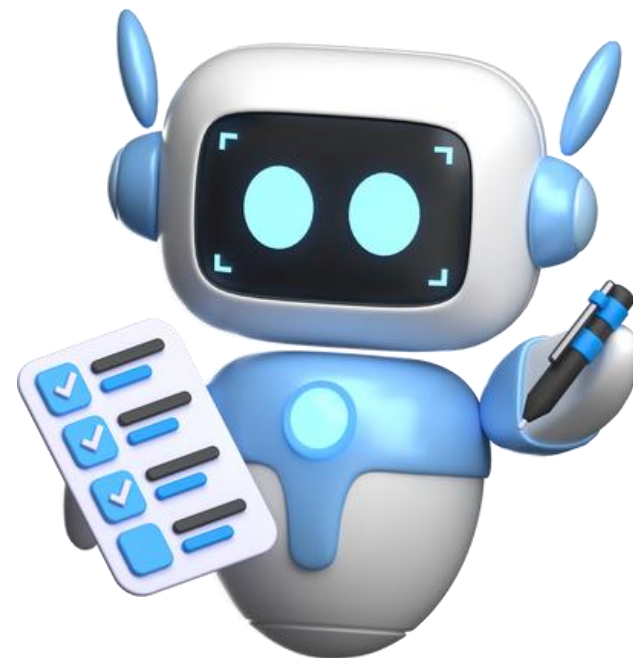


Curso Bacharelado em
Engenharia de Software

Inteligência Artificial e Machine Learning

Prof. Tiago Ruiz

Aula 13/11/2025

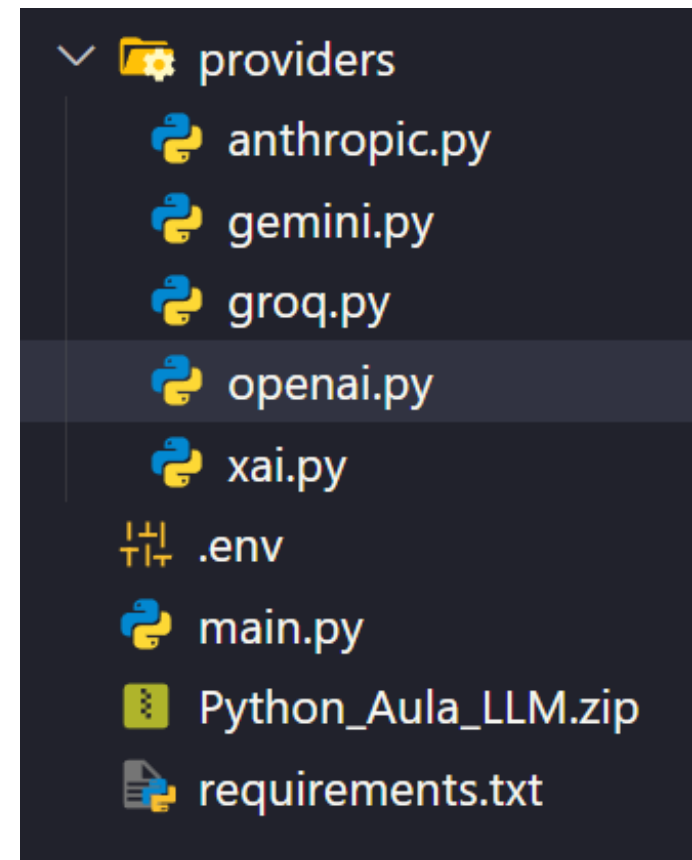


Anthropic

API da Anthropic (Claude)

Continuando a partir da nossa ultima aula aonde criamos toda estrutura para aprender como consumir diversas LLMs, vamos iniciar com a **Anthropic**, mas antes baixe o arquivo disponível no AVA aonde eu fiz algumas modificações no nosso código a partir da ultima aula. Observe que estruturei um pouco melhor os arquivos para usarmos no projeto.

