

Documentação Técnica de Backend - Sistema Flow Editor

Autor: Robson Menezes **Data:** 29 de Janeiro de 2026 **Versão:** 1.0

1. Introdução

Este documento serve como a **documentação técnica oficial** para o desenvolvimento do backend do sistema **Flow Editor**. O Flow Editor é uma ferramenta visual para a construção de fluxos de trabalho (flows) baseados em nós (nodes) e conexões (links).

O objetivo principal desta documentação é detalhar os **Contratos de API (API Contracts)**, fornecendo especificações claras e completas para todos os endpoints RESTful que o backend deve implementar. Estes contratos garantem a interoperabilidade e a comunicação eficiente entre o frontend (editor visual) e o serviço de backend, que é responsável pela persistência, validação e execução dos flows.

2. Convenções e Padrões

Para garantir a consistência e a manutenibilidade do sistema, as seguintes convenções e padrões devem ser estritamente seguidos na implementação do backend.

2.1. Identificadores (IDs)

Todos os identificadores de entidades (flows, nodes, links) devem ser strings únicas.

Recomenda-se o uso de UUIDs ou strings únicas com prefixos para clareza e consistência entre frontend e backend.

- **Flow ID:** Exemplo: flow-a1b2c3d4
- **Node ID:** Exemplo: node-1706563200-xyz
- **Link ID:** Exemplo: link-1706563201-abc

2.2. Referência a Variáveis de Sessão

A sintaxe para referenciar variáveis de sessão dentro das configurações dos nodes deve seguir o padrão de *handlebars* ou similar: `{{nodeId.variableName}}`.

Exemplo: Para acessar o status da resposta de um node com ID node-123, utilize `{{node-123.response.status}}`.

2.3. Estados de Execução

O ciclo de vida de uma execução de flow é definido pelos seguintes estados:

- **pending:** A solicitação de execução foi recebida e está na fila.
- **running:** O flow está atualmente em processamento.
- **completed:** A execução do flow foi concluída com sucesso.
- **failed:** A execução do flow foi interrompida devido a um erro.
- **cancelled:** A execução foi cancelada manualmente.

3. Contratos de API

Os endpoints da API estão agrupados por funcionalidade principal.

3.1. Templates de Nodes

Endpoints para obtenção dos modelos de nodes disponíveis para o editor.

GET /api/node-templates

Retorna uma lista de todos os templates de nodes disponíveis para uso no editor.

Response (200 OK): Array de Node Templates

Cada item no array representa um template de node e possui a seguinte estrutura:

- type (string): Identificador único do tipo de node (ex: “http”, “transform”).
- name (string): Nome para exibição no editor (ex: “HTTP Request”).
- icon (string): Ícone (emoji ou código) para representação visual (ex: “ ”).
- description (string): Breve descrição da funcionalidade do node.
- category (string, opcional): Categoria do node (ex: “Integrations”).
- outputs (Array de Objetos): Lista de portas de saída do node. Cada objeto output contém:
 - id (string): Identificador do output (ex: “success”, “error”).
 - label (string): Rótulo para exibição (ex: “Success”).
 - type (string, opcional): Tipo de dado esperado (ex: “any”, “object”).
- defaultConfig (Objeto, opcional): Configuração padrão do node, com pares chave-valor (ex: {“field1”: “value1”}).

Exemplo de Resposta:

```
[
  {
    "type": "http",
    "name": "HTTP Request",
    "icon": " ",
    "description": "Faz uma requisição HTTP para uma URL",
    "category": "Integrations",
    "outputs": [
      { "id": "success", "label": "Success", "type": "object" },
      { "id": "error", "label": "Error", "type": "object" }
    ]
  },
  {
    "type": "switch",
    "name": "Switch",
    "icon": " ",
    "description": "Roteamento condicional baseado em valores",
    "outputs": [
      { "id": "default", "label": "Default", "type": "any" }
    ]
  }
]
```

3.2. Gerenciamento de Flows

Endpoints para persistência e recuperação dos flows.

