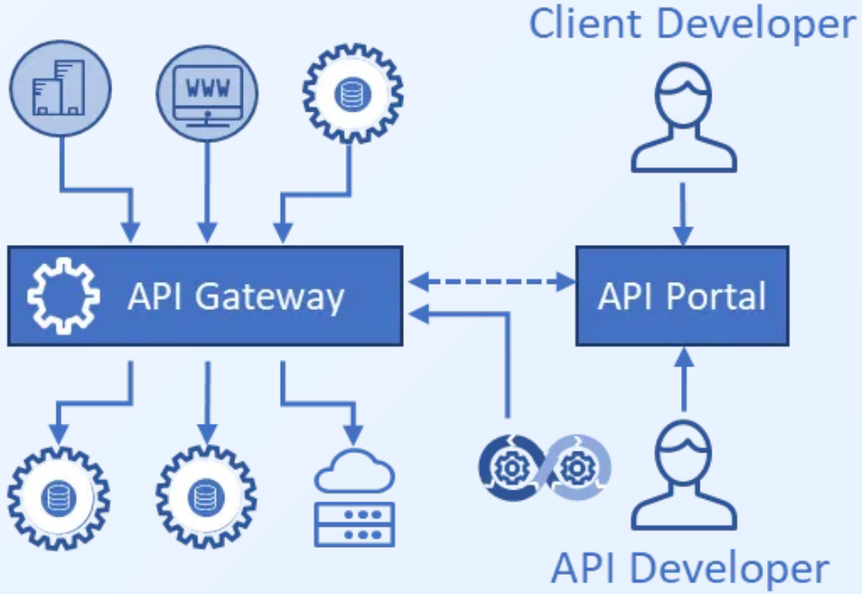


Consumo de APIs e Integração de Dados Externos

Aula 07 - UC2

Ciência de Dados - 2025

Objetivos da Aula

- Compreender o **conceito e a arquitetura** de APIs e sua importância no ecossistema de dados
- Aprender a **consumir APIs públicas** usando Python e a biblioteca requests
- Entender os **mecanismos de autenticação** em APIs e como implementá-los
- Manipular **dados de APIs para análise** com Pandas, transformando JSON em DataFrames
- Realizar **integração prática** com APIs reais em um projeto de ciência de dados

Estas habilidades são fundamentais para qualquer cientista de dados que precise trabalhar com fontes externas de informação.

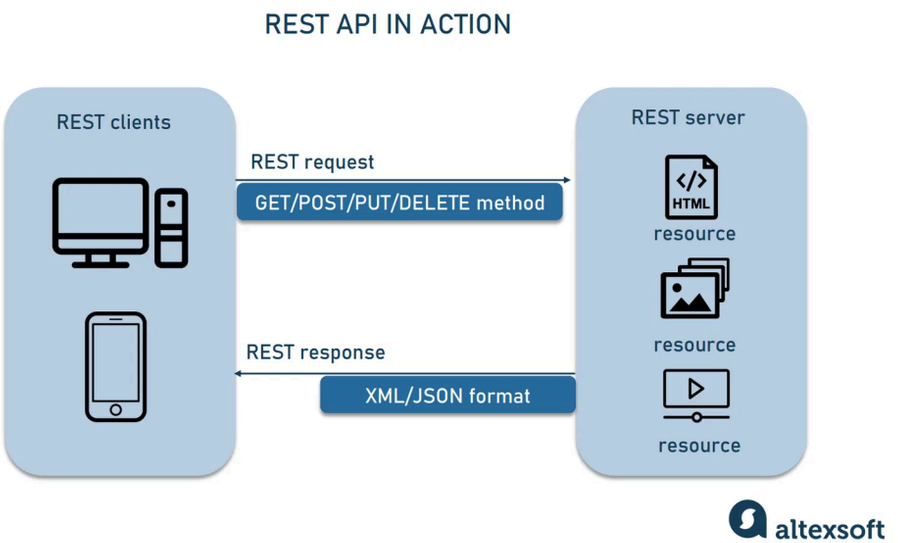
Introdução às APIs - "O Garçom Digital"

"Como um aplicativo de clima consegue a previsão do tempo de qualquer lugar do mundo?"

