Apostila do Curso

Conteúdo e Atividades

Fundamentos do n8n

Fundamentos do N8N

Nome: _____

Quantidade de Aulas
8 aulas

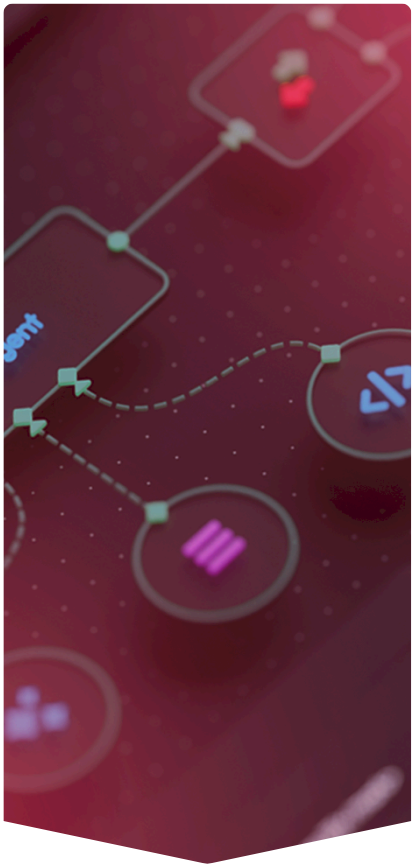
Carga horária
12 horas

Sobre o curso

O curso de n8n é a porta de entrada para qualquer um que queira se aventurar no mundo das automações. Ao longo deste curso, você vai aprender os fundamentos do N8N e da automação, entendendo como a plataforma funciona e por que é uma das principais ferramentas no-code da atualidade.

O que aprender com este curso?

Você irá criar workflows completos, conectando aplicações para automatizar tarefas do dia a dia, além de configurar agentes de inteligência artificial capazes de interpretar dados e executar ações inteligentes.



Sumário

1.1. O que é automação?

Automação é o **uso de tecnologia para executar tarefas com intervenção humana mínima**, transformando passos antes manuais em um fluxo repetível, confiável e mensurável. Na prática, você descreve **o que deve acontecer**, **quando e sob quais condições**, e sistemas (softwares, robôs, serviços em nuvem) passam a executar isso de forma consistente.

Exemplo:

Imagine uma loja online que vende roupas.

Sem automação, cada pedido exigiria um funcionário fazendo tudo manualmente: verificar pagamento, registrar em planilhas, avisar o estoque, mandar confirmação para o cliente etc.

1.2. N8N

O N8N é uma plataforma de **automação e orquestração de workflows** que combina uma interface visual com a possibilidade de rodar código quando necessário.

A ideia central é permitir que times técnicos **conectem qualquer aplicação**, **API ou dados e automatizem fluxos** complexos mantendo controle sobre os dados e a infraestrutura.

N8N - Homepage

Escaneie o código QR para acessar o conteudo ou acesse:
https://n8n.io

Acessar

A plataforma evoluiu para incluir nós e integrações focadas em IA e alcançou grande tração, centenas de integrações e dezenas de milhares de estrelas no GitHub.

1.3. n8n.cloud

A n8n.cloud é a **versão hospedada e gerenciada do N8N**, oferecida pela própria empresa que criou a plataforma. Em vez de você precisar instalar e manter o n8n no seu próprio servidor, a equipe da N8N já entrega tudo pronto na nuvem.

1.3.1. Vantagens da n8n.cloud

Zero manutenção: não precisa configurar servidores nem lidar com Docker.

Segurança gerenciada: patches e atualizações são aplicados pela equipe oficial.

Alta disponibilidade: a empresa garante que o serviço fique online.

Suporte dedicado: planos incluem suporte técnico da equipe do N8N.

1.4. Self-hosted

O N8N self-hosted significa rodar o n8n **na sua própria infraestrutura**, ou seja, você baixa a ferramenta e instala em um ambiente que você controla.

1.4.1. Responsabilidades no self-host

Instalação: configurar ambiente, banco de dados, dependências.

Atualizações: aplicar novas versões do n8n.

Segurança: configurar SSL, firewalls, autenticação, backups.

Monitoramento: garantir que os fluxos não parem se o servidor cair.