GET /api/flows/{flowId}

Retorna um flow específico, incluindo sua estrutura completa de nodes e links.

Path Parameters:

- `flowId` (string): O ID único do flow a ser recuperado.

Response (200 OK): Objeto Flow

O objeto Flow retornado possui a seguinte estrutura:

- `id` (string): ID único do flow.
- `name` (string): Nome do flow.
- `description` (string): Descrição do flow.
- `nodes` (Array de Objetos): Lista de nodes que compõem o flow. Cada objeto node contém:
 - `id` (string): ID único do node.
 - `type` (string): Tipo do node (deve corresponder a um template existente).
 - `name` (string): Nome customizado do node.
 - `position` (Objeto): Posição do node no editor, com `x` (number) e `y` (number).
 - `inputs` (Array de Objetos): Portas de entrada do node (geralmente fixas, ex: `{"id": "in", "label": "Input"}`).
 - `outputs` (Array de Objetos): Portas de saída do node (podem ser dinâmicas, ex: `{"id": "string", "label": "string"}`).
 - `config` (Objeto): Configuração salva do node, com pares chave-valor (ex: `{"url": "https://api.example.com", "method": "GET"}`).
- `links` (Array de Objetos): Lista de conexões entre os nodes. Cada objeto link contém:
 - `id` (string): ID único do link.
 - `from` (Objeto): Objeto com `node` (ID do node de origem) e `port` (ID da porta de origem).
 - `to` (Objeto): Objeto com `node` (ID do node de destino) e `port` (ID da porta de destino).
- `createdAt` (string): Data e hora de criação (ISO date string).
- `updatedAt` (string): Data e hora da última atualização (ISO date string).

Exemplo de Resposta:

```
{
  "id": "flow-abc",
  "name": "Meu Primeiro Flow",
  "description": "Um flow de exemplo para demonstração",
  "nodes": [
    {
      "id": "node-1",
      "type": "http",
      "name": "Requisição HTTP",
      "position": { "x": 100, "y": 100 },
      "inputs": [ { "id": "in", "label": "Input" } ],
      "outputs": [ { "id": "success", "label": "Sucesso" } ],
      "config": { "url": "https://api.example.com", "method": "GET" }
    }
  ],
  "links": [
    {
      "id": "link-1",
      "from": { "node": "node-1", "port": "success" },
      "to": { "node": "node-2", "port": "in" }
    }
  ],
  "createdAt": "2026-01-29T10:00:00Z",
  "updatedAt": "2026-01-29T10:30:00Z"
}
```

POST /api/flows/{flowId}

Salva ou atualiza a estrutura de um flow existente. O corpo da requisição deve conter a representação completa dos nodes e links do flow.

Path Parameters:

- flowId (string): O ID único do flow a ser salvo/atualizado.

Request Body:

O corpo da requisição é um objeto JSON com a seguinte estrutura:

- nodes (Array de Objetos): Array completo de nodes, seguindo o mesmo formato detalhado no GET /api/flows/{flowId}.
- links (Array de Objetos): Array completo de links, seguindo o mesmo formato detalhado no GET /api/flows/{flowId}.

Exemplo de Request Body:

```
{
  "nodes": [
    // ... array de nodes ...
  ],
  "links": [
    // ... array de links ...
  ]
}
```

Response (200 OK): Confirmação de Atualização

- success (boolean): Indica sucesso da operação (true).
- flowId (string): ID do flow atualizado.
- updatedAt (string): Data e hora da atualização (ISO date string).

Exemplo de Resposta:

```
{
  "success": true,
  "flowId": "flow-abc",
  "updatedAt": "2026-01-29T10:35:00Z"
}
```

3.3. Configuração de Nodes

Endpoints para gerenciar a configuração específica de um node.

GET /api/nodes/{nodeId}/config

Retorna o schema de configuração (que define o formulário de edição) e os valores atuais de um node específico.

Path Parameters:

- `nodeId` (string): O ID único do node.

