

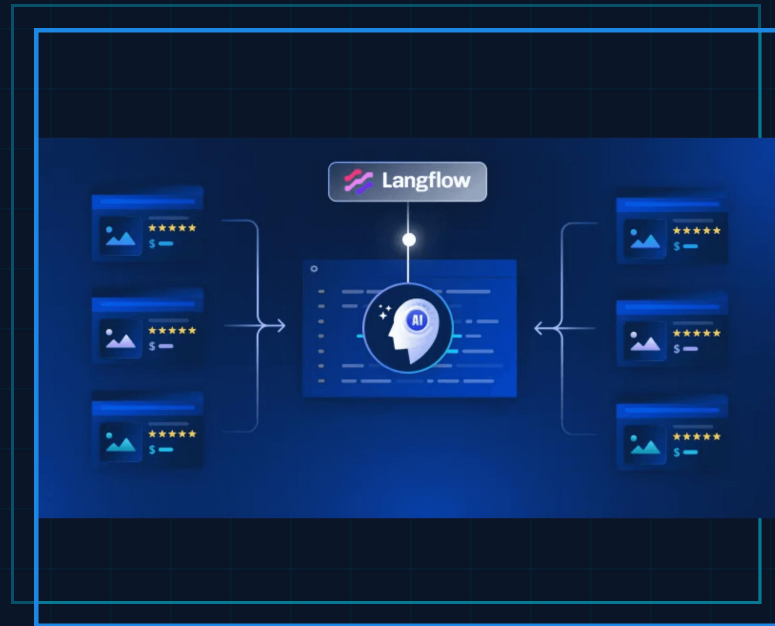
# Minicurso Langflow

## Construindo Aplicações de IA de Forma Visual

Prototipagem rápida com LangChain  
sem código complexo

Mailson Santos

github: <https://github.com/mailsonsantos/minicurso-langflow>



# Langflow Democratiza o Desenvolvimento de IA com Interface Visual

## 01 Interface visual para LangChain

Transforma a complexidade do LangChain em componentes visuais arrastar-e-soltar, eliminando a necessidade de escrever código extensivo para prototipagem.

## 02 Prototipagem 10x mais rápida

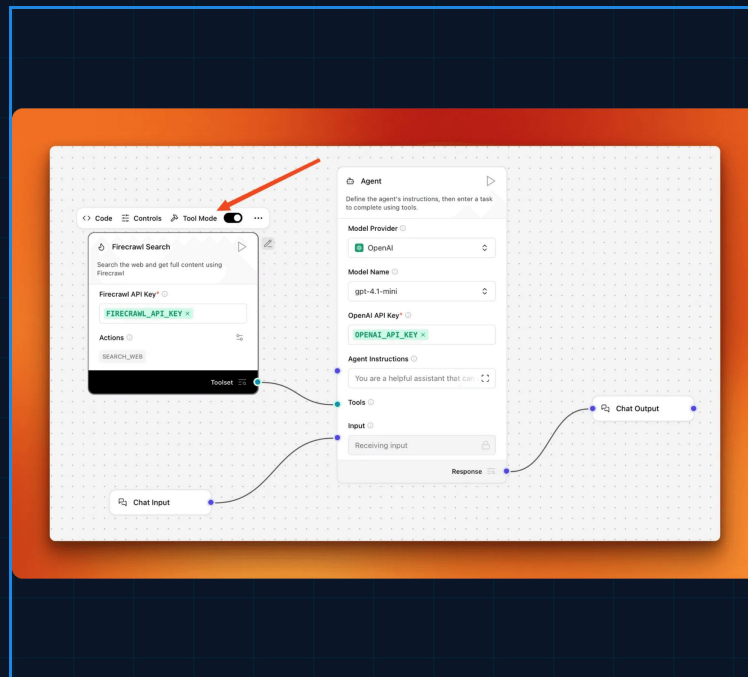
A abordagem low-code/no-code reduz o tempo de desenvolvimento de dias para horas, permitindo iterações rápidas e validação de conceitos.

## 03 Open Source e extensível

Comunidade ativa com mais de 20 mil estrelas no GitHub, garantindo transparência, customização e evolução contínua da plataforma.

## 04 Visualização clara do fluxo

O canvas visual permite entender imediatamente como os dados fluem entre componentes, facilitando debugging e otimização.



# Construiremos 4 Aplicações Práticas de IA Hoje

---



01

## Chatbot Simples (Hello World)

Primeiro contato com a interface, conectando input do usuário diretamente ao modelo Groq para respostas instantâneas.