- Criei o arquivo Providers
- Dentro de cada arquivo, estruturei o nosso client
- Criei uma função “Test” com as configurações dos modelos.



Anthropic

API da Anthropic (Claude)

Antes de mais nada, vamos descomentar o campo da anthropic e em seguida vamos ao site da anthropic buscar uma chave para o acesso:

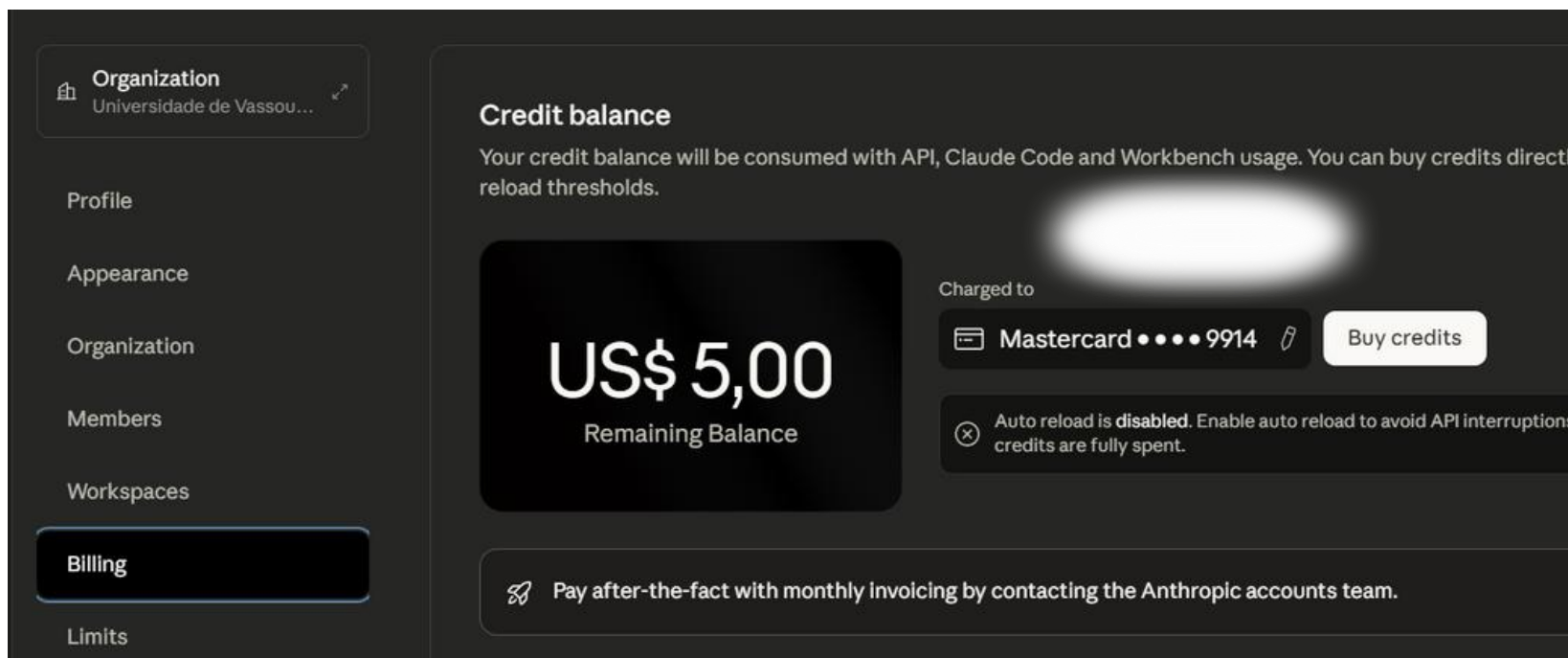
```
1
2  #OpenAI:
3  # from providers import openai
4  # print(openai.test().output_text)
5
6  #Anthropic:
7  from providers import anthropic
8
9  #Xai:
10 # from providers import xai
```

Anthropic

API da Anthropic (Claude)

Ao adquirir uma nova chave na Anthropic, você pode começar a criar a sua organização, e em seguida criar uma chave de acesso para a API.