Escalabilidade: ajustar para suportar mais carga de trabalho quando necessário.

1.4.2. Vantagens do self-host

Custo zero: sem mensalidades.

Privacidade: os dados não passam por servidores de terceiros.

Personalização avançada: você pode alterar o código ou conectar sistemas internos que não podem ser expostos à nuvem pública.

Liberdade de infraestrutura: roda em qualquer ambiente que você controlar.

1.5. Docker

Docker é uma plataforma para **empacotar**, **distribuir** e **executar** aplicações dentro de **containers**, unidades leves e portáteis que isolam a aplicação e tudo o que ela precisa do sistema onde vai rodar.

A ideia central é garantir que "o programa que funciona na minha máquina" funcione da mesma maneira em qualquer outro lugar.

Docker - Homepage

Escaneie o código QR para acessar o conteudo ou acesse:
https://www.docker.com

Acessar

1.5.1. Docker Desktop

Docker Desktop é o produto oficial que traz o Docker para **máquinas desktop** (Windows e macOS) com uma experiência gráfica + integração ao sistema, facilitando o uso para desenvolvedores. Em Linux normalmente o Docker roda nativamente sem precisar do Desktop (pois o Docker Engine integra-se ao kernel Linux).

O que é Kernel?

O kernel é o núcleo do sistema operacional, a parte mais fundamental dele. Ele funciona como uma ponte entre o hardware (processador, memória, placas) e o software.

1.5.2. Conceitos-chave

Imagem: um snapshot imutável que contém o sistema de arquivos e instruções para rodar a aplicação.

Container: uma instância em execução de uma imagem, um processo isolado que compartilha o kernel do host.

Dockerfile: a receita que descreve passo a passo como construir uma imagem.

1.6. Hora da Prática: Configurando seu ambiente

Chegou a hora de pôr em prática o conteúdo que vimos em aula! Durante a aula vimos o passo-a-passo para a instalação do **Docker Desktop** em nossa máquina, portanto, sua tarefa nesta prática será **instalar e configurar o Docker Desktop**, deixando seu ambiente de trabalho pronto para seus projetos e práticas.

Dica: não se esqueça de verificar se os **recursos do Windows** necessários para o Docker funcionar estão ativados no sistema.

Anotações

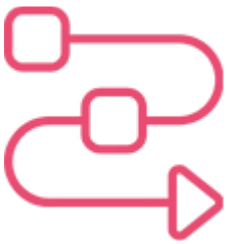
2.1. Workflow

Um **Workflow** no n8n é o **fluxo de automação** que você cria para que tarefas sejam executadas de forma automática.

Ele funciona como um **mapa visual**, onde você conecta diferentes blocos chamados Nodes, cada um responsável por uma ação específica, até formar um processo completo.

Em outras palavras: um Workflow é a representação **gráfica e lógica** da sua automação.

2.2. Nodes



No n8n, um **Node** é um **bloco de construção** do workflow. Cada Node faz uma tarefa bem definida: receber um evento, chamar um serviço externo, aplicar uma regra, transformar dados, etc. Conectando vários Nodes, você compõe a automação inteira.

2.2.1. Tipos principais de Nodes

- **Trigger Nodes:** São os **gatilhos** da automação, ou seja, iniciam o workflow quando algo acontece.
- **Service Nodes:** Também chamados de **"Action Nodes"**, eles executam ações em serviços externos.
- **Logic Nodes:** São eles que controlam o fluxo com regras e ramificações.
- **Utility Nodes:** Estes Nodes servem para editar e transformar dados.

2.2.2. Controle de fluxo ramificações

- **If/Else e Switch:** Decidem o caminho com base em condições.
- **Merge:** Junta ramos (por índice, por key ou combinação).
- **Split in Batches:** Processa listas grandes em lotes.
- **Wait:** Pausa o fluxo até um horário/evento.
- **Do Nothing:** Põe um "ponto final" ao fluxo.

2.3. Inputs e Outputs

Inputs e **outputs** são conceitos fundamentais em programação e em sistemas computacionais.

De forma direta: input é aquilo **que entra** num processo (dados, sinais, eventos); output é o **que sai** depois que o processo fez seu trabalho (resultado, efeito, resposta).