Response (200 OK): Objeto de Configuração

O objeto de configuração retornado possui a seguinte estrutura:

- `nodeId` (string): ID do node.
- `nodeType` (string): Tipo do node.
- `schema` (Objeto): Estrutura do formulário de configuração. Contém:
 - `title` (string): Título do formulário.
 - `description` (string, opcional): Descrição do formulário.
 - `fields` (Array de Objetos): Lista de campos do formulário. Cada objeto `field` contém:
 - `id` (string): ID do campo (chave no objeto `config`).
 - `label` (string): Rótulo para exibição.
 - `type` (string): Tipo de campo. Pode ser: "text", "number", "textarea", "select", "json", "code", "dynamic".
 - `required` (boolean): Indica se o campo é obrigatório.
 - `defaultValue` (any, opcional): Valor padrão do campo.
 - `placeholder` (string, opcional): Texto de placeholder.
 - **Para** `type`: "select": `options` (Array de Objetos com `value` e `label`).
 - **Para** `type`: "number": `min`, `max`, `step` (number).
 - **Para** `type`: "json" **ou** "code": `language` (string, ex: "javascript", "json").
 - **Para** `type`: "dynamic": `dynamicType` (string, ex: "cases", "rules").
- `config` (Objeto): Valores atuais de configuração do node (pares chave-valor, ex: {"url": "https://api.example.com"}).
- `outputs` (Array de Objetos): Lista de outputs atuais do node (pode ser dinâmico, seguindo o formato de outputs em templates de nodes).

Exemplo de Resposta:

```
{
  "nodeId": "node-1",
  "nodeType": "http",
  "schema": {
    "title": "Configuração de Requisição HTTP",
    "fields": [
      { "id": "url", "label": "URL", "type": "text", "required": true },
      { "id": "method", "label": "Método", "type": "select", "options":
[ { "value": "GET", "label": "GET" } ] }
    ],
    "config": {
      "url": "https://api.example.com",
      "method": "GET"
    },
    "outputs": [
      { "id": "success", "label": "Success" }
    ]
  }
}
```

POST /api/nodes/{nodeId}/config

Salva a nova configuração de um node específico.

Path Parameters:

- nodeId (string): O ID único do node.

Request Body:

O corpo da requisição é um objeto JSON contendo a nova configuração:

- config (Objeto): Objeto com os novos valores dos campos do formulário (pares chave-valor). Para nodes com campos dinâmicos (ex: switch), pode incluir arrays como cases.

Exemplo de Request Body:

```
{
  "config": {
    "url": "https://api.new-example.com",
    "method": "POST",
    "cases": [
      { "value": "option1", "description": "Primeira opção" }
    ]
  }
}
```

Response (200 OK): Confirmação e Outputs Atualizados

- success (boolean): Indica sucesso da operação (true).
- nodeId (string): ID do node atualizado.
- updatedOutputs (Array de Objetos, opcional): Lista de outputs atualizados. Isso é crucial para nodes com outputs dinâmicos, onde a configuração pode alterar as portas de saída disponíveis.

Exemplo de Resposta:

```
{
  "success": true,
  "nodeId": "node-1",
  "updatedOutputs": [
    { "id": "case1", "label": "Opção 1" },
    { "id": "default", "label": "Padrão" }
  ]
}
```

3.4. Variáveis da Sessão

Endpoints para consulta das variáveis disponíveis para uso em um ponto específico do flow.

GET /api/flows/{flowId}/variables

Retorna todas as variáveis geradas pelos nodes anteriores no flow, que podem ser utilizadas em nodes subsequentes.

Path Parameters:

- `flowId` (string): O ID único do flow.

Query Parameters (Opcional):

- `nodeId` (string): Se fornecido, filtra as variáveis disponíveis até o ponto de execução daquele node específico.
- `type` (string): Se fornecido, filtra as variáveis por tipo (ex: “string”, “object”, “number”).

