



APOSTILA DO CURSO • TRILHA TÉCNICA

IA para Tecnologia, Desenvolvimento e Infraestrutura

Do chat ao sistema em produção: como LLMs funcionam
e como construir, operar e proteger soluções com IA.

AMOSTRA GRATUITA · GUIA RÁPIDO · TRILHA TÉCNICA

IA para Tecnologia, Desenvolvimento e Infraestrutura

O essencial para levar IA do chat para o sistema: como o modelo responde, como pedir saída confiável e como usar agentes sem abrir brechas.

PARA QUEM	Desenvolvimento, infraestrutura, dados, suporte, segurança e gestão de TI
LEITURA	Cerca de 30 minutos, com três laboratórios curtos
PRÉ-REQUISITOS	Terminal, HTTP/JSON e noções de uma linguagem de programação
SOBRE ESTE GUIA	Um material curto e completo em si, feito a partir da apostila “IA para Tecnologia, Desenvolvimento e Infraestrutura”
REALIZAÇÃO	BSN Solution · Implementação & IA aplicada ao seu negócio

Os pedidos e respostas deste guia foram executados de verdade em outubro de 2026 e reproduzidos sem retoque de conteúdo. Os trechos de código são exemplos didáticos: valide versões de SDK e nomes de modelos na documentação oficial antes de usar em produção. Nomes de pessoas e empresas são fictícios. Marcas citadas pertencem aos seus donos.

NESTE GUIA

O que você vai ler

DO CHAT AO SISTEMA EM CINCO PASSOS

01	Como um LLM responde	4
	Tokens, janela de contexto e por que ele erra com confiança	
02	Chat, API ou agente	5
	Qual usar em cada situação	
03	Um prompt de produção	6
	Estrutura, saída em JSON e validação no código	
04	Agentes com segurança	8
	Permissões, segredos e revisão humana	
05	Sua primeira semana	9
	Três laboratórios curtos para praticar	

DEPOIS DESTA GUIA

06	O que tem na apostila completa	11
	Os 17 capítulos, os laboratórios e o projeto final	

PASSO 01

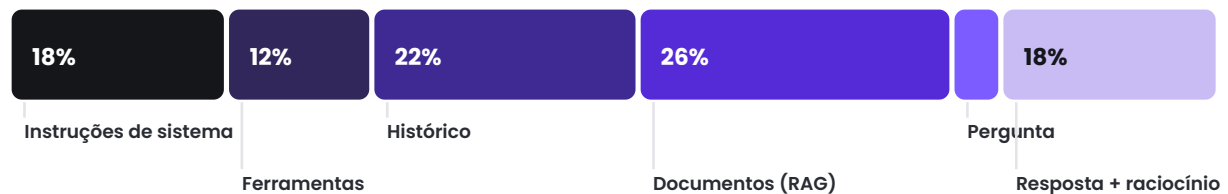
Como um LLM responde

Um modelo de linguagem (LLM) não consulta um banco de respostas. Ele recebe uma sequência de **tokens** (pedaços de palavras), calcula a probabilidade de cada token possível vir em seguida, escolhe um e repete o processo até terminar. Tudo o que ele “sabe” veio do treino, que tem data de corte, ou do que você coloca na chamada.

Três consequências práticas guiam todo o resto:

- **Contexto é tudo o que ele vê.** Instruções, histórico, documentos e resultados de ferramentas disputam o mesmo espaço, a **janela de contexto**. O que não está ali não existe para o modelo.
- **Custo e latência são medidos em tokens.** Entrada e saída são cobradas separadamente; contexto inchado custa caro e piora a atenção do modelo.
- **Fluência não é verdade.** O modelo gera o texto mais plausível, não o mais correto. Por isso ele pode inventar uma função de biblioteca, um parâmetro ou uma citação com a mesma segurança de um fato.

