# **Guia Técnico Definitivo para a Estrutura JSON de Workflows no n8n**

Este documento serve como um relatório técnico exaustivo, detalhando a estrutura, interpretação e melhores práticas para a criação de arquivos JSON de workflows no n8n. O objetivo é fornecer uma especificação tão precisa e completa que possa ser utilizada para a geração programática de automações complexas, servindo como uma referência canônica para desenvolvedores, arquitetos de automação e sistemas de inteligência artificial.

## **A Anatomia de um Workflow JSON no n8n**

Um arquivo JSON de workflow do n8n é a representação completa e portável de uma automação. Ele encapsula a lógica, os metadados e as configurações visuais de tudo o que é construído na tela do n8n. Compreender sua estrutura é o primeiro passo para a manipulação e geração programática.

### **O Objeto Raiz: Uma Planta Baixa de Alto Nível**

Na sua essência, o JSON de um workflow é um único objeto que serve como o contêiner para toda a definição da automação. Este objeto raiz contém várias chaves de nível superior que descrevem o workflow, seus componentes e seu estado. As chaves primárias são: name, nodes, connections, settings, staticData, id, active, createdAt, updatedAt e tags.

É fundamental entender que, embora o JSON defina a lógica de forma autônoma, ele não é totalmente autossuficiente para a execução. O arquivo JSON representa o *plano lógico*, mas depende de entidades externas dentro da instância do n8n para o *contexto de execução*. A principal dependência externa são as credenciais. A interface de linha de comando (CLI) do n8n reflete essa separação, com comandos distintos para exportar workflows (export:workflow) e credenciais (export:credentials).1 Discussões na comunidade e tutoriais de migração confirmam que, para mover uma automação funcional, é necessário mover o JSON do workflow e, separadamente, gerenciar as credenciais, seja através da migração da chave de criptografia da instância (

N8N\_ENCRYPTION\_KEY) ou da reconfiguração manual.2 Esta distinção entre a definição lógica (o JSON) e o contexto de execução (a instância com suas credenciais) é a pedra angular para a geração programática bem-sucedida.

### **Metadados e Configurações em Nível de Workflow**

As propriedades de nível superior no objeto raiz definem as características gerais e o comportamento do workflow.

* **name (string):** O nome legível por humanos do workflow, que aparece na interface do usuário do n8n. É uma boa prática que este nome seja descritivo do propósito da automação.
* **id (string):** O identificador único do workflow dentro do banco de dados da instância n8n. Ao importar um workflow via UI, um novo id é geralmente atribuído. No entanto, ao usar a CLI para importar, se um id no arquivo JSON corresponder a um workflow existente no banco de dados, o workflow existente será **sobrescrito**.1 Este comportamento é crítico para scripts de implantação automatizada (CI/CD).
* **active (boolean):** Define o estado de ativação do workflow. Um valor true indica que o workflow está ativo e seus gatilhos (triggers) estão "escutando" por eventos. Um valor false o desativa.
* **tags (array de strings):** Um array de strings que representam as tags associadas ao workflow, usadas para organização e filtragem na interface do n8n.
* **createdAt / updatedAt (string, formato ISO 8601):** Timestamps que registram a data e hora de criação e da última modificação do workflow, respectivamente.
* **settings (objeto):** Um contêiner para configurações específicas do workflow. Isso pode incluir a configuração de um "Error Workflow" (um workflow a ser executado em caso de falha), o fuso horário de execução (timezone), e outras opções que governam o comportamento geral da execução.
* **staticData (objeto):** Utilizado para armazenar dados "fixados" (pinned data). Este recurso permite que um desenvolvedor teste um workflow com um conjunto de dados de entrada fixo, sem a necessidade de executar o nó de gatilho repetidamente. É primariamente um auxílio ao desenvolvimento e pode ser omitido em JSONs destinados à produção, pois os dados fixados não são usados em execuções de produção reais.4

## **Desconstruindo o Array nodes: A Lógica Central**

O coração de qualquer workflow n8n reside no array nodes. Cada objeto dentro deste array representa um único bloco funcional na tela do n8n, seja um gatilho, uma ação ou uma função lógica.

