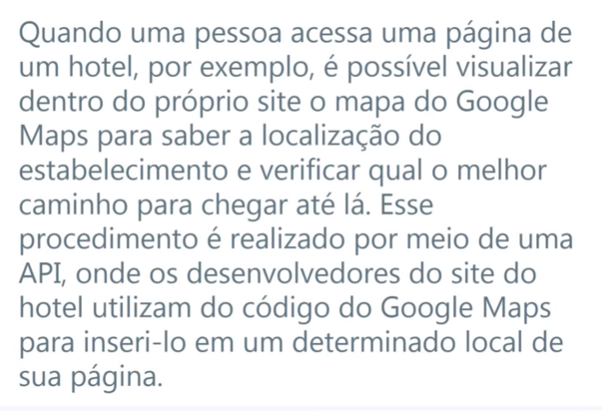
## Resumo

<https://www.youtube.com/watch?v=RVIJnOHKOLc>

interface de programação de aplicação (api) é o conjunto de padrões de programação que permite a construção de aplicativos e a sua utilização de maneira não tão evidente para o usuário.

ná pratica é um script que é introduzido em um programa e retorna um resultado

ex



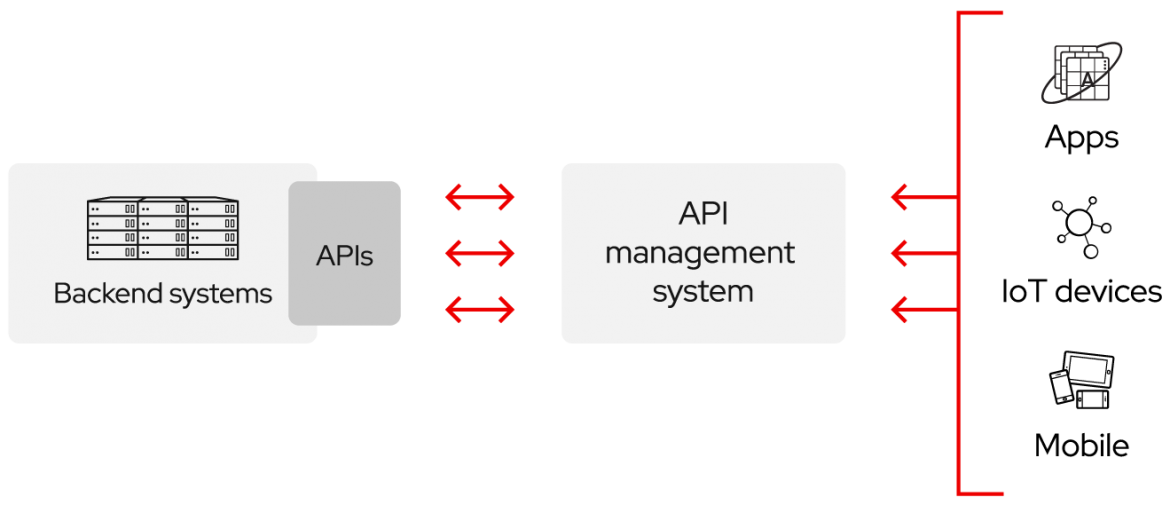
API é um conjunto de definições e protocolos usado no desenvolvimento e na [integração](https://www.redhat.com/pt-br/topics/integration) de software de aplicações. API é um acrônimo em inglês que significa interface de programação de aplicações.

Uma API permite que sua solução ou serviço se comunique com outros produtos e serviços sem precisar saber como eles foram implementados. Isso simplifica o desenvolvimento de aplicações, gerando economia de tempo e dinheiro. Ao desenvolver novas ferramentas e soluções (ou ao gerenciar aquelas já existentes), as APIs oferecem a flexibilidade necessária para simplificar o design, a administração e o uso, além de fornecer oportunidades de inovação.

As APIs costumam ser vistas como contratos, com documentações que representam um acordo entre as partes interessadas. Se uma dessas partes enviar uma solicitação remota estruturada de uma forma específica, isso determinará como o software da outra parte responderá.

As APIs simplificam a forma como os desenvolvedores integram novos componentes de aplicações a uma arquitetura preexistente. Por isso, elas ajudam na colaboração entre as empresas e as equipes de TI. Muitas vezes, as necessidades empresariais mudam rapidamente para responder aos mercados digitais em transformação. Nesse ambiente, novos concorrentes podem redefinir o setor inteiro com uma nova aplicação. Para manter a competitividade, é importante oferecer suporte à implantação e desenvolvimento rápidos de serviços inovadores. O desenvolvimento de [aplicações nativas em cloud](https://www.redhat.com/pt-br/topics/cloud-native-apps) é uma forma conhecida de aumentar a velocidade de criação. Ele depende de uma arquitetura de aplicações de [microsserviços](https://www.redhat.com/pt-br/topics/microservices) conectada por meio de uma API.