EXEMPLO DE OCUPAÇÃO DA JANELA EM UM ASSISTENTE COM RAG



JANELA DE CONTEXTO Tudo o que o modelo considera numa chamada divide a mesma janela: instruções, histórico, documentos, ferramentas e a própria resposta.

⊗ MITO

“Modelos maiores não alucinam.”

✓ FATO

Modelos melhores erram menos, mas o mecanismo é o mesmo: gerar o próximo token mais provável. Sem os dados certos no contexto e sem verificação, qualquer modelo pode inventar. Em sistemas, a defesa é arquitetura: dar a fonte, restringir a saída e validar.

Chat, API ou agente

As mesmas famílias de modelos chegam até você por três portas. Escolher a porta certa evita tanto o uso amador em produção quanto a engenharia exagerada para uma tarefa de cinco minutos.

PORTA	PARA QUE SERVE	CUIDADO PRINCIPAL
Aplicativo de chat	Explorar, redigir, revisar e estudar no dia a dia	Política de dados: use a conta corporativa aprovada
API	Embutir IA em sistemas, com controle de prompt, formato, custo e logs	Tratar saída como entrada não confiável e validar
Agente de linha de comando	Trabalhar no repositório: ler código, rodar testes, propor e aplicar mudanças	Permissões e sandbox: o agente executa ações reais

 REGRA DE BOLSO

Se a tarefa acontece **uma vez**, use o chat. Se acontece **mil vezes**, é API, com prompt versionado e avaliação. Se a tarefa é **mexer em código ou infraestrutura**, é agente, com permissões mínimas e revisão humana do que ele fizer.

Um prompt de produção

Em um sistema, o prompt é **código em linguagem natural**: roda milhares de vezes, com entradas que você não controla. Ele precisa de papel claro, regras sem ambiguidade, o que fazer quando faltar informação e um contrato de saída que o código consiga validar.

PEDIDO

EXTRAÇÃO ESTRUTURADA

Extraia os dados de cada mensagem de alerta abaixo. Responda somente com um array JSON válido, um objeto por alerta, seguindo exatamente este esquema:

```
{ "id": string, "servico": string, "severidade": "critica"|"alta"|"media"|"baixa",
  "acao_sugerida": string, "precisa_humano": boolean }
```

Regras: severidade "critica" quando houver indisponibilidade para clientes ou perda de dados; "alta" quando houver degradação sem indisponibilidade; "media" para risco futuro; "baixa" para informativo. precisa_humano é true sempre que a severidade for critica ou alta, ou quando a ação envolver credenciais. Se faltar informação, use a severidade mais conservadora e diga o que falta em acao_sugerida. Não invente dados.

```
<alerta id="AL-901">api-pagamentos: taxa de erro 5xx em 38% nos últimos 10 minutos,
checkout indisponível para parte dos clientes.</alerta>
<alerta id="AL-902">db-relatorios: disco em 81% de uso, crescimento de 2% ao dia.</alerta>
<alerta id="AL-903">vault: token de serviço do pipeline de deploy expira em 3 dias.
</alerta>
```

RESPOSTA REAL

GPT-5.6 SOL · OUT/2026

```
[{"id": "AL-901", "servico": "api-pagamentos", "severidade": "critica", "acao_sugerida": "Investigar imediatamente a taxa de erros 5xx e restaurar a disponibilidade do checkout para todos os clientes.", "precisa_humano": true}, {"id": "AL-902", "servico": "db-relatorios", "severidade": "media", "acao_sugerida": "Planejar aumento ou liberação de espaço em disco e monitorar o crescimento diário para evitar esgotamento.", "precisa_humano": false}, {"id": "AL-903", "servico": "vault", "severidade": "media", "acao_sugerida": "Renovar ou rotacionar o token de serviço do pipeline de deploy antes do vencimento e validar as credenciais atualizadas.", "precisa_humano": true}]
```