### **Propriedades Universais dos Nós: A Linguagem Comum**

Apesar da vasta diversidade de nós, todos compartilham um conjunto de propriedades fundamentais que definem sua identidade e aparência no canvas.

* **id (string):** Um identificador único para o nó *dentro do escopo do workflow*. Este ID é crucial, pois é referenciado pelo objeto connections para estabelecer as ligações lógicas entre os nós. Ele não deve ser confundido com o id global do workflow.
* **name (string):** O rótulo legível por humanos exibido no nó dentro do canvas. A melhor prática, conforme demonstrado na especificação do projeto "Página Rei", é tornar este nome altamente descritivo da função do nó (ex: "Buscar Detalhes do Produto na Amazon") para melhorar a legibilidade e manutenção.6
* **type (string):** O identificador programático do tipo do nó. Este é um campo crítico que instrui o n8n sobre qual componente de software renderizar na interface e qual lógica executar no backend. Exemplos incluem n8n-nodes-base.webhook, n8n-nodes-base.code, e n8n-nodes-base.httpRequest.6
* **typeVersion (integer):** A versão do código do nó. Este campo é de suma importância para a compatibilidade. As atualizações do n8n podem introduzir novas versões de nós existentes, com parâmetros ou comportamentos alterados.21 Um workflow exportado de uma instância mais recente do n8n pode conter um  
  typeVersion que uma instância mais antiga não consegue interpretar, resultando em falhas de importação ou comportamento inesperado. A geração programática de workflows deve, idealmente, fixar uma typeVersion específica para garantir a estabilidade ou estar ciente do potencial de incompatibilidade entre versões.
* **position (array de números):** Um array com dois valores numéricos, [x, y], que representam as coordenadas do nó no canvas do editor. Esta propriedade é puramente para o layout visual e não afeta a lógica de execução.
* **notes (string, opcional):** Um campo de texto para comentários adicionados pelo usuário. É útil para documentar lógicas complexas ou fornecer instruções diretamente no nó, visível na interface do usuário.

### **O Objeto parameters: O Coração da Configuração do Nó**

O objeto parameters é onde a configuração específica de cada nó é definida. Sua estrutura é altamente variável e única para cada type de nó. É aqui que URLs, métodos de requisição, corpos de dados, scripts e outras lógicas são especificados.

* **Exemplo - Nó Webhook:** Os parâmetros incluem httpMethod (ex: 'POST'), path (o slug da URL do webhook), responseMode (ex: 'onReceived'), e authentication (ex: 'none').6
* **Exemplo - Nó Code:** O parâmetro principal é js (ou python), que contém o código a ser executado como uma string. Strings de múltiplas linhas são representadas com caracteres de nova linha (\n).7 Outro parâmetro importante é  
  mode, que pode ser runOnceForAllItems ou runOnceForEachItem, definindo como o nó itera sobre os dados de entrada.9
* **Exemplo - Nó HTTP Request:** Este nó tem um objeto de parâmetros complexo, incluindo url, method, sendBody (boolean), bodyContentType (ex: 'json'), jsonBody (para o corpo da requisição), authentication (ex: 'basicAuth'), e objetos para definir headerParameters e queryParameters.6
* **Exemplo - Nó AWS Lambda:** Os parâmetros chave são functionName (o nome da função Lambda a ser invocada) e payload, que geralmente contém uma expressão n8n para passar dados de um nó anterior, como {{ $('Nome do Nó Anterior').item.json }}.6

### **O Objeto credentials: Referenciando Segredos de Forma Segura**

Este é um dos conceitos de segurança mais importantes no n8n. O JSON do workflow **nunca armazena segredos (como chaves de API ou senhas) em texto plano**. Em vez disso, ele armazena uma referência a uma credencial que está guardada de forma segura e criptografada na instância do n8n.