As APIs são uma maneira simplificada de conectar a própria infraestrutura por meio do desenvolvimento de aplicações nativas em nuvem. No entanto, elas também possibilitam o compartilhamento de dados com clientes e outros usuários externos. As APIs públicas agregam valor comercial porque simplificam e ampliam a forma como você se conecta aos parceiros, além de possivelmente monetizar seus dados. Um exemplo famoso é a API do [Google Maps](https://www.google.com/maps).



Por exemplo, imagine uma empresa distribuidora de livros. Essa distribuidora de livros *poderia* oferecer aos clientes uma aplicação em que os atendentes de uma livraria pudessem verificar a disponibilidade de um título diretamente com a distribuidora. Essa aplicação poderia ser cara de desenvolver, limitada pela plataforma e exigiria longos períodos de desenvolvimento e manutenção contínua.

Como alternativa, a distribuidora, poderia fornecer uma API para verificar a disponibilidade no estoque. Essa abordagem proporciona vários benefícios, incluindo:

* O acesso aos dados por meio de uma API ajuda os clientes a consolidarem informações sobre seu inventário em um único local.
* A distribuidora de livros pode fazer alterações nos sistemas internos sem causar impacto nos clientes, contanto que o comportamento da API não mude.
* Com uma API disponibilizada publicamente, os desenvolvedores que trabalham para a distribuidora de livros, os vendedores ou terceiros poderiam desenvolver uma aplicação para ajudar os clientes a encontrar os livros que procuram. Isso poderia resultar no aumento das vendas ou outras oportunidades de negócios.

Para resumir, com as APIs, você libera o acesso aos seus recursos sem abrir mão da [segurança e do controle](https://www.redhat.com/pt-br/topics/security). É você quem determina como isso será feito e quem terá acesso. [A segurança das APIs](https://www.redhat.com/pt-br/topics/security/api-security) dependem de um bom gerenciamento. É possível conectar APIs, bem como criar aplicações que fazem uso dos dados ou funcionalidades disponibilizadas por elas, usando uma plataforma de integração distribuída que ligue todos os elementos, incluindo sistemas legados e dispositivos de Internet das Coisas (IoT).

Existem três abordagens para políticas de lançamento de APIs.

#### **API Privada**

A API é usada apenas internamente. Isso oferece às empresas um maior controle.

#### **API de Parceiros**

A API é compartilhada com parceiros de negócios específicos. Isso pode fornecer fluxos de receita adicionais sem comprometer a qualidade.

#### **API Pública**

A API é disponibilizada para todos. Terceiros podem desenvolver aplicações que interajam com a sua API e isso pode se tornar uma fonte de inovação.

### **Inovação com API**

A exposição das suas APIs aos parceiros ou ao público pode:

* Criar novos canais de receita ou ampliar os existentes.
* Expandir o alcance da sua marca.
* Facilitar a inovação aberta ou aumentar a eficiência por meio da colaboração e de desenvolvimento externos.

Parece bom, não é? Mas como as APIs podem fazer tudo isso?

Vamos voltar para o exemplo da empresa distribuidora livros.

Suponhamos que um dos parceiros da empresa desenvolva uma aplicação para ajudar as pessoas a encontrar livros nas prateleiras de livrarias. Essa experiência aprimorada atrai mais consumidores para livraria (que é cliente da distribuidora) e amplia o canal de receita existente.

Talvez um terceiro use uma API pública para desenvolver uma aplicação que permita que as pessoas comprem livros diretamente da distribuidora em vez de em uma livraria. Isso abre um novo canal de receita para a distribuidora de livros.

O compartilhamento de APIs (com parceiros selecionados ou aberta a todos) pode ter efeitos positivos. Cada parceria amplia o reconhecimento da sua marca para além dos esforços de marketing da sua empresa. Abrir a tecnologia para todos, como fazem as APIs públicas, estimula os desenvolvedores a criar um ecossistema de aplicações com base na sua API. Quanto mais pessoas usam sua tecnologia, maior é a sua chance de gerar novos negócios.