Response (200 OK): Objeto de Variáveis

O objeto retornado contém o ID do flow e um array de grupos de variáveis, cada um associado a um node:

- `flowId` (string): ID do flow.
- `variables` (Array de Objetos): Lista de grupos de variáveis, agrupadas por node. Cada objeto de grupo contém:
 - `nodeId` (string): ID do node que gerou as variáveis.
 - `nodeName` (string): Nome do node (para exibição).
 - `nodeType` (string): Tipo do node.
 - `variables` (Array de Objetos): Lista de variáveis disponíveis geradas por este node. Cada objeto `variable` contém:
 - `name` (string): Nome da variável (ex: “response”, “status”).
 - `type` (string): Tipo da variável (ex: “string”, “number”, “object”, “array”, “boolean”).
 - `description` (string, opcional): Descrição da variável.
 - `example` (any, opcional): Exemplo de valor da variável.
 - `path` (string): Caminho completo para referência (ex: `{{node-123.response}}`).

Exemplo de Resposta:

```
{
  "flowId": "demo",
  "variables": [
    {
      "nodeId": "node-123",
      "nodeName": "HTTP Request",
      "nodeType": "http",
      "variables": [
        {
          "name": "response",
          "type": "object",
          "description": "Resposta completa da requisição",
          "example": { "status": 200, "data": {} },
          "path": "{{node-123.response}}"
        },
        {
          "name": "status",
          "type": "number",
          "description": "Status code HTTP",
          "example": 200,
          "path": "{{node-123.status}}"
        }
      ]
    }
  ]
}
```

3.5. Execução e Testes

Endpoints para iniciar e monitorar a execução de um flow.

POST /api/flows/{flowId}/execute

Inicia a execução assíncrona de um flow ou de um node específico dentro de um flow.

Path Parameters:

- `flowId` (string): O ID único do flow a ser executado.

Request Body:

O corpo da requisição é um objeto JSON com as seguintes opções:

- `startNodeId` (string, opcional): O ID do node a partir do qual a execução deve começar. Se omitido, o flow será executado a partir do seu ponto de entrada padrão.
- `inputData` (Objeto, opcional): Dados de entrada que serão passados para o primeiro node da execução.
- `debug` (boolean, opcional): Se `true`, habilita a coleta de logs detalhados durante a execução para fins de depuração.

Exemplo de Request Body:

```
{
  "startNodeId": "node-start",
  "inputData": { "key": "value" },
  "debug": true
}
```

Response (200 OK - Aceita a Execução):

Retorna uma confirmação de que a execução foi aceita e iniciada.

- `executionId` (string): ID único da execução iniciada, usado para monitorar seu status.
- `status` (string): Estado inicial da execução, que será `pending`.
- `startedAt` (string): Data e hora de início da execução (ISO date string).

Exemplo de Resposta:

```
{
  "executionId": "exec-xyz",
  "status": "pending",
  "startedAt": "2026-01-29T11:00:00Z"
}
```

GET /api/executions/{executionId}

Retorna o status atual e o resultado final de uma execução de flow.

Path Parameters:

- executionId (string): O ID único da execução a ser consultada.

Response (200 OK): Objeto de Execução

O objeto Execution retornado possui a seguinte estrutura:

- executionId (string): ID da execução.
- flowId (string): ID do flow que foi executado.
- status (string): Estado atual da execução (ver Seção 2.3 para os possíveis estados: “pending”, “running”, “completed”, “failed”, “cancelled”).
- result (Objeto, opcional): Disponível quando status = completed. Contém:
 - success (boolean): Indica se o flow terminou com sucesso.
 - data (any): Dados de saída gerados pelo flow.
 - duration (number): Tempo total de execução em milissegundos.
- error (Objeto, opcional): Disponível quando status = failed. Contém:
 - message (string): Mensagem de erro.
 - code (string): Código de erro.
 - nodeId (string, opcional): ID do node onde o erro ocorreu.
- logs (Array de Objetos, opcional): Logs detalhados da execução, disponíveis se debug foi true na requisição de execução. Cada objeto log contém:
 - timestamp (string): Data e hora do log.
 - nodeId (string): ID do node relacionado ao log.
 - level (string): Nível do log (“info”, “warn”, “error”).
 - message (string): Mensagem do log.
 - data (any, opcional): Dados adicionais do log.
- startedAt (string): Data e hora de início da execução.
- completedAt (string, opcional): Data e hora de conclusão da execução.