A estrutura do objeto credentials dentro de um nó é um dicionário. A chave corresponde ao tipo de credencial necessário para o nó (ex: aws, httpBasicAuth, openApi).6 O valor é outro objeto que contém o

id e o name da credencial específica, conforme salva na instância do n8n.3

**Exemplo de Referência de Credencial:**

JSON

"credentials": {  
 "httpBasicAuth": {  
 "id": "123",  
 "name": "Credencial API WordPress"  
 }  
}

Ao gerar um JSON programaticamente, a estratégia mais eficaz é usar um "placeholder". O JSON deve incluir um objeto credentials com um id de placeholder (ex: "id": "placeholder-for-import") e um name descritivo. Durante o processo de importação na interface do n8n, o usuário será solicitado a mapear essa referência (identificada pelo name e pelo tipo httpBasicAuth) para uma credencial real existente em sua instância. Esta abordagem torna o processo de importação transparente e à prova de erros, sendo uma prática recomendada fundamental.3

## **Definindo a Lógica com o Objeto connections**

O objeto connections é a representação em JSON das "fiação" visíveis no canvas do n8n. Ele define o grafo de execução, especificando como os dados fluem de um nó para o outro.

### **A Estrutura de uma Conexão**

O objeto connections não é uma simples lista de pares, mas sim um mapa direcional e centrado na saída. As chaves deste objeto são os name dos nós de *saída*. Os valores são objetos que descrevem todas as conexões que se originam daquele nó. Essa estrutura torna trivial a tarefa de identificar todos os nós que recebem dados de um nó específico.

**Exemplo de Estrutura de Conexão:**

JSON

"connections": {  
 "Webhook": {  
 "main": [  
 {  
 "node": "Code",  
 "type": "main",  
 "index": 0  
 }  
 ]  
 }  
}

Neste exemplo, o nó chamado "Webhook" tem uma conexão saindo de sua porta principal (main) e indo para a porta principal (main) do nó chamado "Code".

### **Entradas e Saídas (main, error, etc.)**

* **main:** É o canal de entrada e saída padrão para a maioria dos nós. É representado por um array de arrays de conexões. O aninhamento de arrays permite que uma única porta de saída se conecte a múltiplos nós.
* **Múltiplas Saídas:** Nós condicionais como IF e Switch possuem múltiplas portas de saída. No JSON, elas são representadas por chaves indexadas, como output\_0 para a primeira condição (geralmente 'true' no nó IF) e output\_1 para a segunda ('false'). Embora um nó Code padrão não tenha múltiplas saídas nomeadas 11, a lógica para rotear dados pode ser implementada programaticamente, mas a maneira canônica de criar ramificações é usar um nó  
  IF ou Switch, e posteriormente um nó Merge para unificar os fluxos de execução, se necessário.12 Compreender como essas ramificações são representadas no objeto  
  connections é vital para gerar workflows com lógica condicional.
* **error:** Introduzida na versão 1.15.1, a saída de erro opcional permite a criação de caminhos de tratamento de falhas dedicados.14 No JSON, isso é representado por uma chave  
  error no objeto de conexão do nó de origem, similar à chave main.

## **Um Guia para o Gerenciamento de Credenciais em Workflows JSON**

Esta seção aprofunda o ciclo de vida das credenciais e sua interação com os arquivos JSON de workflow, um tópico de importância crítica para a segurança e portabilidade.

### **O Princípio da Separação: Segurança por Design**

O design do n8n adota o princípio fundamental da separação entre lógica e segredos. O arquivo JSON do workflow contém a *referência* (um ponteiro para a credencial), mas nunca o *segredo* em si. Esta abordagem oferece benefícios de segurança significativos:

1. **Previne a Exposição de Segredos:** Impede que chaves de API, tokens ou senhas sejam acidentalmente commitados em sistemas de controle de versão como o Git, uma causa comum de violações de segurança.15
2. **Permite Credenciais Específicas de Ambiente:** Facilita o uso de credenciais diferentes para ambientes de desenvolvimento, teste e produção. O mesmo JSON de workflow pode ser implantado em diferentes instâncias, cada uma mapeando as referências para suas próprias credenciais locais.
3. **Adere ao Princípio do Menor Privilégio:** A lógica do workflow não precisa de acesso direto ao valor do segredo, apenas a instância do n8n que o executa.

A própria estrutura dos arquivos de credenciais para nós customizados reforça essa ideia: o arquivo .credentials.ts define a *forma* da credencial (os campos necessários), mas não armazena seus valores.17

### **O Processo de Importação e Mapeamento**

Quando um usuário importa um workflow que contém referências a credenciais, a interface do n8n facilita um processo de mapeamento seguro. O n8n lê o name e o type da credencial do objeto credentials no JSON e solicita ao usuário que selecione uma credencial local correspondente. Por isso, a melhor prática para a geração programática é usar nomes de credenciais claros e descritivos no JSON (ex: "Chave API OpenAI para Geração de Conteúdo") para tornar este processo de mapeamento o mais intuitivo possível para o usuário final.