Baixe a chave de acesso no AVA no arquivo **Anthropic.txt**, copie e cole no arquivo de variáveis de ambiente (.env) do nosso sistema.



The screenshot displays the Anthropic account management interface. On the left, a sidebar menu includes options for Organization, Profile, Appearance, Organization, Members, Workspaces, Billing (highlighted), and Limits. The main content area is titled 'Credit balance' and explains that the balance is used for API, Claude Code, and Workbench usage. It shows a current balance of US\$ 5,00. Below this, it indicates the payment method is a Mastercard ending in 9914, with a 'Buy credits' button. A warning message states that auto-reload is disabled. At the bottom, there is a note about paying after-the-fact with monthly invoicing by contacting the Anthropic accounts team.

Organization
Universidade de Vassou...

Credit balance
Your credit balance will be consumed with API, Claude Code and Workbench usage. You can buy credits directly or set reload thresholds.

US\$ 5,00
Remaining Balance

Charged to
Mastercard •••• 9914 [Buy credits](#)

⊗ Auto reload is **disabled**. Enable auto reload to avoid API interruptions when credits are fully spent.

📄 Pay after-the-fact with monthly invoicing by contacting the Anthropic accounts team.

Anthropic

API da Anthropic (Claude)

No arquivo `anthropic.py`, dentro da função `test` vamos fazer a seguinte configuração:

```
1  import os
2  import anthropic
3
4  anthropic_client = anthropic.Anthropic(api_key=os.getenv("ANTROPIC_API_KEY"))
5
6
7  def teste():
8      response = anthropic_client.messages.create(
9          model="claude-3-5-haiku-latest",
10         max_tokens=1024,
11         system = "Seja direto e responda em uma unica frase.",
12         messages= [
13             {"role": "user", "content": "Qual o sentido da vida?"}
14         ]
15     )
16     return response
```

Anthropic

API da Anthropic (Claude)

Em main.py vamos chamar a função assim:



```
1  #Anthropic:
2  from providers import anthropic
3
4  print(anthropic.teste())
```


Anthropic

API da Anthropic (Claude)


Observe que desse modo no terminal, temos uma resposta completa do modelo, trazendo todas as informações, tanto do texto, quanto de tokens consumidos, data da requisição, ids, etc...

```
(.venv) PS C:\Users\Tiago\Desktop\Python_Aula_LLM> & C:/Users/Tiago/Desktop/Python_Aula_LLM/.venv/Scripts/c
ripts/python.exe c:/Users/Tiago/Desktop/Python_Aula_LLM/main.py
Message(id='msg_014QbMR54CGNoBvqj9goUhdd', content=[TextBlock(citations=None, text='Cada pessoa deve encont
rar seu próprio sentido, construindo significado através de suas relações, propósitos e experiências pesso
is.', type='text')], model='claude-3-5-haiku-20241022', role='assistant', stop_reason='end_turn', stop_sequ
ence=None, type='message', usage=Usage(cache_creation=CacheCreation(ephemeral_1h_input_tokens=0, ephemeral_
5m_input_tokens=0), cache_creation_input_tokens=0, cache_read_input_tokens=0, input_tokens=29, output_token
s=35, server_tool_use=None, service_tier='standard'))
(.venv) PS C:\Users\Tiago\Desktop\Python_Aula_LLM> █
```


Anthropic

API da Anthropic (Claude)

Para filtrar e retornar apenas o texto, modifique o seguinte dentro da função test:



```
1  def teste():
2      response = anthropic_client.messages.create(
3          model="claude-3-5-haiku-latest",
4          max_tokens=1024,
5          system = "Seja direto e responda em uma unica frase.",
6          messages= [
7              {"role": "user", "content": "Qual o sentido da vida?"}
8          ]
9      )
10
11  message = response.content[0].text
12  return message
```



