

RETORNA DIGITAL

Kit de Prompts para n8n

30 prompts prontos para planejar, construir, depurar e manter automações com mais clareza.

GRÁTIS

Planeje melhor o fluxo antes de começar a ligar nodes.

Como usar este kit

Escolha o cenário, substitua os campos entre **[colchetes]** e forneça exemplos reais de payload sempre que possível.

Importante: remova tokens e dados sensíveis dos exemplos. Para integrações, confira a documentação atual do n8n e do serviço externo antes de ativar em produção.

1	Descreva o processo
2	Cole um payload seguro
3	Planeje antes de montar
4	Teste antes de ativar

Planejamento de automações

01 Transformar processo manual em workflow

Objetivo: Converter uma rotina repetitiva em um fluxo n8n.

Atue como especialista em automação com n8n.

Quero automatizar este processo manual:

[PROCESSO]

Entradas disponíveis:

[ENTRADAS]

Sistemas envolvidos:

[SISTEMAS]

Resultado esperado:

[RESULTADO]

Restrições:

[RESTRIÇÕES]

Planeje o workflow antes de sugerir nós.

Entregue:

1. gatilho;
2. sequência lógica;
3. dados que entram e saem de cada etapa;
4. decisões e ramificações;
5. tratamento de erros;
6. pontos que precisam de credenciais;
7. riscos de duplicidade;
8. como testar com segurança.

Prefira nós nativos quando atenderem bem ao caso.

Dica: Descreva primeiro o processo humano atual. É mais fácil automatizar algo que já está claro.

02 Quebrar uma automação grande em fluxos menores

Objetivo: Evitar workflows gigantes e difíceis de manter.

Analise esta automação planejada:

[DESCRIÇÃO DO WORKFLOW]

Ela possui estas responsabilidades:

[RESPONSABILIDADES]

Sugira como dividir o fluxo em workflows menores e reutilizáveis.

Considere:

- responsabilidade de cada fluxo;
- dados de entrada e saída;
- quando chamar sub-workflows;
- erros;
- retries;
- observabilidade;
- manutenção.

Não fragmente apenas para reduzir o número de nós. Preserve coesão.

Dica: Separe por responsabilidade, não por quantidade de nós.

03 Escolher o melhor gatilho

Objetivo: Definir como uma automação deve começar.

Preciso iniciar uma automação quando:

[EVENTO]

Sistemas envolvidos:

[SISTEMAS]

Frequência:

[FREQUÊNCIA]

Latência aceitável:

[LATÊNCIA]

Compare formas de disparo adequadas no n8n, como webhook, agendamento ou trigger específico da integração.

Avalie:

- simplicidade;
- atraso;
- confiabilidade;
- risco de duplicidade;
- necessidade de polling;
- manutenção.

Não assumo que existe um trigger nativo sem confirmação.

Dica: Quando existir webhook confiável, ele costuma evitar polling desnecessário.

04 Projetar workflow idempotente

Objetivo: Evitar processar o mesmo evento mais de uma vez.

Quero tornar este workflow idempotente:

[FLUXO]

O evento possui estes identificadores:

[IDENTIFICADORES]

Destino:

[DESTINO]

Explique como impedir processamento duplicado.

Considere:

- chave idempotente;
- persistência;
- consulta antes da escrita;
- concorrência;
- retries;
- falha após efeito parcial.

Proponha a solução mais simples adequada ao risco.

Dica: Idempotência é especialmente importante em pagamentos, webhooks e retries.

05 Planejar automação segura para produção

Objetivo: Preparar um workflow para sair do protótipo.

Este workflow funciona em teste:

[DESCRIÇÃO]

Quero prepará-lo para produção.

Crie um checklist cobrindo:

- credenciais;
- variáveis;
- dados sensíveis;
- retries;
- timeouts;
- erros;
- logs;
- alertas;
- idempotência;
- volume;
- limites de APIs;
- testes;
- rollback.