APIs são como **garçons digitais** que servem dados e funcionalidades entre diferentes sistemas de software. Elas permitem que aplicativos se comuniquem e troquem informações de forma padronizada, sem precisar conhecer os detalhes internos um do outro.

Assim como um garçom leva seu pedido à cozinha e traz sua comida, uma API recebe solicitações de dados, busca essas informações no servidor e as entrega de volta ao solicitante em um formato padronizado.

- **Aplicativos de clima** usam APIs para obter previsões meteorológicas atualizadas
- **Sistemas de pagamento** usam APIs para processar transações de forma segura
- **Aplicativos de mapas** usam APIs para exibir localizações e calcular rotas
- **Redes sociais** oferecem APIs para integrar conteúdo em outros sites



APIs atuam como intermediárias entre aplicações cliente e servidores de recursos

Componentes Chave de APIs

Endpoints

URLs específicas que apontam para recursos da API. Cada endpoint representa uma funcionalidade ou conjunto de dados.

```
https://api.exemplo.com/v1/usuarios
https://api.exemplo.com/v1/produtos?categoria=eletronicos
```

Verbos HTTP

Métodos que definem a ação a ser realizada sobre os recursos:

- **GET**: Buscar/ler dados
- **POST**: Criar novos recursos
- **PUT/PATCH**: Atualizar recursos
- **DELETE**: Remover recursos

Parâmetros de Requisição

Informações adicionais para filtrar ou personalizar a resposta:

- **Query Parameters**: ?cidade=saopaulo&limite=10
- **Path Parameters**: /usuarios/{id}/perfil
- **Headers**: Authorization, Content-Type

Códigos de Status

Indicam o resultado da requisição:

200 OK | 201 Created | 302 Found | 400 Bad Request | 404 Not Found | 500 Server Error

Formatos de Dados

JSON (JavaScript Object Notation) é o formato mais comum para troca de dados em APIs modernas:

```
{
  "id": 123,
  "nome": "Maria Silva",
  "email": "maria@exemplo.com"
}
```

Autenticação em APIs

Métodos de Autenticação

API Key

Chave única que identifica o cliente e autoriza o acesso. Geralmente enviada como parâmetro ou cabeçalho.

```
# Como parâmetro de URL
https://api.exemplo.com/dados?api_key=sua_chave_aqui
```

Autenticação Básica (HTTP Basic)

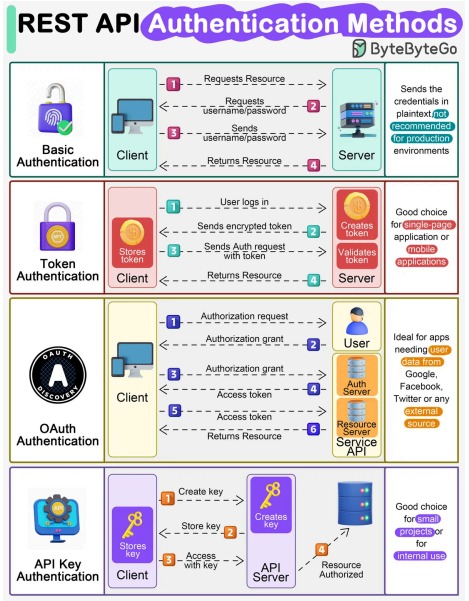
Envia nome de usuário e senha codificados em Base64 no cabeçalho.

```
from requests.auth import HTTPBasicAuth
response = requests.get(url, auth=HTTPBasicAuth('usuario', 'senha'))
```

OAuth 2.0

Protocolo de autorização que usa tokens de acesso com escopo e tempo de vida limitados.

```
headers = {'Authorization': 'Bearer seu_token_aqui'}
response = requests.get(url, headers=headers)
```



Diferentes métodos de autenticação em APIs

Boas Práticas de Segurança

- Nunca armazene chaves de API diretamente no código
- Use variáveis de ambiente ou arquivos de configuração
- Não compartilhe credenciais em repositórios públicos
- Verifique regularmente o uso das suas chaves