2.3.1. Inputs

Input é qualquer informação que um programa, função ou sistema recebe para processar. Pode assumir várias formas:

- Parâmetros de função
- Entrada padrão/terminal
- Arquivos
- Requisições de rede/HTTP

2.3.2. Outputs

Output é o que o sistema produz após processar os inputs. Pode ser:

- Valor de retorno de uma função
- Reposta HTTP
- Gravação em arquivo/banco de dados
- Ação física

OBSERVAÇÃO:

Há dois tipos de output importantes:

Output	Descrição	Exemplo
Output funcional	O resultado que resolve a tarefa	JSON com dados processados
Side Effects	Ações que mudam estado fora do escopo da função	Escrever no BD ou enviar email

Inputs e outputs são as peças de comunicação entre peças de software.

Projetá-los bem, com contratos claros, validação, tratamento de erros e atenção à segurança, é o que permite sistemas confiáveis, testáveis e escaláveis.

2.4. Anatomia de um node

Parâmetros: Aquilo que o Node precisa para trabalhar.

Credenciais (quando necessário): Conexão segura com o serviço externo.

Entradas e saídas: O Node recebe e devolve itens (Inputs e Outputs).

- Um item é um objeto com **json** e, opcionalmente, **binary**.
- A maioria dos Nodes processa uma **lista de itens** (um por vez), e alguns possuem **múltiplas saídas**.

JSON e Binary

JSON é um formato de texto estruturado usado para representar dados de forma organizada. Já Binary, são dados em formato bruto, representados como sequência de bytes.

Anotações

3.1. O que é um Agente de IA?

Um agente de IA é um programa que persegue objetivos de forma autônoma, combinando habilidades de raciocínio, acesso a ferramentas externas e um ciclo contínuo de **percepção → decisão → ação**.

Em vez de responder a um único prompt, um agente formula planos, supervisiona as ações que executa, atualiza seu estado/memória e quando necessário pede intervenção humana.

Quando **bem projetado**, ele amplia capacidades humanas, economiza tempo e orquestra serviços complexos. Quando **mal projetado**, acarreta riscos de segurança, custo e decisões incorretas.

Bom design = objetivo claro + schema previsível + observabilidade + controle humano

3.1.1. Como um agente opera? (Fluxo conceitual)

O ciclo típico é: **perceber → interpretar → planejar → executar → avaliar**.

Ao receber um input, o agente monta contexto + memória relevante, chama o módulo de raciocínio para decidir ações, dispara as ações via ferramentas, valida o resultado e atualiza seu estado/memória. Se algo falhar, replaneja ou escalona para um humano.

3.1.2. Tipos e arquitetura

- **Reativo:** Responde imediatamente a inputs com ações simples.
- **Deliberativo:** Gera planos multi-passos antes de agir.
- **Híbrido:** Combina planejamento e reatividade.
- **Multi-agente:** Vários agentes especializados cooperam ou competem para resolver tarefas complexas.
- **Embodied/Robótico:** Integrados a hardware para ações físicas.

3.2. Estrutura de um Agente de IA

- **Modelo de Linguagem:** Também chamado de LLM, o Modelo de Linguagem é a base do Agente de IA, funcionando como o cérebro do agente. Usam como base modelos de IA como ChatGPT e Gemini.
- **Prompt Base:** Todo Agente de IA precisa de um Prompt Base para guiá-lo, dando instruções como: objetivo, como agir, personalidade e especificações mais aprofundadas. Basicamente, são as instruções base do agente.
- **Memória:** A memória serve para que o agente consiga guardar informações e o contexto da conversa em que ele está inserido. Podem ser de curto ou longo prazo.

- **RAG:** Retrieval-Augmented Generation (RAG) é uma estratégia de "treinamento" do agente, fazendo o uso de buscas em repositórios para fornecer contexto ao LLM.

- **Tolls:** Tools são as ferramentas que o agente utiliza para realizar as tarefas/objetivo passados para ele no prompt. Permitem que o agente realize operações reais.

Agente ≠ Prompt

Um prompt é uma entrada pontual: você pergunta algo e recebe uma resposta. Agentes adicionam estado, planejamento, execução e responsabilização pelas ações. Eles não só "respondem", eles fazem.

