Apostila do Curso

Conteúdo e Atividades

Fundamentos do n8n Lite

Fundamentos do N8N Lite

Nome: _____

Quantidade de Aulas
3 aulas

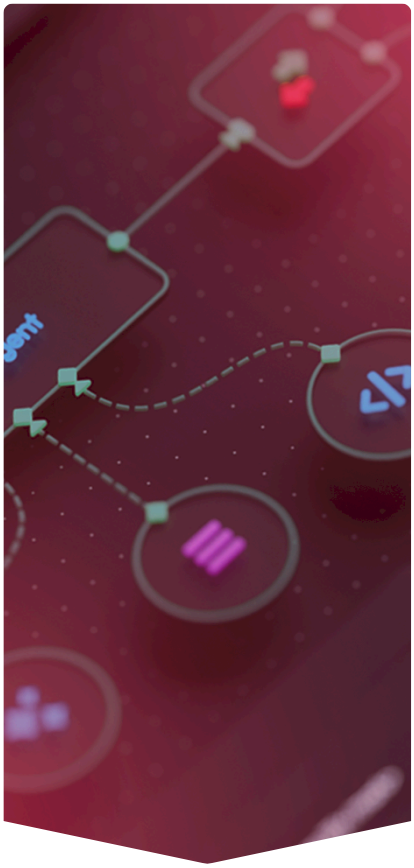
Carga horária
4.5 horas

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 destas três aulas, 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 entender como funciona os workflows e os agentes de IA.



Sumário

1. Introdução ao N8N e automação

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

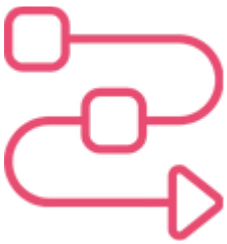
2.1. Workflow

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