Capítulo 15 – APIs e Prompts Básicos

Aléssio Miranda Jr.

Setembro - 2026/2

Capítulo em Revisão

Este conteúdo está sendo revisado. Algumas informações podem estar preliminares, conter erros de formatação ou sofrer alterações na versão final.

O que você verá neste capítulo

Entramos no Arco 3: a **IA Generativa Aplicada**. Nos arcos anteriores, treinamos modelos do zero (Arco 1) e consumimos modelos pré-treinados localmente (Arco 2). Agora, daremos o salto para o paradigma que domina o mercado: consumir modelos gigantes via **APIs na nuvem**. Neste capítulo, exploraremos as APIs dos principais provedores (OpenAI, Google Gemini, Anthropic), formalizaremos a anatomia de um Prompt, e construiremos nossas primeiras chamadas HTTP para gerar texto com Large Language Models.

1 O Paradigma das APIs de LLMs

Modelos como GPT-4, Gemini e Claude são enormes demais para rodar no laptop de um estudante. O GPT-4 possui estimados 1,7 trilhão de parâmetros — seria necessário um cluster de servidores com centenas de GPUs A100 para executá-lo. Por isso, esses modelos são oferecidos como **serviços na nuvem** (AIaaS — *AI as a Service*), acessíveis via chamadas HTTP.

O fluxo é simples: você envia um texto (o *Prompt*) via requisição HTTP para o servidor do provedor, o servidor processa o texto através da rede neural gigante, e retorna a resposta gerada em formato JSON.

• Teoria: title=Economia das APIs de LLMs

Os provedores cobram por **token** processado (entrada + saída):

- **GPT-4o** (OpenAI): $2.50/1M tokens de entrada, $10.00/1M tokens de saída.
- **Gemini 1.5 Pro** (Google): $1.25/1M tokens de entrada, $5.00/1M tokens de saída.
- **Claude 3.5 Sonnet** (Anthropic): $3.00/1M tokens de entrada, $15.00/1M tokens de saída.

Para referência: 1 milhão de tokens ≈ 750.000 palavras ≈ 3.000 páginas de texto. Uma consulta típica (500 tokens de entrada + 300 de saída) custa frações de centavo.

2 A Anatomia de um Prompt

Um Prompt não é apenas "uma pergunta para a IA". Ele é uma instrução estruturada composta por componentes com papéis distintos:

- **System Prompt:** Define o "papel" e as restrições globais da IA. Ex: "Você é um assistente jurídico especializado em direito trabalhista brasileiro. Responda sempre em português formal."
- **User Prompt:** A mensagem do usuário final. Ex: "Quais são os direitos de um trabalhador demitido sem justa causa?"
- **Assistant Prompt:** Respostas anteriores da IA (usadas para manter o contexto da conversa).

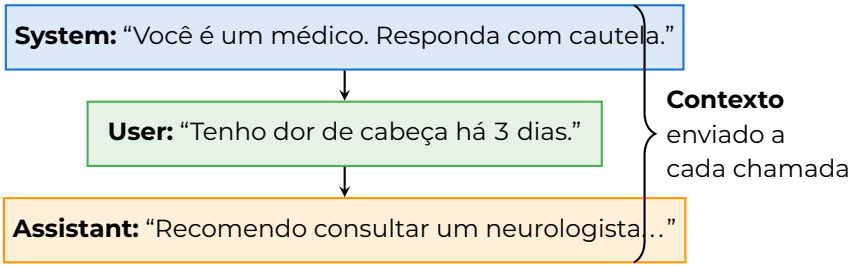

Figura 1: Estrutura de mensagens enviadas à API. O System Prompt persiste em todas as chamadas, estabelecendo o comportamento global do modelo.

3 Primeira Chamada à API (Python)

▶ Prática: title=Chat Completion com a API da OpenAI

```python
from openai import OpenAI

client = OpenAI(api_key="sk-...")

resposta = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "system",
         "content": "Voce e um professor de IA."},
        {"role": "user",
         "content": "Explique Backpropagation em 3 linhas."}
    ],
    temperature=0.7,
    max_tokens=200
)

print(resposta.choices[0].message.content)
```

3.1 Parâmetros Fundamentais

Tabela 1: Parâmetros de controle da geração de texto via API.