3.3. Capacidades e Vantagens

1. Automação de tarefas complexas que exigem encadeamento de passos.

- **Ex:** Triagem, enrichment de dados, Workflows de aprovação.

2. Escala e tempo de respostas: Agentes podem operar 24/7 e processar grandes volumes com latência menor que humanos.

3. Composição de ferramentas: Orquestram múltiplos serviços sem intervenção humana.

4. Aprendizado e ajuste contínuo: Logs + retraining/finetuning podem melhorar performance ao longo do tempo.

3.4. Limitações e Riscos

1. Alucinações/Repostas incorretas: Modelos podem inventar fatos; por isso outputs factuais exigem verificação (RAG, checagem em bases confiáveis).

2. Segurança: Execução de ações sensíveis (pagamentos, remoções) pode causar danos se não houver controles rígidos.

3. Privacidade e compliance: Agentes que manipulam dados pessoais precisam de governança, criptografia e políticas de retenção.

4. Custo e latência: Chamadas a LLMs custam e introduzem latência; planejamento pesado escala de custo.

5. Fragilidade a mudanças: APIs externas ou esquemas de dados que mudam podem quebrar o agente; precisa monitoramento e manutenção.

Anotações

4.1. Desafio: Tutor de IA

Chegou a hora de colocar em prática o poder dos agentes de IA no n8n!

Sua tarefa nesta prática é desenvolver um "Agente Tutor", um assistente inteligente que ajudará o usuário a resolver problemas de estudo, mas atenção porque tem uma regra importante: **o agente não pode simplesmente dar a resposta.**

O dever do tutor é o seguinte: **guiar o raciocínio**, **fazer perguntas**, **dar dicas** e **explicar conceitos** que ajudem o usuário a entender o caminho até a solução.

Exemplo:

O usuário inicia a conversa com algo, como: "Estou com dificuldade para entender equações de segundo grau". Então, o tutor agente responde: "Entendido! Vamos começar pelo básico: você lembra o que significam as letras A, B e C na equação? Elas são essenciais para aplicar a fórmula corretamente."

Você está livre para escolher a área de atuação do tutor - matemática, física, biologia - apenas lembre-se que o que importa é seguir a regra citada antes.

Dicas

1. Pense na memória
2. Escreva um prompt claro e educativo
3. Teste pequenas interações
4. Ajuste o comportamento pelo contexto

Anotações

5.1. O que são APIs?

API é a sigla para **Application Programming Interface**, ou seja, **Interface de Programação de Aplicações**.

Na prática, uma API é como um **contrato de comunicação** que permite que dois sistemas de software conversem entre si de forma organizada. Esse contrato define **o que** pode ser pedido, **como deve** ser pedido e **como será** a resposta.

Uma boa metáfora para entender melhor como APIs funcionam, é pensar em uma **garçonete em um restaurante**:

- Você (usuário) faz o pedido.
- A garçonete (API) leva esse pedido para a cozinha (servidor).
- A cozinha prepara a comida (processa a solicitação).
- A garçonete traz o prato pronto até a sua mesa (reposta).

Você **não precisa saber como** a cozinha **funciona internamente**, só precisa saber o que pode pedir e como pedir. Da mesma forma, ao usar uma API, você não precisa entender o código do sistema por trás dela, basta seguir o contrato de comunicação.

5.1.1. Estrutura Básica

APIs modernas se organizam em alguns elementos principais:

Endpoint ou URI: é o "endereço" da API, apontando para o recurso que você quer acessar.

> **Ex:** https://api.exemplo.com/users/123

Protocolo: Indica como a comunicação ocorrerá. Existem dois tipos de protocolo:

1. **http://** - Com a falta do "S" no final indica que a comunicação **não é criptografada**.
2. **https://** - Com o "S" no final indica que a comunicação **é criptografa** e **segura**.

Host ou Domínio: É o endereço do servidor em que a API está hospedada. Quando necessário, o domínio é dividido por **pontos finais**, e nestes casos, pode apresentar:

1. **Subdomínio:** utilizado por muitas empresas para separar API do site principal.
2. **Domínio principal:** nome registrado da organização.
3. **Porta ou Port:** é opcional e por padrão, temos :80 para HTTP e :443 para HTTPS.

Versão: indica a versão atual do contrato da API que estamos utilizando.