Separe itens obrigatórios dos recomendáveis.

Dica: Um fluxo que funciona uma vez ainda não é necessariamente um fluxo confiável.

Dados e expressões

06 Criar expressão n8n

Objetivo: Montar uma expressão a partir de dados existentes.

Preciso criar uma expressão no n8n.

Valor desejado:
[RESULTADO DESEJADO]

Dados disponíveis:
[JSON DE ENTRADA]

Node de origem, se relevante:
[NODE]

Crie a expressão e explique:

- de onde vem cada valor;
- como tratar campo ausente;
- resultado esperado com o JSON fornecido.

Não invente campos que não existam na entrada.

Dica: Cole um exemplo real do JSON de entrada para reduzir erro de referência.

07 Corrigir expressão quebrada

Objetivo: Descobrir por que uma expressão não retorna o esperado.

Esta expressão n8n não funciona como esperado:

[EXPRESSÃO]

Entrada:

[JSON]

Resultado atual:

[ATUAL]

Resultado esperado:

[ESPERADO]

Analise:

- caminho dos dados;
- tipo dos valores;
- arrays;
- null/undefined;
- conversões;
- sintaxe.

Proponha a menor correção possível e mostre o resultado com o exemplo fornecido.

Dica: Erros de expressão muitas vezes são erros de estrutura de dados, não de sintaxe.

08 Normalizar JSON de APIs diferentes

Objetivo: Criar um formato interno comum.

Recebo dados de diferentes fontes:

Fonte A:
[JSON A]

Fonte B:
[JSON B]

Preciso gerar este formato interno:
[FORMATO DESEJADO]

Planeje a transformação no n8n.

Prefira operações simples em nós de transformação quando possível e use Code somente se realmente reduzir complexidade.

Mostre:

- mapeamento;
- valores opcionais;
- defaults;
- tratamento de arrays;
- resultado final de exemplo.

Dica: Um formato interno estável deixa o restante do workflow muito mais simples.

09 Tratar listas e itens

Objetivo: Planejar processamento correto de coleções.

Meu workflow recebe esta coleção:

[JSON]

Preciso executar:
[OPERAÇÃO]

Explique a melhor forma de processar os itens no n8n considerando:

- transformação item a item;
- agregação;
- divisão em lotes quando necessário;
- preservação de dados de origem;
- volume.

Evite loops manuais quando o modelo de itens do n8n já resolver o caso.

Dica: Antes de criar loops, entenda se o node já executa naturalmente para cada item.

10 Limpar e validar dados

Objetivo: Bloquear dados ruins antes de chamar serviços externos.

Preciso validar estes dados antes de continuar o workflow:

[JSON]

Regras:

[REGRAS]

Crie uma estratégia n8n para:

- campos obrigatórios;
- tipos;
- normalização;
- strings vazias;
- formatos inválidos;
- registros rejeitados;
- mensagem de erro.

Separe dados válidos dos inválidos sem perder contexto.

Dica: Validar cedo reduz chamadas desnecessárias a APIs e erros difíceis de rastrear.

Integrações e HTTP

11 Montar requisição HTTP

Objetivo: Traduzir documentação de API para o HTTP Request.

Preciso chamar esta API a partir do n8n:

[DOCUMENTAÇÃO OU EXEMPLO CURL]

Dados dinâmicos:

[DADOS]

Explique como configurar o HTTP Request:

- método;
- URL;
- autenticação;
- headers;
- query params;
- body;
- formato da resposta;
- timeout;
- erros esperados.

Não invente parâmetros ausentes da documentação.

Dica: Cole a documentação atual ou um cURL funcional sempre que possível.

12 Converter cURL para n8n

Objetivo: Reproduzir uma chamada existente dentro do workflow.

Converta esta chamada cURL em uma configuração de HTTP Request no n8n:

[CURL]

Identifique:

- método;
- endpoint;
- headers;
- autenticação;
- body;
- parâmetros;
- valores que devem virar expressões.