Parâmetro	Range	Padrão	Efeito
Temperature	0.0–2.0	1.0	T=0: determinístico. T=1: criativo. T>1.5: caótico.
Max Tokens	1–128K	Modelo	Limite de tokens na resposta.
Top-P	0.0–1.0	1.0	Nucleus Sampling: ignora tokens improváveis.
Frequency Penalty	−2.0–2.0	0	Penaliza repetições de palavras.

▷ Exemplo: title=Temperature na Prática

- $T = 0.0$ — Extração de dados, classificação, JSON estruturado (resposta precisa e reprodutível).
- $T = 0.7$ — Chatbots, assistentes, resumos (equilíbrio entre criatividade e coerência).
- $T = 1.5$ — Geração criativa de texto, brainstorming (alta variabilidade, risco de incoerência).

△ Importante: title=Segurança das Chaves de API

A chave de API (`sk-...`) é como uma **senha de cartão de crédito**. Quem possuir essa chave pode fazer chamadas ilimitadas na sua conta, gerando custos financeiros reais. **Nunca** comite a chave no Git. Use variáveis de ambiente:

```
import os
client = OpenAI(api_key=os.environ["OPENAI_API_KEY"])
```

4 Provedores Alternativos

O ecossistema de APIs de LLMs é competitivo. Além da OpenAI, os principais provedores são:

- **Google Gemini:** API gratuita com limite generoso para uso pessoal. Suporte nativo a multimodalidade (texto + imagem + áudio + vídeo).
- **Anthropic (Claude):** Destaque em segurança e raciocínio longo. Janela de contexto de até 200K tokens (um livro inteiro).
- **Modelos Open-Source via Ollama:** Ferramentas como `ollama` permitem rodar modelos como LLaMA 3 e Mistral **localmente**, sem custo e sem enviar dados para a nuvem. Ideal para dados sensíveis.

✔ Takeaways

- LLMs gigantes (GPT-4, Gemini, Claude) são consumidos como **serviços na nuvem** via chamadas HTTP — o paradigma AIaaS (*AI as a Service*).
- Os provedores cobram por **token** processado (entrada + saída). Controlar o custo exige monitorar o número de tokens enviados e recebidos.
- Um Prompt é composto por três papéis: **System** (persona e restrições globais), **User** (mensagem do usuário) e **Assistant** (respostas anteriores para manter contexto).
- O parâmetro **Temperature** controla a aleatoriedade: $T{=}0$ é determinístico (ideal para extração de dados), $T{=}0.7$ equilibra criatividade e coerência, $T{>}1.5$ é caótico.
- A chave de API é equivalente a uma **senha de cartão de crédito**. Jamais comite no Git — use variáveis de ambiente (`os.environ`).
- Modelos open-source (LLaMA, Mistral) podem ser executados **localmente** via Ollama, sem custo e sem enviar dados à nuvem.

5 Conclusão

A primeira chamada HTTP para um LLM é o “Hello World” da IA Generativa. Na próxima aula, evoluiremos da pergunta simples para a **Engenharia de Prompts Sistêmica**: técnicas avançadas para extrair o máximo de qualidade e estrutura das respostas, incluindo *Few-Shot Prompting*, *Chain-of-Thought* e extração de dados em formato JSON estruturado.

✎ Questões: Autoavaliação

1 **[Direta]** Explique a diferença funcional entre os três papéis de mensagem (System, User, Assistant) enviados à API de um LLM. Por que o System Prompt persiste em todas as chamadas?

2 **[Direta]** Qual é o efeito prático de configurar `temperature=0.0` em uma chamada à API? Em que tipo de aplicação corporativa essa configuração é obrigatória?

3 **[Direta]** Compare os modelos de precificação por token dos provedores OpenAI, Google e Anthropic. Se uma aplicação processa 10 milhões de tokens de entrada por mês, qual provedor seria mais econômico?