Path ou Caminho: identifica o recurso dentro da API, e assim como o domínio, é dividido em partes, porém, por barras.

Query string: são os parâmetros de consulta e é utilizado para **filtrar**, **ordenar** ou **personalizar** a resposta.

5.1.2. Método HTTP

Indica o tipo de ação que você quer fazer.

Método	Função
GET	Buscar informação
POST	Criar algo novo
PUT / PATCH	Atualizar algo existente
DELETE	Remover

As APIs são o **coração da integração moderna**. Elas permitem que aplicativos diferentes "falem a mesma língua" sem precisar acessar diretamente os bancos de dados ou a lógica interna uns dos outros. Isso traz:

- Segurança
- Escalabilidade
- Eficiência
- Automação

5.2. Hora da prática: Responda ao Questionário

1. Qual a principal função de um endpoint (URI) em uma API?

a) Indicar a versão atual da API.

b) Indicar o endereço ou recurso que queremos acessar.

c) Criptografar a comunicação entre cliente e servidor.

d) Definir se a resposta será em JSON ou XML.

2. Qual a diferença entre http:// e https://?

a) http:// é mais rápido, e https:// mais lento.

b) http:// indica que a comunicação não é criptografada, enquanto https:// sim.

c) http:// indica que a comunicação é criptografada, enquanto https:// não.

d) Ambos significam a mesma coisa.

3. Em uma URI, o que é o path ou caminho?

a) Uma subdivisão do domínio principal.

b) O número que define a porta de acesso.

c) Um protocolo usado para criptografar os dados.

d) A parte que identifica o recurso dentro da API, separada por barras.

4. Qual é a função da query string em uma API?

a) Permitir a comunicação entre cliente e servidor.

b) Informar o domínio da API que está sendo usado.

c) Filtrar, ordenar ou personalizar a resposta da API.

d) Indicar a versão atual do contrato da API.

5.3. O que são Webhooks?

Webhooks são mecanismos de **comunicação por evento** entre aplicações: quando algo acontece em um sistema (o "emissor"), ele envia **automaticamente** uma requisição HTTP para uma URL configurada em outro sistema (o "receptor").

Em vez de o receptor ficar consultando (polling) se houve novidade, o emissor **empurra** a informação assim que o evento ocorre.

5.3.1. Como eles funcionam?

1. Registro / Subscrição: o consumidor (destino) registra um endpoint público, https://meuapp.com/webhook/stripe, por exemplo, no serviço emissor, geralmente via painel ou API de configuração.

> Muitas plataformas pedem também um secret ou mecanismo de verificação no momento do registro.

2. Evento ocorre no emissor: como por exemplo um pagamento aprovado, push de repositório, um envio de formulário ou um deploy concluído.

3. Emissor envia uma requisição HTTP POST para o endpoint registrado, contendo um payload com os dados do evento e headers adicionais.

4. Recepção e validação: o receptor valida:

1) Se a requisição veio de uma fonte confiável

2) Se o payload está bem formado

3) Se não é replay

Se for válido, o receptor processa o evento.

5. Resposta HTTP: o receptor deve responder rapidamente com um **status 2xx** (normalmente 200 OK ou 204 No Content) para indicar sucesso.

Se o receptor retornar erro (4xx/5xx) ou não responder, o emissor normalmente **re-tentará** a entrega seguindo uma política de retries e, eventualmente, notificará o administrador.

Exemplo de requisição de um Body (JSON):

json

```
{
  "evento": "paymnt.succeeded",
  "data":{
    "id": "pay_123",
    "amount": 19990,
    "currency": "BRL",
    "customer": { "id": "cus_456", "email":
"ana@ex.com" }
  }
}
```

5.4. Diferença entre Webhooks e APIs

Direção da comunicação

- **API (pull):** o cliente pede dados quando precisa.
- **Webhook (puxh):** o servidor envia dados quando algo acontece.

Quando usar

- APIs são melhores para consultas on-demand, operações CRUD e leitura/atualização específicas.
- Webhooks são melhores para notificações em tempo real e eventos que exigem reação imediata.

Arquitetura

- APIs **exigem polling** se quiser receber atualizações, já webhooks **evitam** polling e reduzem latência e tráfego.