02

## Sistema com Memória Contextual

Evolução do chatbot básico adicionando componente de memória que permite conversas naturais onde a IA lembra do contexto anterior.



03

## RAG - Conversa com Documentos

Sistema avançado que permite fazer perguntas sobre PDFs e websites, combinando recuperação de informações com geração de respostas contextualizadas.



04

## Agentes Inteligentes com Ferramentas

IA que não apenas responde, mas age usando ferramentas externas como calculadoras, demonstrando capacidade de raciocínio e ação.

# Ambiente Completo em 3 Comandos Docker

## ✓ Pré-requisitos mínimos

Docker, Docker Compose instalados e conta gratuita na Groq para obter API Key (processo leva menos de 5 minutos).

## 📦 Arquitetura docker-compose.yml

Três serviços integrados - Langflow (interface principal na porta 7860), Ollama (embeddings locais) e ChromaDB (armazenamento vetorial).

## > Comando único de inicialização

Sobe toda a infraestrutura em background:

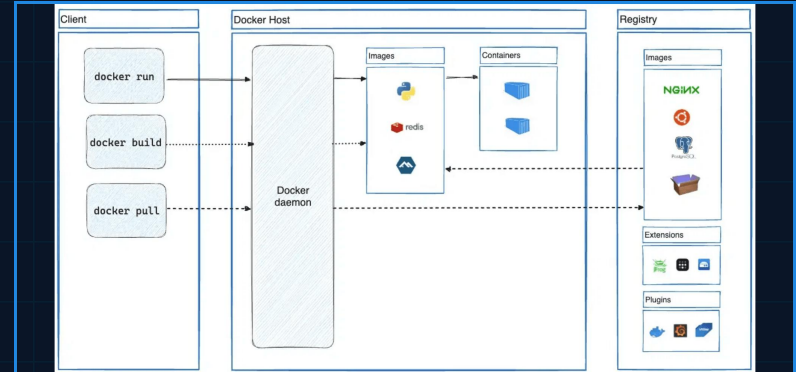
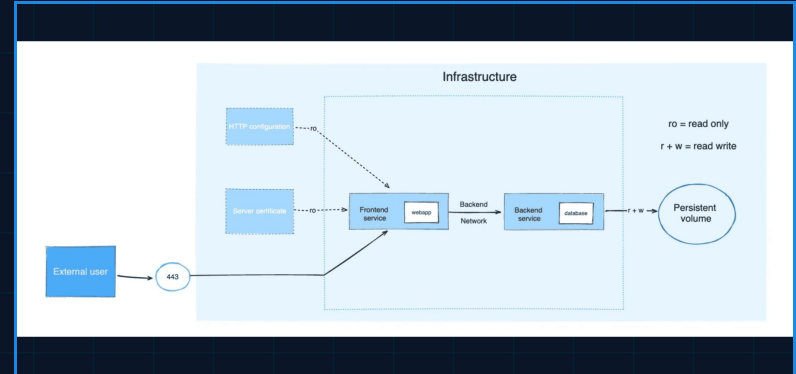
```
docker-compose up -d
```

## 🌐 Acesso imediato

Após 30-60 segundos, acesse [localhost:7860](http://localhost:7860) no navegador e comece a construir fluxos de IA sem instalações adicionais.

## 🗄️ Persistência de dados

Volumes Docker garantem que seus fluxos e configurações sejam preservados entre reinicializações.



# Interface Intuitiva com 3 Áreas Principais

## 📁 Sidebar de Componentes

Biblioteca organizada por categorias (Models, Memory, Loaders, Tools) com mais de 50 componentes prontos para arrastar ao canvas.

## 🔗 Canvas de Trabalho

Área central onde você conecta componentes visualmente, criando o fluxo de dados da sua aplicação de IA com feedback visual em tempo real.

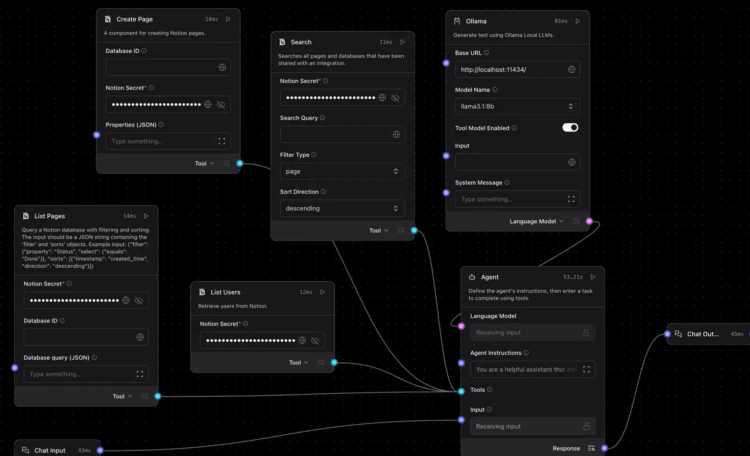
## 🎮 Playground Integrado

Chat de teste embutido que permite validar seu fluxo instantaneamente sem sair da interface, acelerando o ciclo de desenvolvimento.