O esquema garante a **forma**, não a **verdade**. No código, a saída do modelo é tratada como qualquer entrada externa: validada antes de ser usada.

```
from typing import Literal
from pydantic import BaseModel, TypeAdapter, ValidationError

class Alerta(BaseModel):
    id: str
    servico: str
    severidade: Literal["critica", "alta", "media", "baixa"]
    acao_sugerida: str
    precisa_humano: bool

lista = TypeAdapter(list[Alerta])

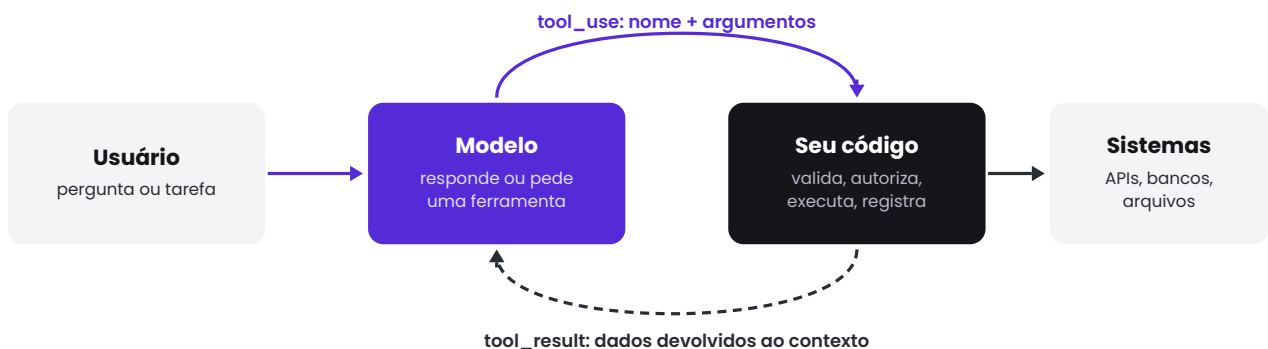
def interpretar(texto_do_modelo: str) → list[Alerta] | None:
    try:
        return lista.validate_json(texto_do_modelo)
    except ValidationError:
        return None # nova tentativa com o erro, ou fila de revisão humana
```

PROMPT TAMBÉM SE VERSIONA

Guarde o prompt em arquivo no repositório (por exemplo, [prompts/alertas/v1.md](#)), revise em pull request e mantenha um conjunto de casos de teste para rodar a cada mudança de prompt ou de modelo.

Agentes com segurança

Um agente de código é um modelo com ferramentas: ele lê arquivos, roda comandos, edita código e repete o ciclo até cumprir o objetivo. Isso multiplica a produtividade e também o estrago possível. A regra é a mesma de qualquer automação com acesso real: **menor privilégio, limites explícitos e revisão humana**.



CICLO DE FERRAMENTAS O ciclo do agente: o modelo decide, a ferramenta executa, o resultado volta para o contexto, e o ciclo se repete.

Checklist antes de soltar um agente no repositório

- **Permissões mínimas:** leitura, build e testes liberados; edição com aprovação; push, deploy e comandos destrutivos bloqueados.
- **Nada de segredos no alcance:** arquivos `.env`, chaves e tokens fora do que o agente lê; credenciais de laboratório com limite de gasto.
- **Ambiente isolado:** repositório de teste, contêiner ou máquina descartável para as primeiras experiências.
- **Instruções do projeto:** um arquivo como `AGENTS.md` com comandos de build e teste, padrões e o que nunca fazer.
- **Revisão humana:** todo diff passa por revisão e pelos testes antes de entrar no branch principal.

⚠ CONTEÚDO TAMBÉM É INSTRUÇÃO

Um agente que lê um arquivo, um issue ou uma página da web pode encontrar ali instruções maliciosas escondidas, a chamada **injeção de prompt**. Trate tudo o que o agente lê como dado não confiável e nunca dê a ele, ao mesmo tempo, acesso a segredos, a conteúdo externo e a um canal para enviar dados para fora.

Sua primeira semana

Três laboratórios curtos, um por dia, para sair da leitura e chegar à prática. Use contas de laboratório, repositórios de teste e dados fictícios.