### **Migração Programática e a N8N\_ENCRYPTION\_KEY**

Para uma migração totalmente programática de um workflow *e* suas credenciais, sem intervenção manual, o processo é mais complexo. Requer a exportação tanto do workflow (n8n export:workflow) quanto das credenciais (n8n export:credentials) via CLI.1

O arquivo de credenciais exportado, no entanto, contém valores criptografados. Para que uma nova instância do n8n possa usar essas credenciais, ela **deve** possuir a mesma chave de criptografia da instância original. Essa chave é definida pela variável de ambiente N8N\_ENCRYPTION\_KEY ou armazenada no arquivo de configuração (.n8n/config).2 A alternativa é usar a flag

--decrypted durante a exportação, mas isso representa um risco de segurança extremo, pois expõe todos os segredos em texto plano no arquivo resultante.1 Portanto, a abordagem mais segura para uma migração automatizada (comum em pipelines de DevOps) envolve a sincronização da

N8N\_ENCRYPTION\_KEY entre as instâncias.

## **Versionamento, Compatibilidade e "Breaking Changes"**

O n8n é uma plataforma em evolução dinâmica. Para criar JSONs robustos que resistam ao teste do tempo, é essencial compreender como as atualizações podem afetar a compatibilidade.

### **Análise de "Breaking Changes" Recentes**

A análise das notas de lançamento e discussões da comunidade revela várias mudanças disruptivas ("breaking changes") que podem invalidar workflows ou alterar seu comportamento de maneiras sutis.

* **Mudança de Sintaxe no Nó Code:** Por volta da versão 1.78.0, a sintaxe para acessar dados de nós anteriores mudou de $items('NodeName') para $('NodeName').all(). Esta foi uma mudança não óbvia que quebrou workflows em produção e não foi acompanhada por um aumento de versão major (ex: 2.0.0), causando frustração na comunidade.19
* **Parsing no Nó Webhook:** Na versão 1.2.2, uma mudança na forma como o form-data era processado começou a envolver todos os valores em arrays. Isso quebrou expressões existentes que esperavam um valor escalar (ex: {{$json.body.domain}} teve que ser alterado para {{$json.body.domain}}).20
* **Comportamento de $(...).last():** Na versão 1.47.0, o comportamento de funções de acesso a dados como last(), first() e all() foi modificado para se referir à saída que se conecta ao nó seguinte, em vez de sempre se referir à primeira saída do nó de origem. Isso pode alterar a lógica em cenários complexos com múltiplas saídas.14

### **Estratégias para Garantir a Compatibilidade Futura**

Para mitigar o risco de quebra de compatibilidade, os geradores programáticos de workflows devem adotar as seguintes estratégias:

* **Fixar a typeVersion:** Especificar explicitamente a typeVersion para cada nó no JSON gerado oferece um grau de estabilidade, visando uma versão conhecida e testada do código do nó.
* **Usar Sintaxe Moderna:** Utilizar sempre a sintaxe mais atual recomendada pela documentação (ex: $('NodeName').all() em vez de $items(...)).
* **Codificação Defensiva no Nó Code:** Empregar técnicas de JavaScript como "optional chaining" (?.) e "nullish coalescing" (??) pode tornar o código mais resiliente a mudanças inesperadas nas estruturas de dados provenientes de outros nós.
* **Monitorar as Notas de Lançamento:** Manter um processo para revisar as notas de lançamento do n8n 14 em busca de "breaking changes" é essencial para a manutenção de qualquer ferramenta que gere workflows programaticamente.

## **Guia Técnico Definitivo para a Geração Programática de Workflows**

Esta seção final consolida todo o conhecimento em um guia de referência prático, projetado para ser consumido tanto por humanos quanto por máquinas.