## </> Exercício 1 - Hello World

Conectar **Chat Input** → **Groq Model** → **Chat Output** em menos de 2 minutos. Configurar API Key e selecionar modelo llama-3.1-70b.

**Validação:** Testar no Playground com perguntas simples confirma que a integração com Groq está funcionando corretamente.



# Memória Resolve a Amnésia dos LLMs

## ⚠ O problema da independência

Por padrão, cada mensagem ao LLM é tratada isoladamente, fazendo com que ele "esqueça" tudo que foi dito anteriormente na conversa.

## ✅ Chat Memory como solução

Componente que armazena histórico de mensagens e automaticamente reenvia contexto relevante em cada nova requisição ao modelo.

## 🧪 Exercício 2 - Teste de Memória

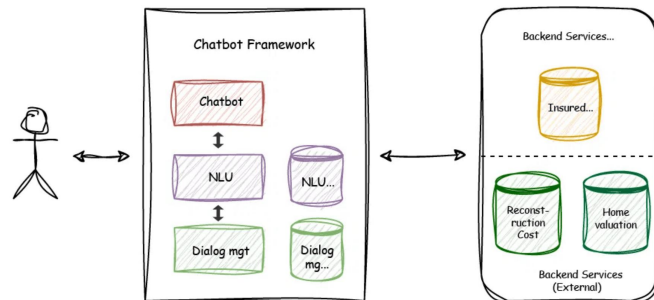
Enviar "Meu nome é João" seguido de "Qual é meu nome?"

→ Sem memória: falha | Com memória: responde corretamente

## ⚖ Custo vs Benefício

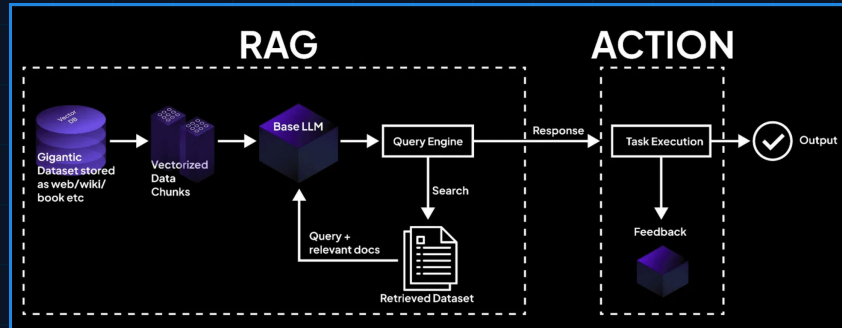
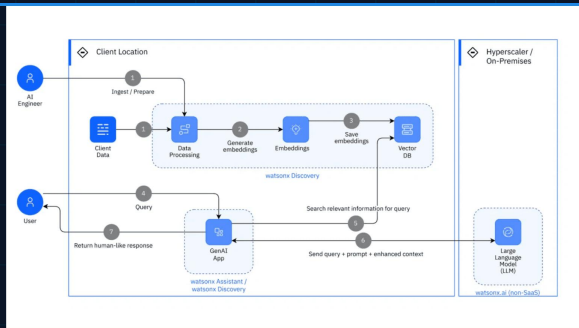
Memória aumenta tokens consumidos (histórico é reprocessado), mas é essencial para conversas naturais e assistentes virtuais eficazes.

## Chatbot Architecture



# RAG Dá Conhecimento Privado à IA

**Limitação dos LLMs:** Modelos só conhecem dados de treinamento (até data de corte) e não têm acesso a documentos internos, PDFs corporativos ou informações atualizadas.



## Fluxo RAG Completo

**Load** (carregar documento) → **Split** (dividir em chunks de 500-1000 tokens) → **Embed** (converter em vetores com Ollama) → **Store** (ChromaDB) → **Retrieve** (buscar relevantes) → **Generate** (resposta contextualizada)

## 🔧 Embeddings Locais com Ollama

Modelo `nomic-embed-text` transforma texto em vetores de 768 dimensões, permitindo busca semântica sem custos de API externa.

## 🗄️ ChromaDB como Memória Vetorial

Banco de dados especializado que encontra chunks mais similares à pergunta do usuário em milissegundos, mesmo com milhares de documentos.