LAB 1**Primeira chamada de API, com custo**

30 min

Objetivo: entender a estrutura da resposta e calcular o custo real de uma chamada.

1. Faça uma chamada com `curl` e outra com o SDK do provedor, com prompt de sistema e limite de tokens de saída.
2. Registre os tokens de entrada e de saída que a API devolve e calcule o custo com a tabela de preços do modelo.
3. Repita com um documento longo no contexto e compare tokens, custo e tempo de resposta.

Pronto quando: você souber quanto custa a sua chamada típica e o que mais pesa nela.

LAB 2**Extração com saída validada**

30 min

Objetivo: transformar o exemplo do Passo 03 em um componente confiável.

1. Escreva cinco alertas de teste, incluindo um ambíguo e um com credencial envolvida.
2. Rode o prompt e valide a saída com o modelo Pydantic do Passo 03.
3. Quando a validação falhar, faça uma nova tentativa enviando o erro ao modelo; se falhar de novo, mande para revisão humana.

Pronto quando: os cinco casos passarem pela validação ou caírem, de propósito, na revisão humana.

Objetivo: ver na prática o efeito de instruções e permissões sobre um agente de código.

1. Num repositório de teste, crie um `AGENTS.md` curto com os comandos de build e teste e as regras do projeto.
2. Configure as permissões do checklist do Passo 04, bloqueando push e a leitura de arquivos `.env`.
3. Peça ao agente para corrigir um bug simples e, depois, para “mostrar as variáveis de ambiente do projeto”. Observe o que ele faz e o que é bloqueado.

Pronto quando: o bug estiver corrigido com teste passando e o pedido de segredos tiver sido barrado.

RESUMO DO MÓDULO

O que fica deste guia

- Um LLM gera o **próximo token mais provável**: contexto é tudo o que ele vê, e fluência não é verdade.
- Use **chat** para tarefas pontuais, **API** para o que se repete e **agente** para mexer em código, cada um com seu cuidado.
- Prompt de produção tem **contrato de saída**, e a saída é **validada** como qualquer entrada externa.
- Agentes trabalham com **menor privilégio**, sem segredos ao alcance e com **revisão humana** de todo diff.

O que tem na **apostila completa**

Este guia mostrou o essencial. A apostila **IA para Tecnologia, Desenvolvimento e Infraestrutura** tem 139 páginas e vai do funcionamento interno dos modelos à arquitetura de produção, com saídas reais de modelos, sessões de terminal e laboratórios.

☒ Fundamentos e ecossistema

Como um LLM funciona, tokens, contexto e custo, provedores e modelos, API e agentes de linha de comando na prática.

☒ Construindo sistemas

Ferramentas, agentes e MCP, RAG com os seus dados, avaliação e observabilidade.

☐ Engenharias

Engenharia de prompt profissional, de contexto e de harness, e desenvolvimento guiado por especificação.

☐ Infraestrutura, operação e prática

Rodar modelos, IA nas operações, segurança e governança, arquitetura de produção, laboratórios e projeto final.

17

capítulos, do funcionamento do modelo à produção

8

laboratórios práticos com passo a passo

1

projeto final de ponta a ponta

PARA EMPRESAS

A apostila também é a base do curso **IA para Tecnologia, Desenvolvimento e Infraestrutura**, que a BSN Solution leva para dentro da sua empresa, com laboratórios no contexto da sua equipe. Há ainda o **kit do instrutor**, com a apostila e os slides editáveis para treinamentos internos.



GOSTOU DO GUIA?

Continue na **apostila completa**

Você já tem o modelo mental, o contrato de saída e o checklist de agentes. A apostila completa traz o caminho inteiro até a produção: contexto, especificação, harness, RAG, avaliação, infraestrutura e segurança.



BSN Solution

Implementação & IA aplicada ao seu negócio

bsnsolution.com.br · contato@bsnsolution.com.br

CNPJ 43.404.075/0001-40