.

3.3 Passos para Teste

1. **Inicie o formulário:** O campo *E-mail* aparece desabilitado, pois o valor padrão de *Deseja receber comunicações?* é *"Não"*.
2. **Selecione "Sim":** Ao alterar a opção para *"Sim"*, o campo *E-mail* é habilitado automaticamente.
3. **Insira o e-mail:** Preencha um e-mail válido e verifique que o campo aceita edição normalmente.
4. **Altere novamente para "Não":** O campo *E-mail* deve voltar a ficar desabilitado, impedindo modificações.

Esse teste demonstra como utilizar o ENABLE do core do JSONForms, preenchendo uma lacuna da documentação oficial e ampliando a flexibilidade na modelagem de formulários condicionais.

Referências

- JSONForms. **API Reference – Core**. Disponível em: https://jsonforms.io/api/core/.
- JSONForms. **Documentação Oficial**. Disponível em: https://jsonforms.io/.
- Peixoto, C.; Faria, L. **Introdução ao JSON Schema**. São Paulo: Novatec, 2022.
- Google. **Technical Writing Courses**. Disponível em: https://developers.google.com/tech-writing.
- Brasil. **Lei Geral de Proteção de Dados Pessoais (LGPD)** – Lei nº 13.709, de 14 de agosto de 2018.