Aponte também qualquer segredo que deve ser armazenado como credencial e não hardcoded.

Dica: Antes de compartilhar um cURL, remova tokens reais.

13 Paginar uma API

Objetivo: Buscar todas as páginas sem loop infinito.

Esta API é paginada:

[DOCUMENTAÇÃO/RESPOSTA]

Preciso coletar:

[OBJETIVO]

Planeje a paginação no n8n.

Considere:

- cursor, offset ou página;
- condição de parada;
- limite;
- rate limit;
- repetição de página;
- acumulação dos resultados;
- retomada em caso de falha.

Use os campos reais fornecidos.

Dica: Defina claramente a condição de parada para não criar execuções infinitas.

14 Receber webhook com segurança

Objetivo: Desenhar entrada robusta para eventos externos.

Vou receber este webhook:

[EXEMPLO DE PAYLOAD]

Origem:

[SERVIÇO]

Requisitos de segurança/documentação:

[REQUISITOS]

Planeje o workflow para:

- validar origem ou assinatura quando aplicável;
- validar payload;
- responder rapidamente;
- evitar duplicidade;
- processar em segurança;
- registrar correlação;
- tratar retries do remetente.

Não invente um método de assinatura que não esteja na documentação do provedor.

Dica: Para webhooks, pense em duplicidade e reenvio desde o início.

15 Integrar API sem node nativo

Objetivo: Criar integração sustentável usando HTTP.

Preciso integrar [SERVIÇO] ao n8n, mas quero assumir que não existe node nativo adequado.

Documentação:
[DOCUMENTAÇÃO]

Fluxo desejado:
[FLUXO]

Planeje a integração usando recursos genéricos.

Inclua:

- autenticação;
- chamadas;
- transformação;
- paginação;
- erros;
- limites;
- renovação de token, se aplicável;
- testes.

Não invente endpoints.

Dica: Uma integração por HTTP bem documentada pode ser mais previsível que depender de abstrações.

Code node e lógica

16 Decidir entre expressão, node e código

Objetivo: Evitar Code node desnecessário.

Preciso executar esta transformação no n8n:

[TRANSFORMAÇÃO]

Entrada:

[JSON]

Compare três opções:

- expressão;
- nós nativos;
- Code node.

Avalie legibilidade, manutenção e complexidade.

Escolha a opção mais simples que resolva o problema e mostre como implementar.

Não use código só porque é possível.

Dica: Código é ótimo quando simplifica, não quando apenas substitui um node claro.

17 Gerar JavaScript para Code node

Objetivo: Criar transformação compatível com o modelo de dados do n8n.

Escreva JavaScript para um Code node do n8n.

Modo:

[ALL ITEMS OU EACH ITEM]

Entrada:

[JSON]

Objetivo:

[OBJETIVO]

Requisitos:

[REGRAS]

O código deve:

- preservar a estrutura esperada pelo n8n;
- tratar campos ausentes;
- evitar mutações desnecessárias;
- ser simples de manter.

Depois mostre um exemplo de saída.

Dica: Informe o modo de execução do Code node porque isso muda o formato esperado.

18 Revisar Code node

Objetivo: Encontrar bugs e simplificar JavaScript dentro do workflow.

Revise este Code node:

[CÓDIGO]

Exemplo de entrada:

[JSON]

Resultado esperado:

[ESPERADO]

Procure:

- estrutura de retorno incorreta;
- acesso inseguro a campos;
- problemas com arrays;
- perda de item linking quando relevante;
- complexidade;
- lógica que poderia ser feita por node nativo.

Proponha a menor correção segura.

Dica: Se o número de itens muda, verifique também a relação entre entrada e saída.

19 Transformar dados em lote

Objetivo: Processar vários itens com uma única lógica.

Tenho estes itens:

[JSON]

Preciso:

[OBJETIVO]

Crie uma solução para processar os itens em conjunto no Code node.

