

Machine Learning Engineer

Python para ML e IA

Por que a pessoa engenheira de machine learning precisa saber APIs?

Leonardo Pena

Objetivo dessa segunda parte



Passo 1

Explicar como
as APIs são
estruturadas



Passo 2

Diferenciar
REST, SOAP,
GraphQL,
gRPC,
WebSockets



Passo
3

Mostrar
exemplos de
endpoints e
requisições



Passo 4

Ilustrar
quando cada
arquitetura é
mais
adequada



Passo
5

Preparar o
caminho
para
integração
prática



Revisão





/ O Papel de APIs no ML

- Interoperabilidade entre sistemas
- Facilitar acesso a modelos em produção
- Isolamento de código interno
- Flexibilidade para evoluir o modelo
- Manutenção sem interromper o serviço



AAPI REST



/ Conceitos principais



- REST = Representational State Transfer
- Uso do protocolo HTTP e métodos (GET, POST, PUT, DELETE)
- Uniformidade de interface e recursos representados por URLs
- Stateless: cada requisição é independente
- Cacheável: melhora desempenho e escalabilidade

/ Conceitos principais



- CRUD: Create (POST), Read (GET), Update (PUT), Delete (DELETE)
- Exemplo de endpoint: /predict para ML
- JSON como formato de troca de dados mais comum
- Facilmente compreendida por diversas linguagens
- Amplo suporte em frameworks e bibliotecas

A decorative graphic consisting of numerous thin, white, wavy lines that create a sense of depth and movement, resembling a stylized landscape or a series of concentric ripples. These lines are concentrated in the bottom left corner of the slide and fade out towards the right.

/ RESTful API

RESTful API (Representational State Transfer API) é um tipo de API que segue os princípios REST, utilizando métodos HTTP padrão (GET, POST, PUT, DELETE)

/ Exemplo de Endpoint REST: /predict

- URL: <https://api.meusistema.com/predict>
- Método: POST com payload de dados
- Resposta: JSON com previsões (ex.:
{"previsão":0.92})
- Separação clara entre cliente e servidor
- Permite versionamento: /v1/predict,
/v2/predict



Outros tipos de API



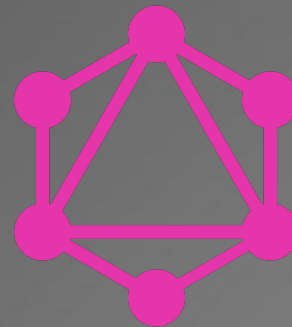


/ SOAP

- SOAP = Simple Object Access Protocol
- Uso mais comum em ambientes corporativos antigos
- Baseado em XML e WSDL (Web Services Description Language)
- Estrutura mais “formal” e verbosa
- Menos comum no desenvolvimento web moderno

/ GraphQL

- Criado pelo Facebook (Meta)
- Cliente define exatamente quais dados quer receber
- Evita over-fetching ou under-fetching
- Ideal para cenários com múltiplas fontes de dados
- Maior flexibilidade em comparação ao REST





/ gRPC

- Usa HTTP/2 e Protobuf (formato binário)
- Comunicação rápida e eficiente
- Ideal para microsserviços e baixa latência
- Bom para sistemas distribuídos de alta performance
- Requer suporte específico no servidor e no cliente

/ WebSockets

- Conexão bidirecional e em tempo real
- Envio e recebimento de dados sem novas requisições HTTP
- Útil para dashboards ao vivo e notificações imediatas
- Não segue o padrão REST
- Demanda abordagem de programação reativa




/ Comparativo

- **REST**: fácil de implementar, amplamente adotada
- **SOAP**: mais complexo, focado em legados
- **GraphQL**: flexibilidade no consumo de dados
- **gRPC**: alto desempenho, baixa latência
- WebSockets: comunicação em tempo real

/ Comparativo

- **REST** é o padrão de fato para APIs públicas
- **SOAP** aparece em integrações corporativas mais antigas
- **GraphQL** ganha espaço em apps complexos de frontend
- **gRPC** popular em infra com microsserviços
- **WebSockets** necessário para eventos em tempo real



/ Foco do curso: API
REST



/ Dicas de Estrutura de Endpoints

- Usar verbos HTTP corretamente (POST, GET)
- Nomear endpoints de forma intuitiva (/predict, /train)
- Evitar URLs muito longas ou complexas
- Retornar códigos de status HTTP adequados (200, 400, 500)
- Manter consistência entre endpoints e parâmetros

/ Conclusão

- Conhecemos as principais arquiteturas de API
- Entendemos vantagens e limitações de cada abordagem
- Vimos exemplos de aplicações e integrações
- Sabemos quando escolher REST, GraphQL, gRPC ou WebSockets

POSTECH

FIAP + alura