Anthropic

API da Anthropic (Claude)

Dessa forma teremos apenas o texto da mensagem do modelo:

```
• (.venv) PS C:\Users\Tiago\Desktop\Python_Aula_LLM> & C:/Users/Tiago/Desktop/Python_Aula_LLM/.venv/Scripts/python.exe c:/Users/Tiago/Desktop/Python_Aula_LLM/main.py  
O sentido da vida é aquele que cada pessoa constrói individualmente através de suas experiências, relacionamentos, propósitos e busca pessoal de significado.
```

xAI

Primeiros passos com a API da xAI (Grok)

Para aprendermos como consumir uma API da xAI, primeiramente vamos descomentar o campo dentro do **main.py** no nosso sistema:



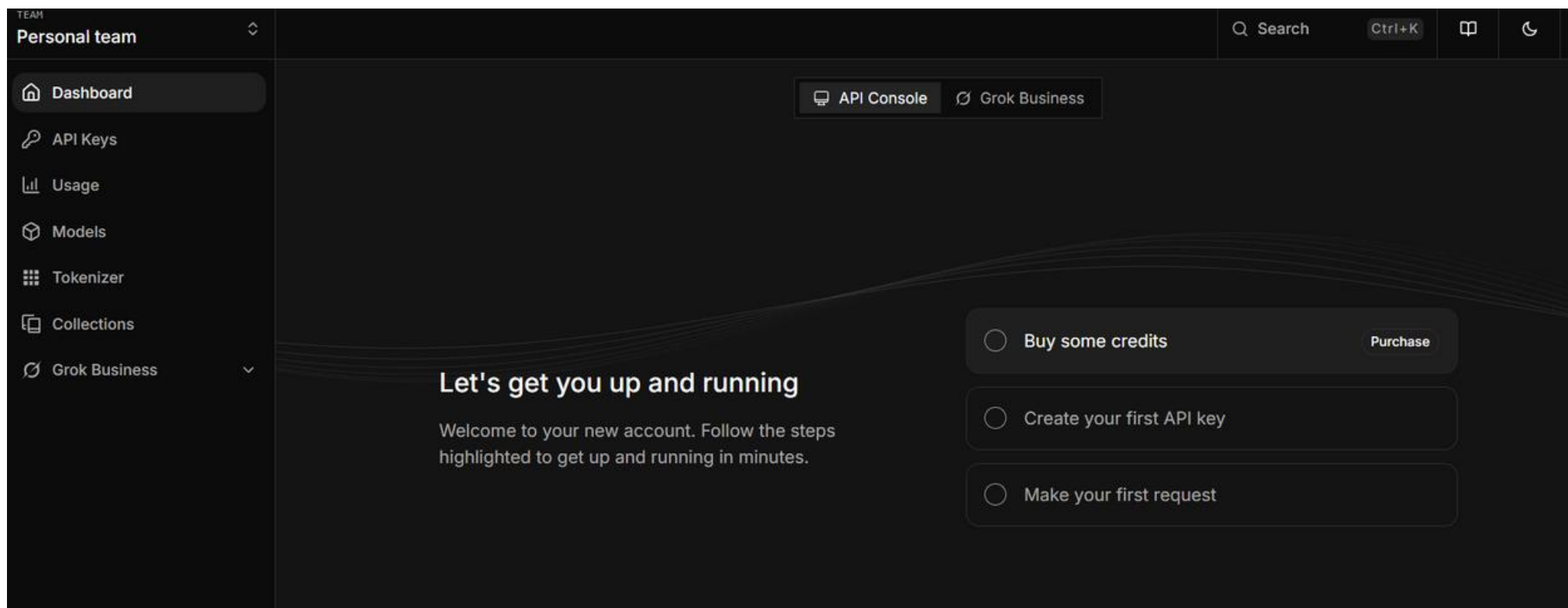
```
1  #Xai:  
2  from providers import xai
```

xAI

Primeiros passos com a API da xAI (Grok)