Explique:

- formato de entrada;
- lógica;
- formato de saída;
- comportamento para lista vazia;
- duplicidades;
- ordem dos itens.

Evite dependências externas.

Dica: Use processamento conjunto quando a regra realmente depende do conjunto inteiro.

20 Migrar lógica de código para nodes

Objetivo: Reduzir manutenção de scripts quando possível.

Tenho este Code node:

[CÓDIGO]

Quero saber se ele pode ser substituído por nodes nativos ou expressões.

Analise cada responsabilidade e proponha:

- o que pode sair do código;
- quais nodes poderiam assumir;
- o que deve permanecer em código;
- impacto na legibilidade.

Não aumente o workflow drasticamente apenas para eliminar algumas linhas de JavaScript.

Dica: O objetivo é manutenção melhor, não atingir zero linhas de código.

Erros, retries e observabilidade

21 Criar estratégia de tratamento de erro

Objetivo: Evitar que falhas desapareçam silenciosamente.

Planeje o tratamento de erros para este workflow:

[WORKFLOW]

Integrações:

[INTEGRAÇÕES]

Falhas possíveis:

[FALHAS]

Defina:

- erros recuperáveis;
- erros permanentes;
- retries;
- backoff, quando aplicável;
- dados que devem ser registrados;
- alertas;
- fluxo de erro;
- ação humana necessária.

Evite retry infinito.

Dica: Retries devem responder a falhas transitórias, não esconder erros de regra.

22 Diagnosticar execução falha

Objetivo: Investigar uma execução n8n de forma sistemática.

Uma execução falhou.

Node:
[NODE]

Mensagem:
[ERRO]

Input do node:
[INPUT]

Configuração relevante:
[CONFIGURAÇÃO]

Workflow esperado:
[FLUXO]

Crie um diagnóstico por hipóteses.

Para cada hipótese informe:

- evidência;
- teste;
- correção.

Não conclua causa raiz sem evidência suficiente.

Dica: Compare uma execução que falhou com uma que funcionou quando possível.

23 Projetar workflow de erro

Objetivo: Centralizar alerta e diagnóstico de falhas.

Quero criar um fluxo dedicado para erros das minhas automações.

Necessidades:
[NECESSIDADES]

Canais de alerta:
[CANAIS]

Defina quais informações devem ser capturadas e enviadas, como:

- workflow;
- execução;
- horário;
- node;
- erro;
- contexto seguro;
- link ou identificador de investigação.

Evite incluir secrets ou dados pessoais desnecessários.

Dica: Alertas úteis precisam ter contexto suficiente para agir, não apenas 'workflow falhou'.

24 Planejar retry seguro

Objetivo: Repetir operação sem gerar efeitos duplicados.

Esta etapa pode falhar temporariamente:

[ETAPA]

Efeito externo:
[EFEITO]

Identificador disponível:
[IDENTIFICADOR]

Projete uma estratégia de retry segura.

- Considere:
- número máximo;
 - intervalo;
 - backoff;
 - idempotência;
 - erro que deve ou não repetir;
 - ação após esgotar tentativas.

Não recomende repetir operações não idempotentes sem proteção.

Dica: Quanto maior o efeito externo, mais importante é proteger contra duplicidade.

25 Criar logs úteis

Objetivo: Definir observabilidade sem vaziar dados.

Quero melhorar a observabilidade deste workflow:

[DESCRIÇÃO]

Defina os eventos e campos que devo registrar.

Priorize:

- início e fim;
- identificadores de correlação;
- quantidade de itens;
- decisões importantes;
- chamadas externas;
- falhas;
- tempo;
- resultado.

Liste também dados que NÃO devem ser registrados por segurança ou privacidade.

Dica: Logue o suficiente para investigar, não o payload inteiro por padrão.

Manutenção, performance e reutilização

26 Revisar workflow complexo

Objetivo: Encontrar pontos frágeis antes que virem incidentes.