### **Checklist de Boas Práticas para Geração de JSON Compatível**

* [ ] **Validar a Sintaxe JSON:** Sempre valide a estrutura do JSON com um linter antes de tentar a importação.
* [ ] **Garantir IDs Únicos:** Assegure que todos os campos id (tanto do workflow quanto dos nós) sejam strings únicas.
* [ ] **Usar Nomes Descritivos:** Utilize campos name claros e descritivos para nós e referências de credenciais para facilitar a manutenção e o mapeamento.
* [ ] **Verificar a Integridade das Conexões:** Garanta que cada nó referenciado no objeto connections exista no array nodes.
* [ ] **Utilizar Placeholders de Credenciais:** Inclua objetos de credenciais com um id de placeholder e um name descritivo para um processo de importação suave.
* [ ] **Escapar Caracteres Corretamente:** Em strings de código de múltiplas linhas (ex: no nó Code), certifique-se de que caracteres especiais como aspas e barras invertidas sejam devidamente escapados e que as novas linhas sejam representadas por \n.
* [ ] **Especificar typeVersion:** Defina a typeVersion para todos os nós para aumentar a estabilidade e previsibilidade do workflow entre diferentes versões do n8n.

### **Exemplos de JSON Anotados**

#### **Exemplo 1: Workflow Básico "Hello World"**

Este é um workflow mínimo que é acionado por um webhook e usa um nó "Set" para definir uma mensagem.

JSON

{  
 "name": "Hello World Simples",  
 "nodes":  
 },  
 {  
 "parameters": {  
 "path": "hello-world",  
 "responseMode": "onReceived",  
 "httpMethod": "GET"  
 },  
 "id": "e4a8b1c2-d3f4-5e6a-7b8c-9d0e1f2a3b4c",  
 "name": "Webhook Trigger",  
 "type": "n8n-nodes-base.webhook",  
 "typeVersion": 1,  
 "position": [  
 450,  
 300  
 ],  
 "webhookId": "some-webhook-id-generated-by-n8n"  
 },  
 {  
 "parameters": {  
 "values": {  
 "string": [  
 {  
 "name": "message",  
 "value": "Olá, n8n! O workflow foi acionado."  
 }  
 ]  
 },  
 "options": {}  
 },  
 "id": "a1b2c3d4-e5f6-7a8b-9c0d-1e2f3a4b5c6d",  
 "name": "Define Mensagem",  
 "type": "n8n-nodes-base.set",  
 "typeVersion": 2,  
 "position": [  
 650,  
 300  
 ]  
 }  
 ],  
 "connections": {  
 "Webhook Trigger": {  
 "main":  
 }  
 },  
 "active": false,  
 "settings": {},  
 "id": "1",  
 "tags":  
}

#### **Exemplo 2: JSON do Workflow "Página Rei" (Estrutura Completa)**

Este exemplo representa a estrutura do workflow complexo detalhado no documento de especificação "Página Rei".6 Ele demonstra o uso de múltiplos nós, expressões, referências de credenciais e lógica sequencial.