🔥 **Exercício 3:** Upload de manual técnico ou artigo científico, seguido de perguntas específicas que só podem ser respondidas com informações do documento.

# Agentes Transformam LLMs em Atores Autônomos

## 🤖 Além da geração de texto

Agentes combinam raciocínio do LLM com capacidade de executar ações no mundo real através de ferramentas (tools).

## 🔄 Ciclo Reasoning → Action

O modelo analisa a pergunta, decide qual ferramenta usar, executa a ação, recebe o resultado e formula resposta final.

## ⚠️ LLMs são péssimos em matemática

Perguntar "Quanto é  $47382 \times 923$ ?" ao LLM puro resulta em respostas incorretas por limitações de raciocínio aritmético.

Exemplo: LLM pode responder -43.7 milhões  
Resposta correta: 43,733,586

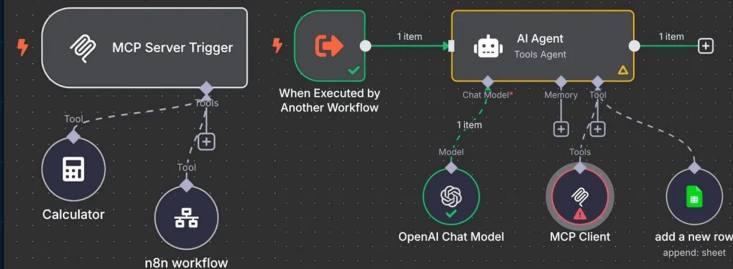
## 🧮 Calculadora como Tool

Conectar ferramenta Calculator ao agente permite que ele delegue cálculos complexos, retornando resultados precisos.

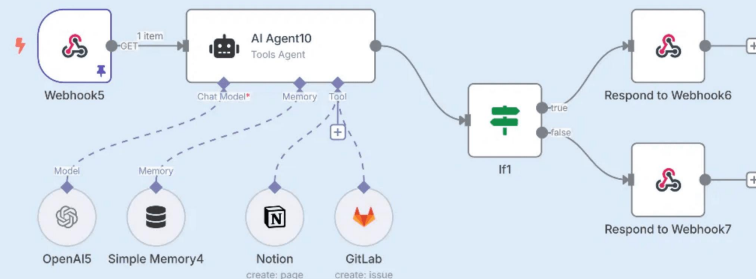
## EXERCÍCIO 4: Agente Matemático

Comparar resposta do chatbot simples vs agente com calculadora para operações matemáticas, demonstrando ganho de precisão.

# BUILD AI AGENTS



## Single Agent + Tools + Router





# Do Protótipo à Produção em Minutos



## Exportação JSON

Salvar fluxo completo como arquivo JSON permite versionamento, compartilhamento com equipe e importação em outras instâncias Langflow.



## API REST automática

Cada fluxo gera endpoint HTTP instantaneamente, permitindo integração com qualquer linguagem (Python, JavaScript, Java, etc.).



## Integração Python

Código de 10 linhas usando biblioteca [requests](#) para enviar mensagens ao fluxo e receber respostas programaticamente.

Arquivo: `teste_api.py`



## Widget HTML embarcável

Arquivo [teste\\_index.html](#) demonstra como incorporar chat em websites existentes com apenas 3 linhas de código iframe.

```
<iframe src="langflow-url"></iframe>
```



## Escalabilidade

Fluxos podem ser deployados em servidores de produção, Kubernetes ou Langflow Cloud para atender milhares de usuários simultâneos.



## Transição Suave


Do protótipo visual ao sistema em produção sem reescrever código, mantendo a mesma lógica de fluxo.

# Próximos Passos e Recursos



## Documentação Oficial

Tutoriais detalhados, referência de componentes e guias de troubleshooting para resolver problemas e aprofundar conhecimentos.

 <https://docs.langflow.org>



## Comunidade Discord

Mais de 15 mil desenvolvedores ativos para tirar dúvidas, compartilhar fluxos e descobrir casos de uso inovadores.

 15,000+ membros ativos



## Marketplace de Componentes

Biblioteca crescente de componentes customizados da comunidade para integrações específicas (Notion, Slack, WhatsApp).



## Certificação e Aprofundamento

Cursos avançados sobre otimização de prompts, fine-tuning de modelos e arquiteturas multi-agente.



## Desafio Prático

Criar agente que busca informações na Wikipedia e responde perguntas, combinando os conceitos de agentes e ferramentas aprendidos hoje.

**Obrigado! Boas construções com Langflow!**