Exemplo de Resposta (Execução Concluída):

```
{
  "executionId": "exec-xyz",
  "flowId": "flow-abc",
  "status": "completed",
  "result": {
    "success": true,
    "data": { "output": "Hello World" },
    "duration": 150
  },
  "startedAt": "2026-01-29T11:00:00Z",
  "completedAt": "2026-01-29T11:00:00.150Z"
}
```

3.6. Variáveis de Saída de Nodes (Detalhe)

Endpoint para obter o schema detalhado das variáveis geradas por um output específico de um node.

GET */api/nodes/{nodeId}/outputs/{outputId}/variables*

Retorna o schema de variáveis geradas por uma porta de saída (outputId) específica de um node (nodeId). Isso é útil para entender a estrutura dos dados que um node produz.

Path Parameters:

- `nodeId` (string): O ID do node.
- `outputId` (string): O ID da porta de saída (ex: “success”, “error”).

Response (200 OK): Lista de Variáveis Detalhadas

O objeto retornado contém o ID do node, o ID do output e um array de variáveis detalhadas:

- `nodeId` (string): ID do node.
- `outputId` (string): ID da porta de saída.
- `variables` (Array de Objetos): Lista de variáveis geradas por este output. Cada objeto `variable` contém:
 - `name` (string): Nome da variável.
 - `type` (string): Tipo da variável (ex: “string”, “object”, “array”).
 - `description` (string, opcional): Descrição da variável.
 - `example` (any, opcional): Exemplo de valor.
 - `required` (boolean, opcional): Indica se a variável é sempre gerada por este output.
 - `schema` (Objeto, opcional): Schema JSON detalhado para objetos complexos, descrevendo suas propriedades.

Exemplo de Resposta:

```
{
  "nodeId": "node-http-request",
  "outputId": "success",
  "variables": [
    {
      "name": "responseBody",
      "type": "object",
      "description": "Corpo da resposta HTTP",
      "required": true,
      "schema": {
        "type": "object",
        "properties": {
          "status": { "type": "number" },
          "data": { "type": "object" }
        }
      }
    },
    {
      "name": "headers",
      "type": "object",
      "description": "Cabeçalhos da resposta HTTP"
    }
  ]
}
```

3.7. Gerenciamento de Nodes e Links (Exclusão)

Endpoints para remoção de componentes do flow.

DELETE /api/nodes/{nodeId}

Remove um node específico do flow e, automaticamente, todas as conexões (links) associadas a ele.

Path Parameters:

- `nodeId` (string): O ID único do node a ser removido.

Response (200 OK): Confirmação de Remoção

- `success` (boolean): Indica sucesso da operação (true).
- `removedLinks` (number): O número de links que foram removidos como consequência da exclusão do node.

Exemplo de Resposta:

```
{
  "success": true,
  "removedLinks": 2
}
```

DELETE /api/links/{linkId}

Remove uma conexão (link) específica entre dois nodes.

Path Parameters:

- `linkId` (string): O ID único do link a ser removido.

Response (200 OK): Confirmação de Remoção

- `success` (boolean): Indica sucesso da operação (true).

Exemplo de Resposta:

```
{
  "success": true
}
```

3.8. Validação do Flow

Endpoint para verificar a integridade e a validade de um flow antes de salvá-lo ou executá-lo.

POST /api/flows/{flowId}/validate

Valida a estrutura e a configuração de um flow, identificando possíveis erros ou avisos que possam impedir sua execução correta.

Path Parameters:

- `flowId` (string): O ID único do flow a ser validado.