JSON

{  
 "name": "Página Rei - Geração de Conteúdo",  
 "nodes":  
 },  
 {  
 "parameters": {  
 "path": "pagina-rei-generate",  
 "responseMode": "onReceived",  
 "httpMethod": "POST"  
 },  
 "id": "webhook-trigger",  
 "name": "Webhook",  
 "type": "n8n-nodes-base.webhook",  
 "typeVersion": 1,  
 "position":   
 },  
 {  
 "parameters": {  
 "jsCode": "const searchTerm = $json.body.search\_term;\nreturn { \n operation: \"SearchItems\", \n payload: { \n Keywords: searchTerm, \n PartnerTag: \"seu-id-br-20\", \n PartnerType: \"Associates\", \n Marketplace: \"www.amazon.com.br\", \n SearchIndex: \"PetProducts\", \n ItemCount: 5,\n Resources:\n }\n};"  
 },  
 "id": "prepare-search-payload",  
 "name": "Prepare SearchItems Payload",  
 "type": "n8n-nodes-base.code",  
 "typeVersion": 2,  
 "position":   
 },  
 {  
 "parameters": {  
 "functionName": "lambda\_proxy\_paapi5",  
 "payload": "={{ $('Prepare SearchItems Payload').item.json }}"  
 },  
 "id": "invoke-search-lambda",  
 "name": "Invoke Search Lambda",  
 "type": "n8n-nodes-aws.awsLambda",  
 "typeVersion": 1,  
 "position": ,  
 "credentials": {  
 "aws": {  
 "id": "aws-credential-id",  
 "name": "AWS Credentials for Lambda"  
 }  
 }  
 },  
 {  
 "parameters": {  
 "jsCode": "const items = $json.body.SearchResult.Items;\nconst asins = items.map(item => item.ASIN);\nreturn { asins };"  
 },  
 "id": "extract-asins",  
 "name": "Extract ASINs",  
 "type": "n8n-nodes-base.code",  
 "typeVersion": 2,  
 "position":   
 },  
 {  
 "parameters": {  
 "jsCode": "const asins = $json.asins;\nreturn {\n operation: \"GetItems\",\n payload: {\n ItemIds: asins,\n PartnerTag: \"seu-id-br-20\",\n PartnerType: \"Associates\",\n Marketplace: \"www.amazon.com.br\",\n Resources:\n }\n};"  
 },  
 "id": "prepare-getitems-payload",  
 "name": "Prepare GetItems Payload",  
 "type": "n8n-nodes-base.code",  
 "typeVersion": 2,  
 "position":   
 },  
 {  
 "parameters": {  
 "functionName": "lambda\_proxy\_paapi5",  
 "payload": "={{ $('Prepare GetItems Payload').item.json }}"  
 },  
 "id": "invoke-getitems-lambda",  
 "name": "Invoke GetItems Lambda",  
 "type": "n8n-nodes-aws.awsLambda",  
 "typeVersion": 1,  
 "position": ,  
 "credentials": {  
 "aws": {  
 "id": "aws-credential-id",  
 "name": "AWS Credentials for Lambda"  
 }  
 }  
 },  
 {  
 "parameters": {  
 "jsCode": "// Este código iteraria sobre o resultado de GetItems e formataria um prompt de texto limpo para a IA.\nconst items = $json.body.ItemsResult.Items;\nlet prompt\_text = \"\";\nitems.forEach((item, index) => {\n prompt\_text += `Produto ${index + 1}:\\nTítulo: ${item.ItemInfo.Title.DisplayValue}\\nPreço: ${item.Offers?.Listings?.Price?.DisplayAmount |  
  
| 'N/A'}\\nCaracterísticas: ${item.ItemInfo.Features?.DisplayValues.join(', ') |  
| 'N/A'}\\n\\n`;\n});\nreturn { prompt\_text };"  
 },  
 "id": "format-ai-prompt",  
 "name": "Format AI Prompt",  
 "type": "n8n-nodes-base.code",  
 "typeVersion": 2,  
 "position":   
 },  
 {  
 "parameters": {  
 "model": "gpt-4-turbo",  
 "messages": {  
 "message":  
 }  
 },  
 "id": "generate-review-openai",  
 "name": "OpenAI",  
 "type": "n8n-nodes-base.openAiChat",  
 "typeVersion": 2,  
 "position": ,  
 "credentials": {  
 "openAiApi": {  
 "id": "openai-credential-id",  
 "name": "OpenAI API Key"  
 }  
 }  
 },  
 {  
 "parameters": {  
 "method": "POST",  
 "url": "https://your-site.com/wp-json/pagina-rei/v1/add-review",  
 "sendBody": true,  
 "bodyContentType": "json",  
 "jsonBody": "={{ { \"search\_term\": $('Webhook').item.json.body.search\_term, \"review\_content\": $('OpenAI').item.json.choices.message.content } }}"  
 },  
 "id": "post-to-wordpress",  
 "name": "HTTP Request",  
 "type": "n8n-nodes-base.httpRequest",  
 "typeVersion": 4.1,  
 "position": ,  
 "credentials": {  
 "httpBasicAuth": {  
 "id": "wordpress-credential-id",  
 "name": "WordPress Application Password"  
 }  
 }  
 }  
 ],  
 "connections": {  
 "Webhook": { "main":] },  
 "Prepare SearchItems Payload": { "main":] },  
 "Invoke Search Lambda": { "main":] },  
 "Extract ASINs": { "main": [[{ "node": "Prepare GetItems Payload", "type": "main", "index": 0 }]] },  
 "Prepare GetItems Payload": { "main": [[{ "node": "Invoke GetItems Lambda", "type": "main", "index": 0 }]] },  
 "Invoke GetItems Lambda": { "main": [[{ "node": "Format AI Prompt", "type": "main", "index": 0 }]] },  
 "Format AI Prompt": { "main": [[{ "node": "OpenAI", "type": "main", "index": 0 }]] },  
 "OpenAI": { "main":] }  
 },  
 "active": true,  
 "settings": {},  
 "id": "2",  
 "tags":  
}