Polling

Polling é uma técnica em que um sistema fica consultando repetidamente outro sistema, em intervalos regulares, para verificar se há novas informações ou mudanças.

6.1. O que são as Google APIs?

As Google APIs são interfaces que o Google disponibiliza para que aplicações externas **leiam**, **escrevam** e **controlem recursos** dos serviços Google (Gmail, Drive, Sheets).

No contexto do n8n, as Google APIs permitem que um workflow, por exemplo: **leia linhas de uma planilha**, **envie um e-mail**, **faça upload de um arquivo** para um Drive corporativo, **crie um evento** no calendário, ou **consulte dados** no BigQuery, tudo usando nodes específicos ou chamadas HTTP quando necessário.

Observação

O n8n traz nodes prontos para muitos dos serviços do Google. Quando um serviço não tiver node pronto, você pode usar o HTTP Request Node com as mesmas credenciais Google.

6.2. Autenticação (OAuth 2.0)

OAuth 2.0 é um **protocolo de autorização** que permite que uma aplicação (o cliente) obtenha permissão limitada para acessar recursos protegidos em nome de um usuário, sem exigir que o usuário entregue sua senha à aplicação.

Importante: OAuth2 trata **autorização** (delegar acesso), não autenticação. Para autenticação usa-se OpenID Connect (OIDC) que usa OAuth2 como base e adiciona *id_token*.

6.2.1. Função dentro do n8n

- Permitir que o n8n aja em nome do usuário sem armazenar a senha.
- Gerenciar tokens que autorizam as chamadas ao Nodes Google (Sheets, Drive, Gmail, etc)
- **Centralizar credenciais:** a credencial OAuth é criada uma vez, depois todos os Nodes desse tipo usam a credencial.

6.2.2. Por que localhost "da problema"?

http://localhost:5678 só existe dentro da sua máquina, por isso, o Authorization Server (Google) não consegue alcançar esse endereço diretamente pela internet.

O truque que permite a integração em desenvolvimento é que o navegador do usuário atua como ponte: Google redireciona o navegador de volta para o localhost e, como o navegador roda na mesma máquina do n8n, o callback funciona.

6.3. Google Cloud

O Google Cloud é a plataforma de **serviços em nuvem** do Google.

Ele fornece **infraestrutura** (servidores, banco de dados, rede), **ferramentas de machine learning**, **armazenamento de arquivos**, **segurança**, além de acesso programático a praticamente todos os produtos Google por meio de APIs.

No contexto do n8n, o Google Cloud é o **ponto de entrada** obrigatório quando você quer usar os serviços do Google via API. O Google Cloud serve como "ponte oficial" entre seus Workflows e os serviços Google.

Ele é útil porque:

- **Gerencia credenciais:** é no Console do Google Cloud que você cria Client ID e Client Secret para autenticação.
- **Define permissões:** você controla quais dados o n8n pode acessar (apenas planilhas, apenas enviar emails, etc).
- **Centraliza segurança e monitoramento:** você pode acompanhar uso de APIs, cotas de consumo e eventuais erros de autenticação.

Se você quiser que seu Workflow do n8n pegue informações de um formulário do Google Forms e grave em um Google Sheet, **o fluxo só vai funcionar se** no Google Cloud a Sheets API estiver ativada e o n8n tiver uma credencial válida para acessar a planilha.

Google Cloud - Homepage

Escaneie o código QR para acessar o conteudo ou acesse:
https://cloud.google.com/?hl=pt-BR

Acessar

6.3.1. Como funciona

O funcionamento pode ser visto como três passos principais:

1. Habilitar o serviço no Google Cloud

- **Exemplo:** ativar a API do Google Drive dentro do seu projeto no console do Google Cloud Console.

2. Criar credenciais no Google Cloud

- **ID do Cliente:** o ID do Cliente (Client ID) funciona como a identidade da sua aplicação. Ele é um identificador público e único que o Google usa para reconhecer qual aplicação está pedindo acesso à API.

> Esse ID é público no sentido de que pode aparecer nas requisições, mas por si só **não garante acesso**.

- **Chave Secreta do Cliente:** a Chave Secreta do Cliente (Client Secret) funciona como a senha da aplicação, por isso deve ser mantida em total sigilo. Ela atua juntamente ao ID do Cliente para comprar no Google, que é, de fato, sua aplicação que está fazendo a requisição.