Consumo Básico de APIs com Python

Biblioteca Requests

A biblioteca **requests** é a ferramenta padrão em Python para fazer requisições HTTP. Ela simplifica o processo de comunicação com APIs.

```python
# Importando as bibliotecas necessárias
import requests
import json

# Fazendo uma requisição GET simples
url = "https://api.github.com/users/octocat"
response = requests.get(url)

# Verificando o status da resposta
if response.status_code == 200:
    print("Requisição bem-sucedida!")
else:
    print(f"Erro: {response.status_code}")

# Convertendo a resposta para JSON
data = response.json()
print(f"Nome: {data['name']}")
```

Passos para Consumir uma API

1. **Importar as bibliotecas**
   Importe `requests` para fazer requisições HTTP e `json` para manipulação de dados JSON.
2. **Definir a URL do endpoint**
   Especifique o endereço completo do endpoint da API que você deseja acessar.
3. **Enviar a requisição**
   Use `requests.get(url)` para requisições GET ou outros métodos como `post()`, `put()`.
4. **Verificar o status da resposta**
   Verifique `response.status_code` para garantir que a requisição foi bem-sucedida (código 200).
5. **Processar a resposta**
   Use `response.json()` para converter a resposta JSON em um dicionário Python.

```json
# Exemplo de resposta JSON
{
  "name": "The Octocat",
  "followers": 8483,
  "public_repos": 8
}
```

Atividade Prática 1: Exploração de APIs Públicas

Atividade Prática 2: Consumo Simples de API

Objetivo da Atividade

Formato: Exercício prático no Google Colab
API: GitHub Public User Data

1 Importar as Bibliotecas

```
import requests
import json
```

2 Definir a URL do Endpoint

```
username = 'octocat'
url = f'https://api.github.com/users/{username}'
```

3 Enviar a Requisição GET

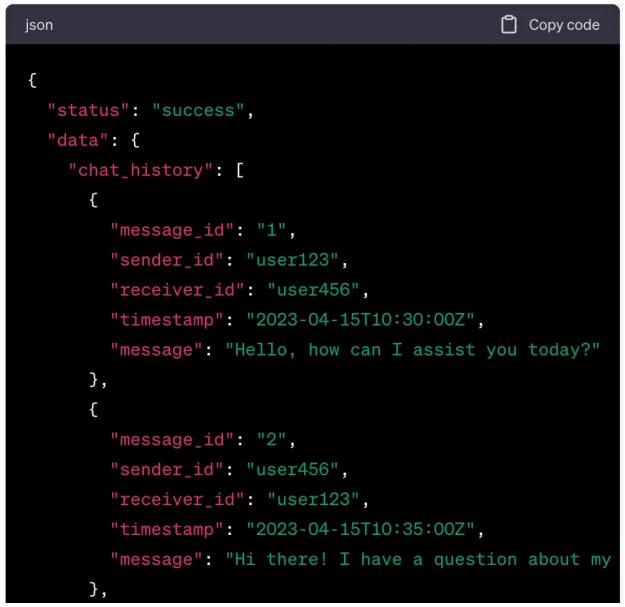
```
response = requests.get(url)
print(f'Status code: {response.status_code}')
```

4 Converter a Resposta para JSON

```
if response.status_code == 200:
  data = response.json()
  print(json.dumps(data, indent=2))
```

5 Extrair Dados Específicos

```
if response.status_code == 200:
  print(f"Nome: {data.get('name')}")
  print(f"Seguidores: {data.get('followers')}")
```



Exemplo de resposta JSON de uma API

Exercício:

Modifique o código para buscar dados de outro usuário
Extraia informações adicionais do perfil
Tente buscar os repositórios do usuário

Atividade Prática 3: Autenticação e Uso de API Keys

Objetivo da Atividade

Formato: Exercício prático no Google Colab | **API:** OpenWeatherMap

1 Obter uma API Key

```
https://home.openweathermap.org/users/sign_up
# Para esta atividade, usaremos uma chave de exemplo
```