### **Tabelas Chave para Interpretação de Máquina**

As tabelas a seguir fornecem uma especificação formal da estrutura JSON, ideal para ser usada como referência por um sistema de geração de código.

**Tabela 1: Propriedades do Objeto Raiz do Workflow**

| Propriedade | Tipo de Dado | Obrigatório | Descrição |
| --- | --- | --- | --- |
| name | String | Sim | O nome legível por humanos do workflow. |
| id | String | Sim | Identificador único do workflow no banco de dados do n8n. |
| active | Boolean | Sim | true se o workflow estiver ativo, false caso contrário. |
| nodes | Array[Object] | Sim | Um array contendo a definição de cada nó no workflow. |
| connections | Object | Sim | Um objeto que define as conexões (fluxo de execução) entre os nós. |
| settings | Object | Opcional | Configurações em nível de workflow, como tratamento de erros e fuso horário. |
| tags | Array | Opcional | Tags para organização na interface do n8n. |
| createdAt | String | Sim | Timestamp (ISO 8601) da criação do workflow. |
| updatedAt | String | Sim | Timestamp (ISO 8601) da última atualização do workflow. |
| staticData | Object | Opcional | Armazena dados fixados (pinned data) para fins de teste. |

**Tabela 2: Propriedades do Objeto de Nó**

| Propriedade | Objeto Pai | Tipo de Dado | Obrigatório/Opcional | Descrição e Propósito |
| --- | --- | --- | --- | --- |
| id | Nó | String | Obrigatório | Identificador único do nó dentro do workflow, usado para conexões. |
| name | Nó | String | Obrigatório | O nome do nó exibido no canvas. |
| type | Nó | String | Obrigatório | Identificador do tipo de nó (ex: n8n-nodes-base.code). Define a funcionalidade. |
| typeVersion | Nó | Integer | Obrigatório | A versão do código do nó, crucial para compatibilidade. |
| position | Nó | Array[Number] | Obrigatório | Coordenadas [x, y] para o posicionamento visual no canvas. |
| notes | Nó | String | Opcional | Comentários ou anotações do usuário para o nó. |
| parameters | Nó | Object | Obrigatório | Objeto contendo todas as configurações específicas da funcionalidade do nó. A estrutura interna varia por type. |
| credentials | Nó | Object | Opcional | Objeto que contém referências a credenciais salvas na instância do n8n. |
| `` | credentials | Object | Obrigatório (se credentials existir) | Chave com o nome do tipo de credencial (ex: aws, httpBasicAuth). |
| id | `` | String | Obrigatório | O ID da credencial salva na instância do n8n. Use um placeholder para portabilidade. |
| name | `` | String | Obrigatório | O nome da credencial salva. Use um nome descritivo para facilitar o mapeamento. |

**Tabela 3: Propriedades do Objeto de Conexão**

| Propriedade | Objeto Pai | Tipo de Dado | Descrição e Propósito |
| --- | --- | --- | --- |
| [Output Node Name] | connections | Object | Chave é o name do nó de onde a conexão se origina. |
| main | [Output Node Name] | Array[Array[Object]] | Array que contém as conexões que saem da porta de saída principal (main). |
| output\_N | [Output Node Name] | Array[Array[Object]] | Array para conexões de saídas múltiplas (ex: nó IF), onde N é um índice (0, 1, 2...). |
| error | [Output Node Name] | Array[Array[Object]] | Array para conexões que saem da porta de tratamento de erros. |
| [Connection Array] | main, output\_N, error | Array[Object] | Um array contendo um ou mais objetos de conexão de destino. |
| node | Objeto de Conexão | String | O name do nó de destino da conexão. |
| type | Objeto de Conexão | String | O nome da porta de entrada no nó de destino (geralmente main). |
| index | Objeto de Conexão | Integer | O índice da porta de entrada no nó de destino (geralmente 0). |