Agora precisamos buscar uma chave de autorização na xAI no site: **<https://console.x.ai>**


- **Crie ou acesse sua conta**
- **Na tela home você pode adquirir créditos para comprar uma chave.**




xAI

Primeiros passos com a API da xAI (Grok)

Na sequencia copie e cole a chave nas variáveis de ambiente (arquivo .env) que criamos:

 .env

```
1 OPENAI_API_KEY = 'sk-proj-36
2 ANTROPIC_API_KEY = 'sk-ant-a
3  XAI_API_KEY = 'xai-HeLhLGAeM
4 GOOGLE_API_KEY = ''
5 GROQ_API_KEY = ''
```

xAI

Primeiros passos com a API da xAI (Grok)

No arquivo **xai.py** vamos um pouco diferente das outras LLMS, nesta iremos usar uma biblioteca que facilita um pouco as requisições que antes eram puras, agora basta usarmos os parâmetros da biblioteca **xai_sdk** que já importamos, assim:

```
1 import os
2 import xai_sdk
3 from xai_sdk.chat import system, user, assistant
4
5
6 xai_cliente = xai_sdk.Client(api_key=os.getenv("XAI_API_KEY"))
7
8
9 def test():
10
11     chat = xai_cliente.chat.create(model="grok-4-fast-reasoning")
12     chat.append(system("Seja direto e responda em uma frase.))
13     chat.append(user(input("Fale com o XAI: ")))
14
15
16     response = chat.sample().content
17
18     return response
```

xAI

Primeiros passos com a API da xAI (Grok)

Agora podemos brincar um pouco, primeiro vamos mudar as cores na resposta do nosso modelo: (Pesquise por tabelas de cores ANSI Python)



```
1
2 def test():
3
4     chat = xai_cliente.chat.create(model="grok-4-fast-reasoning")
5     chat.append(system("Seja direto e responda em uma frase."))
6     chat.append(user(input("Fale com o XAI: ")))
7
8     print('\033[91mXAI Responde: \033[0m', end='')
9     response = chat.sample().content
10
11     return response
```

A red arrow points to line 8 of the code, highlighting the ANSI color formatting used in the print statement.

Existe também a possibilidade de salvarmos histórico das conversas para que o modelo entenda contextos passados, de diversas maneiras:

- Criando uma Lista para armazenar os eventos, e depois percorrendo a lista em uma estrutura **FOR**, passando para o parâmetro **ASSISTANT**
- Criar o chat fora da função (Armazena de forma temporária)
- Armazenando via Log diretamente na configuração do modelo usado no site.