4 **[Reflexiva]** Uma empresa de saúde precisa enviar prontuários médicos confidenciais para análise por IA. Discuta as implicações éticas e legais (LGPD) de enviar esses dados para a API da OpenAI na nuvem versus executar um modelo local via Ollama.

5 **[Reflexiva]** Se a chave de API de uma startup vazar acidentalmente em um repositório público do GitHub, quais seriam as consequências financeiras e de segurança imediatas? Que medidas preventivas e reativas a equipe deveria adotar?

Lab. 15: APIs e Prompts Básicos na Prática

Capítulo em Revisão

Este conteúdo está sendo revisado. Algumas informações podem estar preliminares, conter erros de formatação ou sofrer alterações na versão final.

⋆ Objetivo do Laboratório

Neste laboratório, vamos nos conectar pela primeira vez a um modelo avançado na nuvem. O objetivo é configurar um ambiente Python que consome a API oficial do Google Generative AI (Gemini) ou OpenAI, enviar uma pergunta técnica e recuperar a resposta com segurança (sem vazar credenciais).

6 Atividade Prática

1. Configuração do Ambiente e Chaves

1. Obtenha uma API Key válida num provedor moderno (ex: Google AI Studio com o Gemini, que é gratuita para uso moderado, ou OpenAI).
2. No diretório do laboratório, crie o arquivo `.env` contendo:

```
API_KEY=sua-chave-aqui-sem-aspas
```

3. Verifique que o `.env` está listado no seu arquivo `.gitignore`.
4. Instale a biblioteca do provedor e a de leitura do ambiente:

```
pip install google-generativeai python-dotenv
```

2. O Código: Conectando à API

Crie o arquivo `consumo_api.py`:

```
import os
import google.generativeai as genai
from dotenv import load_dotenv

# 1. Carregando variáveis sensíveis do arquivo .env
load_dotenv()
```

```
API_KEY = os.getenv("API_KEY")

# 2. Configurando o SDK do Google
genai.configure(api_key=API_KEY)

# 3. Instanciando o Modelo
modelo = genai.GenerativeModel('gemini-1.5-flash')

# 4. A Requisição
prompt = "Me explique em um único parágrafo o que é REST API e como ela se comunica."
print(f"Enviando o prompt: '{prompt}'...")

# 5. Resposta
resposta = modelo.generate_content(prompt)
print("\n--- Resposta da IA ---")
print(resposta.text)
```

3. Entrega via Git

1. Execute o arquivo usando `python consumo_api.py` e verifique se a resposta textual condiz com uma inteligência de ponta.
2. Certifique-se de que o arquivo `.env` **NÃO** apareça ao rodar `git status`.
3. Adicione as mudanças no rastreamento: `git add consumo_api.py`.
4. Realize o commit no repositório local: `git commit -m "Cria script básico de consumo de LLM na nuvem usando dotenv"`.
5. Envie para o servidor: `git push origin main`.

7 Critério de Avaliação

O código `consumo_api.py` deve estar no repositório Git sem nenhuma API Key *hardcoded* (`.env` devidamente ignorado). A sua submissão demonstrará a capacidade de abstração do *cloud computing* para inferência em IA na sua máquina local.

✔ Takeaways

- Você executou sua **primeira chamada HTTP para um LLM na nuvem** — o "Hello World" da IA Generativa.
- Aprendeu a proteger chaves de API usando `python-dotenv` e `.gitignore` — a mesma prática usada em toda empresa de tecnologia.
- O modelo `gemini-1.5-flash` respondeu em poucos segundos usando a infraestrutura do Google — sem GPU local, sem treinamento, sem custo (tier gratuito).
- O fluxo `load_dotenv()` → `configure(api_key)` → `generate_content()` é o padrão que você usará em **todos** os próximos labs.

◇ Informação

Gancho para a Próxima Aula: Você fez uma pergunta simples e recebeu uma resposta livre. Mas e se precisar extrair dados **estruturados** (JSON) de textos caóticos? Na próxima aula, dominaremos a **Engenharia de Prompts Sistêmica**: System Prompts, JSON Mode e temperatura controlada para transformar o LLM em um extrator de dados corporativo.