2 Armazenar a API Key

```
# Definir a API Key como variável
api_key = "sua_api_key_aqui" # Substitua pela sua chave
```

3 Construir a URL com Parâmetros

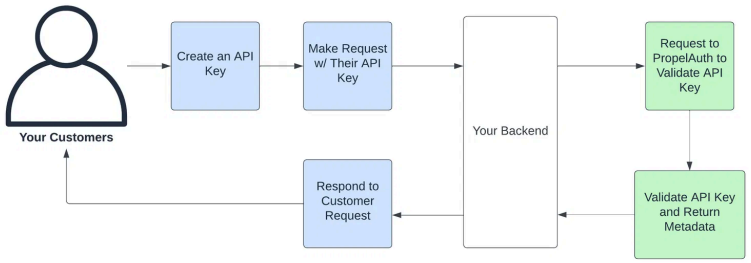
```
import requests

cidade = "São Paulo"
base_url = "https://api.openweathermap.org/data/2.5/weather"

params = {
  "q": cidade,
  "appid": api_key,
  "units": "metric",
  "lang": "pt_br"
}
```

4 Enviar Requisição Autenticada

```
# Enviando a requisição GET com parâmetros
response = requests.get(base_url, params=params)
```



Exemplo de autenticação com API Key em requisições

Exercício:

Modifique o código para consultar o clima de outras cidades
Adicione tratamento de erros para casos como cidade não encontrada
Experimente a previsão de 5 dias usando o endpoint: `api.openweathermap.org/data/2.5/forecast`

Manipulação de Dados de APIs com Pandas

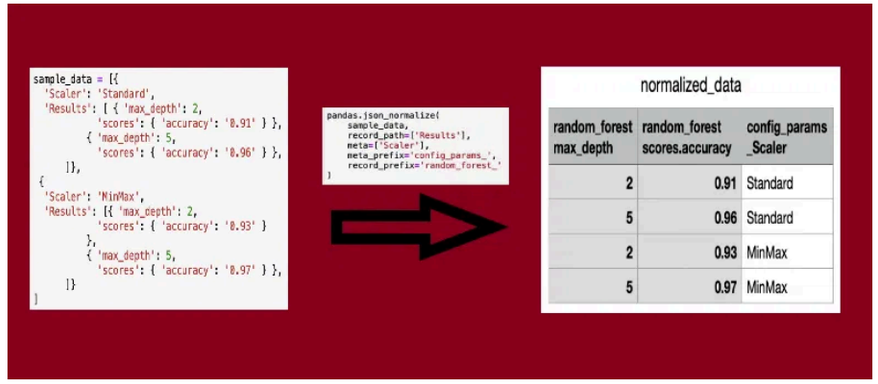
De JSON para DataFrame

O Pandas facilita a transformação de dados JSON obtidos de APIs em DataFrames estruturados, permitindo análises poderosas e manipulações eficientes.

```
# Importando as bibliotecas
import requests
import pandas as pd

# Obtendo dados da API
response =
requests.get('https://api.github.com/users/octocat/repos')
repos = response.json()

# Transformando em DataFrame
df = pd.DataFrame(repos)
df_clean = df[['name', 'language', 'stargazers_count', 'forks']]
```



Processo de transformação de dados JSON de API para DataFrame do Pandas

Manipulações Comuns

- **Seleção de colunas:** Escolha apenas os dados relevantes usando `df[['coluna1', 'coluna2']]`
- **Filtragem de dados:** Use condições lógicas como `df[df['stars'] > 100]`
- **Ordenação:** Organize os dados com `df.sort_values('coluna', ascending=False)`

Dica: Lidando com Estruturas Aninhadas

Dados de APIs frequentemente vêm em estruturas JSON aninhadas. Use estas técnicas:

- Use `json_normalize()` para achatar estruturas aninhadas
- Acesse dados aninhados com `df['coluna'].apply(lambda x: x.get('subchave'))`