#### Referências citadas

1. CLI commands - n8n Docs, acessado em julho 31, 2025, <https://docs.n8n.io/hosting/cli-commands/>
2. N8N Workflows & Credentials Migration: Export & Import Tutorial - DevSnit, acessado em julho 31, 2025, <https://devsnit.com/en/n8n-workflows-and-credentials-migration-tutorial/>
3. Nodes Credentials Setup - n8n Community, acessado em julho 31, 2025, <https://community.n8n.io/t/nodes-credentials-setup/54511>
4. n8n Quick Start Tutorial: Build Your First Workflow [2025] - YouTube, acessado em julho 31, 2025, <https://www.youtube.com/watch?v=4cQWJViybAQ>
5. AWS Lambda and Box: Automate Workflows with n8n, acessado em julho 31, 2025, <https://n8n.io/integrations/aws-lambda/and/box/>
6. Página Rei Automation Specification\_.docx
7. Teach Your AI to Use n8n Code Node / JS Expressions: My Comprehensive AI System Prompt to Use for n8n Tasks - Reddit, acessado em julho 31, 2025, <https://www.reddit.com/r/n8n/comments/1huce7n/teach_your_ai_to_use_n8n_code_node_js_expressions/>
8. "Set" Node: Value Field in Expression Mode Outputs Literal JavaScript Code Instead of Evaluated Result · Issue #15900 · n8n-io/n8n - GitHub, acessado em julho 31, 2025, <https://github.com/n8n-io/n8n/issues/15900>
9. Code Node for Beginners - Tips & Tricks - n8n Community, acessado em julho 31, 2025, <https://community.n8n.io/t/code-node-for-beginners/22031>
10. N8N - JSON2VIDEO Documentation, acessado em julho 31, 2025, <https://json2video.com/docs/v2/no-code-integrations/n8n>
11. Code node with multiple outputs : r/n8n - Reddit, acessado em julho 31, 2025, <https://www.reddit.com/r/n8n/comments/1cxtrc3/code_node_with_multiple_outputs/>
12. Merge - n8n Docs, acessado em julho 31, 2025, <https://docs.n8n.io/integrations/builtin/core-nodes/n8n-nodes-base.merge/>
13. Creating a workflow that Combines the output from 2 different node to get a JSON output with key (output coming from one Node) and its Value (Output coming from different node) - n8n Community, acessado em julho 31, 2025, <https://community.n8n.io/t/creating-a-workflow-that-combines-the-output-from-2-different-node-to-get-a-json-output-with-key-output-coming-from-one-node-and-its-value-output-coming-from-different-node/27312>
14. Release notes | n8n Docs, acessado em julho 31, 2025, <https://docs.n8n.io/release-notes/>
15. Backup workflows to git repository on Github - N8N, acessado em julho 31, 2025, <https://n8n.io/workflows/2532-backup-workflows-to-git-repository-on-github/>
16. Save your workflows into a GitHub repository - N8N, acessado em julho 31, 2025, <https://n8n.io/workflows/817-save-your-workflows-into-a-github-repository/>
17. Credentials files - n8n Docs, acessado em julho 31, 2025, <https://docs.n8n.io/integrations/creating-nodes/build/reference/credentials-files/>
18. Automated n8n Credential Backups to Google Drive with Scheduled Execution, acessado em julho 31, 2025, <https://n8n.io/workflows/4517-automated-n8n-credential-backups-to-google-drive-with-scheduled-execution/>
19. Breaking Change in Code Node - Upgrade to 1.78.0 - Questions ..., acessado em julho 31, 2025, <https://community.n8n.io/t/breaking-change-in-code-node-upgrade-to-1-78-0/79573>
20. Breaking changes in webhook format after upgrade? - Questions - n8n Community, acessado em julho 31, 2025, <https://community.n8n.io/t/breaking-changes-in-webhook-format-after-upgrade/29461>
21. n8n/CHANGELOG.md at master · gogosoon/n8n - GitHub, acessado em julho 31, 2025, <https://github.com/gogosoon/n8n/blob/master/CHANGELOG.md>