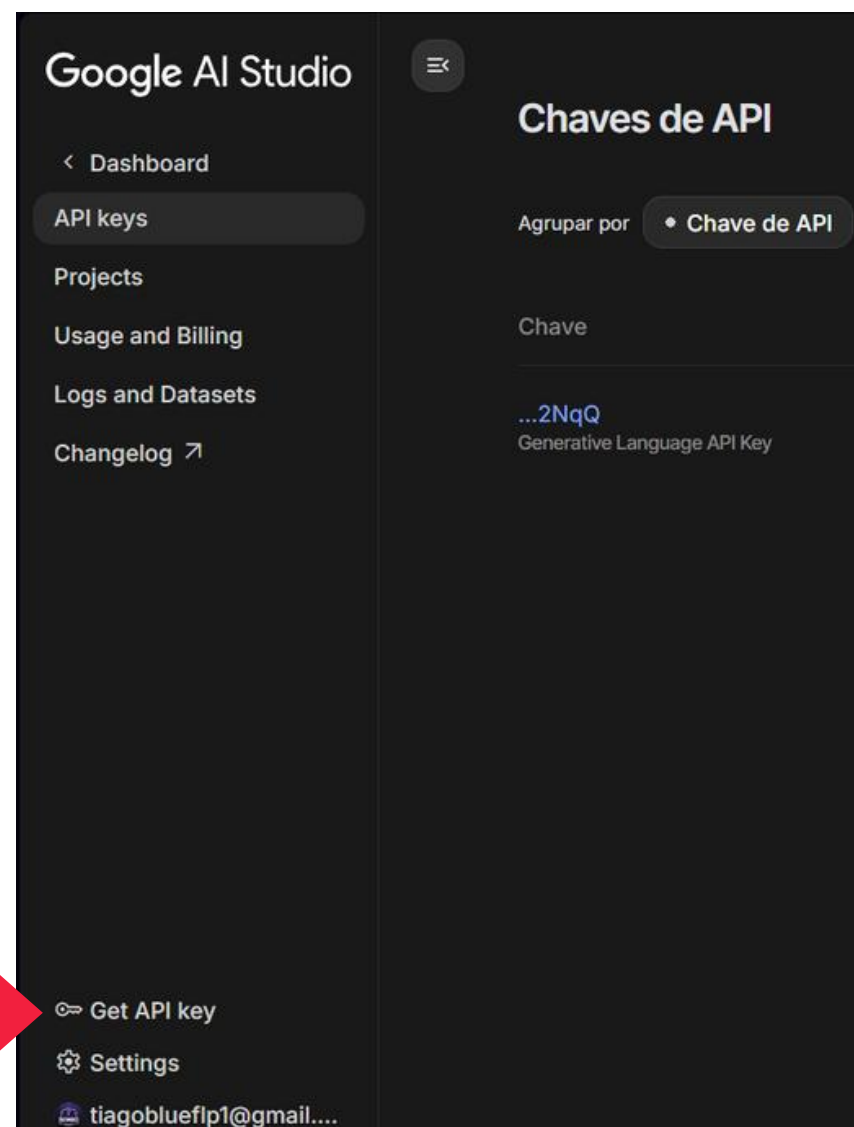
Gemini

Primeiros passos com a API do Google (Gemini)

Para criarmos nosso modelo LLM com o Gemini do google, vá até o site:

<https://aistudio.google.com>

- Entre ou Crie sua conta
- Clique em GET API KEY
- Crie um nome de Projeto
- Crie a Chave (Copie e Cole nas variáveis de ambiente)