Atividade Prática 4: Extração e Manipulação para Análise

Objetivo da Atividade

Extrair dados de uma API e transformá-los em DataFrame para análise.
Formato: Exercício no Google Colab

1 Obter Dados da API

```
import requests
import pandas as pd

username = "pandas-dev"
url = f"https://api.github.com/users/{username}/repos"
response = requests.get(url)
repos_data = response.json()
```

2 Transformar em DataFrame

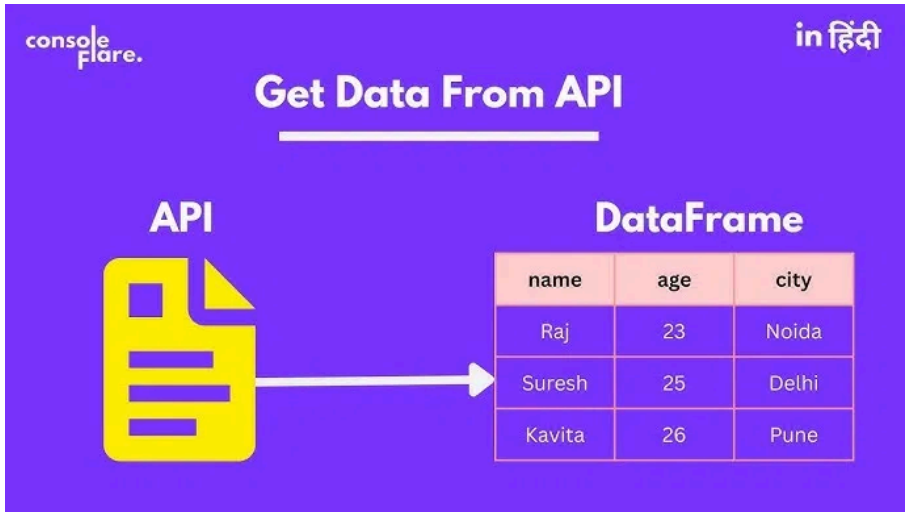
```
repos_list = []
for repo in repos_data:
  repos_list.append({
    "nome": repo["name"],
    "estrelas": repo["stargazers_count"],
    "linguagem": repo["language"]
  })

df = pd.DataFrame(repos_list)
```

3 Limpar e Preparar os Dados

```
# Preencher valores nulos na coluna linguagem
df["linguagem"] = df["linguagem"].fillna("Não especificada")

# Ordenar por número de estrelas
df = df.sort_values("estrelas", ascending=False)
```



Fluxo de transformação: API → JSON → DataFrame → Análise

Desafio:

Modifique o código para buscar repositórios de outra organização
Adicione mais colunas ao DataFrame com informações relevantes
Crie uma visualização gráfica das linguagens mais usadas

Encerramento e Próximos Passos

O Que Aprendemos Hoje

- Conceitos fundamentais de APIs e sua arquitetura REST
- Como consumir APIs públicas usando a biblioteca requests
- Métodos de autenticação em APIs e como implementá-los
- Transformação de dados JSON em DataFrames do Pandas

→ Próximos Passos

1 Explorar APIs Mais Complexas

Experimentar APIs com OAuth 2.0 e dados mais complexos.

2 Integração com Banco de Dados

Armazenar dados de APIs em bancos para análises históricas.

3 Automação de Coleta de Dados

Implementar scripts para coleta periódica de dados.

Recursos Adicionais

- **Documentação do Requests:** https://docs.python-requests.org/
- **Documentação do Pandas:** https://pandas.pydata.org/docs/
- **Diretório de APIs Públicas:** https://github.com/public-apis/public-apis

Conexão com a Próxima Aula

Na próxima aula, vamos aprofundar nossos conhecimentos em visualização de dados com Matplotlib e Seaborn, utilizando os dados que aprendemos a coletar via APIs.

> *"APIs são as pontes invisíveis que conectam o vasto ecossistema digital. Dominar seu uso é essencial para qualquer cientista de dados que busca trabalhar com informações do mundo real."*
>
> ***— Daniel Fireman, Cientista de Dados***