Faça uma revisão técnica deste workflow:

[DESCRIÇÃO OU EXPORT JSON]

Objetivo:

[OBJETIVO]

Avalie:

- clareza;
- duplicação;
- nodes sem necessidade;
- tratamento de erro;
- idempotência;
- credenciais;
- performance;
- volume;
- observabilidade;
- reutilização.

Priorize achados por impacto e esforço.

Dica: Revisões periódicas evitam que workflows cresçam como macarrão digital.

27 Otimizar workflow de alto volume

Objetivo: Reduzir custo e tempo de execução.

Este workflow processa aproximadamente [VOLUME] itens por [PERÍODO].

Fluxo:
[FLUXO]

Métricas atuais:
[MÉTRICAS]

Identifique possíveis gargalos e proponha como medir antes de otimizar.

Considere:

- chamadas repetidas;
- processamento item a item;
- batching;
- filtros tardios;
- payloads grandes;
- APIs limitadas;
- concorrência.

Não proponha otimização sem explicar como validar o ganho.

Dica: Filtrar cedo e reduzir chamadas externas costuma valer mais que micro-otimizações.

28 Criar sub-workflow reutilizável

Objetivo: Extrair uma capacidade comum para vários fluxos.

Tenho esta lógica repetida em vários workflows:

[LÓGICA]

Entradas variáveis:

[ENTRADAS]

Saída desejada:

[SAÍDA]

Projete um sub-workflow reutilizável.

Defina:

- contrato de entrada;
- validações;
- resultado;
- erros;
- nomes claros;
- versionamento/mudanças;
- como os fluxos chamadores devem tratá-lo.

Evite criar um sub-workflow genérico demais.

Dica: Um bom sub-workflow tem contrato pequeno e responsabilidade clara.

29 Documentar workflow para outra pessoa

Objetivo: Criar documentação operacional útil.

Crie documentação para este workflow n8n:

[WORKFLOW]

Público:

[PÚBLICO]

Inclua:

- objetivo;
- gatilho;
- integrações;
- fluxo principal;
- credenciais necessárias sem expor valores;
- entradas e saídas;
- erros esperados;
- como testar;
- como desativar com segurança;
- troubleshooting.

Evite descrever cada node se isso não acrescentar entendimento.

Dica: Documente decisões e dependências, não apenas o desenho visual.

30 Gerar prompt para construir workflow com IA

Objetivo: Preparar contexto para criar uma automação com menos retrabalho.

Crie um prompt técnico para eu pedir a uma IA que me ajude a construir um workflow.

Objetivo:
[OBJETIVO]

Gatilho:
[GATILHO]

Sistemas:
[SISTEMAS]

Exemplos de entrada:
[DADOS]

Resultado:
[RESULTADO]

Restrições:
[RESTRIÇÕES]

O prompt final deve exigir que a IA:

- planeje antes de montar;
- não invente nodes ou campos;
- use documentação atual quando necessário;
- prefira nodes nativos;
- indique credenciais sem expor secrets;
- trate erros e duplicidade;
- forneça passos testáveis.

Mantenha o prompt conciso e operacional.

Dica: Este prompt é útil para começar qualquer automação nova com uma especificação melhor.

Boas práticas

- Não coloque tokens, chaves ou senhas diretamente nos prompts.
- Use exemplos reais de payload com dados sensíveis removidos.
- Teste workflows com dados controlados antes de ativar em produção.
- Planeje idempotência para webhooks, pagamentos e operações com retry.
- Prefira nodes nativos e expressões quando deixarem o fluxo mais claro.
- Use Code node quando ele simplificar uma transformação ou regra que ficaria pior visualmente.
- Consulte a documentação atual para nodes, APIs e comportamentos que possam ter mudado.

RETORNA DIGITAL

**Ferramentas prontas para trabalhar,
vender e automatizar.**

Este material faz parte da coleção gratuita da Retorna Digital.

retornaservicos.com.br