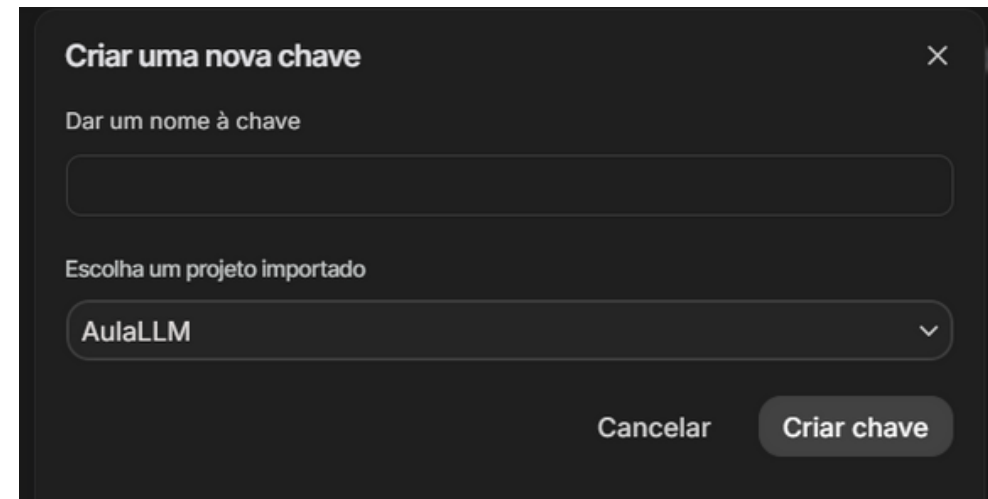
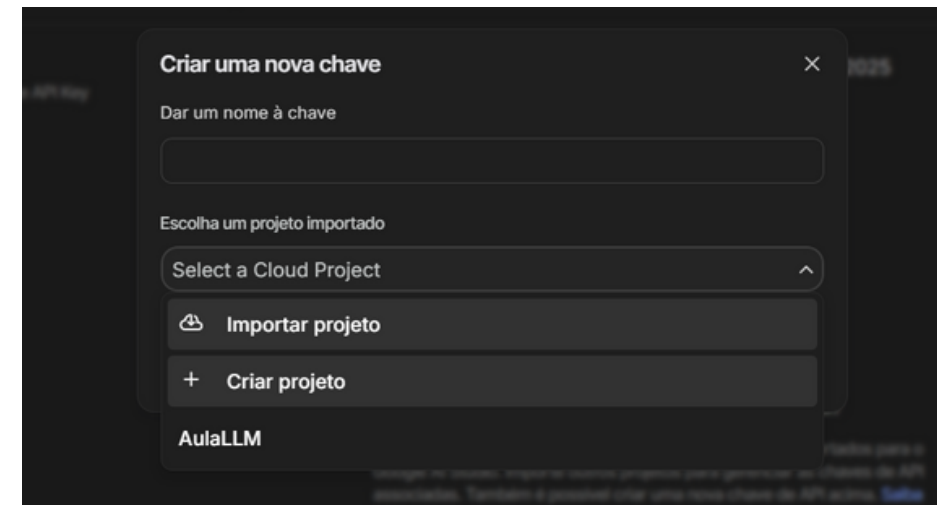
Gemini

Primeiros passos com a API do Google (Gemini)

Para criarmos nosso modelo LLM com o Gemini do google, vá até o site:

<https://aistudio.google.com>


- Entre ou Crie sua conta
- Clique em GET API KEY
- Crie um nome de Projeto
- Crie a Chave (Copie e Cole nas variáveis de ambiente)



Gemini

Primeiros passos com a API do Google (Gemini)

Copie e cole nas variáveis de Ambiente:



```
OPENAI_API_KEY = 'sk-proj  
ANTROPIC_API_KEY = 'sk-ar  
XAI_API_KEY = 'xai-HeLhLQ  
GOOGLE_API_KEY = 'AIzaSyC  
GROQ_API_KEY = ''
```

Gemini

Primeiros passos com a API do Google (Gemini)

Na API do Gemini a configuração é bem parecida com as que vimos até agora, mas com algumas peculiaridades, que chamamos de **gerenate content** vamos configurar assim:

```
1  import os
2  from google import genai
3
4  client = genai.Client(api_key=os.getenv("GOOGLE_API_KEY"))
5
6
7  def test():
8
9      response = client.models.generate_content(
10          model="gemini-2.5-flash",
11          config = genai.types.GenerateContentConfig(
12              system_instruction="Seja direto e responda em uma frase."
13          ),
14
15          contents="Qual o Sentido da vida?"
16
17      )
18
19      return response.text
```

Gemini

Primeiros passos com a API do Google (Gemini)

Observe que a resposta no terminal fica bem pequena, respeitando as instruções passadas dentro de config.

Desafio:

Faça o Teste, comente o trecho de config e rode o código novamente.

Observe que o consumo de tokens aumenta muito.

```
1  def test():
2
3      response = client.models.generate_content(
4          model="gemini-2.5-flash",
5          # config = genai.types.GenerateContentConfig(
6          #     system_instruction="Seja direto e responda em uma frase."
7          # ),
8
9          contents="Qual o Sentido da vida?"
10
11     )
12
13     return response.text
```


Muito Obrigado!

Fim da aula de Hoje



Contatos:

@TiagoBombista 

(21)96696-2564 