Response (200 OK): Objeto de Validação

O objeto retornado detalha o status de validação do flow:

- `valid` (boolean): Indica se o flow é considerado válido para execução (`true`) ou se possui erros (`false`).
- `errors` (Array de Objetos, opcional): Lista de erros encontrados. Cada objeto `error` contém:
 - `type` (string): Tipo de erro (ex: “missing_config”, “invalid_connection”, “circular_dependency”).
 - `message` (string): Mensagem de erro detalhada.
 - `nodeId` (string, opcional): ID do node relacionado ao erro.
 - `linkId` (string, opcional): ID do link relacionado ao erro.
 - `field` (string, opcional): Campo específico relacionado ao erro.
- `warnings` (Array de Objetos, opcional): Lista de avisos (warnings) que não impedem a execução, mas podem indicar configurações não ideais. Cada objeto `warning` contém:
 - `type` (string): Tipo de aviso.
 - `message` (string): Mensagem do aviso.
 - `nodeId` (string, opcional): ID do node relacionado ao aviso.

Exemplo de Resposta (Flow Válido):

```
{
  "valid": true,
  "errors": [],
  "warnings": []
}
```

Exemplo de Resposta (Flow com Erros):

```
{
  "valid": false,
  "errors": [
    {
      "type": "missing_config",
      "message": "O node 'Requisição HTTP' não possui a URL configurada.",
      "nodeId": "node-http-request",
      "field": "url"
    }
  ],
  "warnings": [
    {
      "type": "unused_node",
      "message": "O node 'Log' não está conectado a nenhum outro node.",
      "nodeId": "node-log"
    }
  ]
}
```

4. Fluxo de Trabalho de Desenvolvimento (Frontend-Backend)

A seguir, um resumo da sequência de chamadas de API esperadas para as principais interações do usuário no Flow Editor. Esta seção visa guiar a implementação da lógica de interação entre o frontend e o backend.

- **Carregar Editor:**
 1. GET /api/node-templates: Para obter a lista de todos os tipos de nodes disponíveis.
 2. GET /api/flows/{flowId}: Para carregar a estrutura (nodes e links) de um flow salvo.
- **Adicionar Novo Node:**
 1. Após o usuário adicionar um node no frontend, a estrutura completa do flow (com o novo node) é enviada para POST /api/flows/{flowId} para persistência.
- **Configurar Node:**
 1. GET /api/nodes/{nodeId}/config: Para obter o schema de configuração e os valores atuais do node, que serão usados para renderizar o formulário de edição no frontend.
 2. POST /api/nodes/{nodeId}/config: Após o usuário salvar as configurações, os novos valores são enviados para este endpoint. A resposta pode incluir updatedOutputs se as portas de saída do node foram alteradas.
- **Conectar Nodes:**
 1. Após o usuário criar um link entre nodes no frontend, a estrutura completa do flow (com o novo link) é enviada para POST /api/flows/{flowId} para persistência.
- **Usar Variáveis (Autocompletar/Drag & Drop):**
 1. GET /api/flows/{flowId}/variables: Para obter a lista de variáveis disponíveis que podem ser referenciadas em campos de configuração de nodes.
- **Testar/Executar Flow:**
 1. POST /api/flows/{flowId}/execute: Para iniciar a execução assíncrona do flow.
 2. GET /api/executions/{executionId}: Este endpoint deve ser consultado periodicamente (polling) para monitorar o status e obter o resultado final da execução.
- **Salvar Flow:**
 1. POST /api/flows/{flowId}/validate: Recomenda-se validar o flow antes de salvá-lo para garantir sua integridade.
 2. Se a validação for bem-sucedida, POST /api/flows/{flowId} é chamado para persistir as alterações.

5. Conclusão

Os contratos de API detalhados neste documento fornecem a base para a implementação robusta e escalável do backend do Flow Editor. A adesão estrita a estas especificações é crucial para o sucesso do projeto. O backend deve ser capaz de gerenciar a persistência de flows complexos, fornecer os schemas de configuração dinâmicos e orquestrar a execução assíncrona dos fluxos de trabalho definidos. Esta versão da documentação foi otimizada para facilitar a edição e manutenção em ambientes como o LibreOffice, garantindo um fluxo de trabalho mais suave para a equipe de desenvolvimento.