3. Configurar credencial no n8n

- No n8n, você cria uma nova credencial do Google e insere os dados gerados no Google Cloud. Durante a primeira conexão, o n8n pede autorização e armazena os tokens de acesso/atualização.
- Depois disso, qualquer Node do n8n que use Google pode consumir diretamente os dados autorizados.

6.4. Hora da Prática: Resume pra mim?

Neste exercício, você será responsável por desenvolver um workflow no n8n que conecte os serviços do Google Docs a um agente de IA.

O objetivo é criar um fluxo inteligente capaz de receber um documento, interpretar seu conteúdo e, automaticamente, gerar um resumo conciso e informativo utilizando o agente.

Dicas:

1. Atenção à autenticação
2. Como o conteúdo do documento chegará ao agente
3. Escreva um prompt claro e educativo

Anotações

7. Aula Prática II: Envio de formulário

7.1. On Form Submission

O On form submission (ou *Form Trigger*) é um gatilho que permite criar um **formulário simples** diretamente dentro do n8n e usá-lo para iniciar um workflow.

Ele é pensado para **prototipagem**, **testes** e **demonstrações rápidas.** Você monta o formulário na interface do n8n, compartilha a URL gerada e, quando alguém preencher, o workflow dispara.

7.1.1. Tipos de campos

O editor do On Form Trigger oferece campos típicos como: **text**, **email**, **number**, **textarea**, **dropdown**, etc. Alguns ambientes permitem marcar campos como *required* (obrigatório) ou incluir placeholders.

Placeholder

Placeholder é um texto temporário exibido em um campo de entrada, fornecendo uma dica ou exemplo sobre o que o usuário deve digitar.

7.1.2. Fluxo básico

1. **Criação do formulário:** você adiciona o Form Trigger e define os campos do formulário.
2. **URL pública/preview:** o n8n gera uma URL ou um preview no editor.
3. **Envio:** quando um usuário envia o form, o node dispara o Workflow.
4. **Entrega de dados:** os dados do envio chegam para o próximo Node:
5. **Tratamento posterior:** você usa Nodes subsequentes (Edit Fields, Service Nodes, etc) para armazenar ou enviar informações.

7.2. Form trigger x Produção

O Form Trigger do n8n foi desenvolvido como uma **ferramenta de testes e prototipagem**, e não como uma solução definitiva para formulários em ambientes reais de produção.

Ele funciona muito bem quando queremos validar rapidamente um fluxo, demonstrar como os dados entram em um workflow ou experimentar cenários de integração de forma ágil e prática, sem precisar recorrer a ferramentas externas logo de início.

No entanto, em situações do mundo real, quando falamos de projetos que estarão disponíveis para clientes, o formulário geralmente não será criado dentro do próprio n8n. Nesses casos, ele virá de um serviço externo, como o Typeform, Google Forms, ou qualquer outra plataforma que permita a coleta de dados na web.

7.3. Service Node: Google Sheets

O Google Sheets Node no n8n é uma ferramenta extremamente útil e versátil para integrar planilhas do Google diretamente em fluxos de automação.

Ele permite **ler**, **escrever**, **atualizar** e até mesmo **organizar dados** sem que seja necessário abrir manualmente o Google Sheets. Essa funcionalidade é poderosa porque muitas vezes planilhas são utilizadas como **bases de dados rápidas**, **pontos de controle de processos** ou mesmo como **interfaces de entrada e saída de informações** para equipes que ainda não utilizam sistemas mais robustos.

Funcionalidade

- Dentro do n8n, o Node do Google Sheets facilita a conexão com essas planilhas, garantindo que qualquer alteração feita nelas possa ser usada como gatilho ou que dados processados em outros Nodes sejam registrados automaticamente.

7.4. Desafio: Verificador de Status

Imagine que uma equipe mantém uma planilha no Google Sheets para organizar tarefas. Cada linha representa um item, e uma das colunas indica o status.

Sua tarefa é criar um Workflow no n8n que torne possível acessar as informações desta planilha e passá-las para um Agente de IA que irá resumir as tarefas para você, pontuando: qual a tarefa, quem é o responsável pela tarefa e qual o prazo.

Anotações