Tornar a tecnologia pública pode gerar resultados novos e inesperados. Às vezes, esses resultados revolucionam setores inteiros. No caso da nossa empresa distribuidora de livros, novas organizações (um serviço de empréstimos de livros, por exemplo) podem mudar fundamentalmente a maneira de fazer negócios. Com as APIs públicas e em parceria, você pode se beneficiar dos esforços criativos de uma comunidade muito maior do que a sua equipe interna de desenvolvedores. Novas ideias podem surgir de qualquer lugar. Por disso, as empresas precisam estar cientes das mudanças no mercado e prontas para agir. As APIs podem ajudar.

### **Um breve histórico das APIs**

As APIs surgiram nos primeiros dias da computação, muito antes do computador pessoal. Naquela época, elas eram normalmente usadas como bibliotecas para sistemas operacionais. Embora a API enviasse mensagens entre mainframes em alguns momentos, ela era quase sempre local para os sistemas em que operava. Depois de quase 30 anos, as APIs se expandiram para além dos ambientes locais. No início dos anos 2000, elas estavam se tornando uma tecnologia importante para a integração remota de dados.

## **APIs remotas**

As APIs remotas foram projetadas para interagir por meio de uma rede de comunicações. Quando falamos “remota”, queremos dizer que os recursos manipulados pela API estão em algum lugar fora do computador que faz a solicitação. Como a rede de comunicações mais usada é a Internet, a maioria das APIs são projetadas com base em padrões da web. Nem todas as APIs remotas são web, mas é justo afirmar que, em geral, as APIs web são remotas.

As APIs web normalmente usam o protocolo HTTP para mensagens de solicitação e fornecem uma definição da estrutura das mensagens de resposta. Essas mensagens de resposta geralmente têm o formato de arquivo XML ou JSON. Tanto XML quanto JSON são formatos de preferência porque apresentam os dados de forma simplificada, o que facilita a manipulação por outras aplicações.

## **Quais foram as melhorias feitas nas APIs?**

Conforme as APIs se transformavam em APIs web, vários esforços foram feitos para facilitar um pouco a sua criação e torná-las mais úteis.

### **Um pouco de SOAP e um monte de REST**

Com a proliferação das APIs web, uma especificação de protocolo foi desenvolvida para ajudar a padronizar a troca de informações: o Simple Object Access Protocol, mais conhecido como SOAP. As APIs projetadas com SOAP usam o XML como formato de mensagem e recebem solicitações por HTTP ou SMTP. O SOAP facilita o compartilhamento de informações por aplicações executadas em ambientes diferentes ou escritos em linguagens diferentes.

Outra especificação é a Representational State Transfer (REST). APIs web que adotam as restrições de arquitetura da REST são chamadas de APIs RESTful. A REST é fundamentalmente diferente do SOAP: o SOAP é um protocolo e a REST é um estilo de arquitetura. Isso significa que não há um padrão oficial para APIs RESTful web. Conforme definido na dissertação de Roy Fielding [“Architectural Styles and the Design of Network-based Software Architectures”](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm), as APIs serão consideradas RESTful se estiverem em conformidade com seis restrições de arquitetura:

* **Arquitetura cliente-servidor:** a arquitetura REST é composta por clientes, servidores e recursos. Ela lida com as solicitações via HTTP.
* **Sem monitoração de estado:** nenhum conteúdo do cliente é armazenado no servidor entre as solicitações. Em vez disso, as informações sobre o estado da sessão são mantidas com o cliente.
* **Capacidade de cache:** o armazenamento em cache pode eliminar a necessidade de algumas interações entre o cliente e o servidor.
* **Sistema em camadas:** as interações entre cliente e servidor podem ser mediadas por camadas adicionais. Essas camadas podem oferecer recursos extras, como balanceamento de carga, caches compartilhados ou segurança.
* **Código sob demanda (opcional):** os servidores podem ampliar a funcionalidade de um cliente por meio da transferência de códigos executáveis.
* **Interface uniforme:** essa restrição é essencial para o design de APIs RESTful e inclui quatro vertentes:
  + **Identificação de recursos nas solicitações:** os recursos são identificados nas solicitações e separados das representações retornadas para o cliente.
  + **Manipulação de recursos por meio de representações:** os clientes recebem arquivos que representam recursos. Essas representações precisam ter informações suficientes para permitir a modificação ou exclusão.
  + **Mensagens autodescritivas:** cada mensagem retornada para um cliente contém informações suficientes para descrever como ele deve processá-las.
  + **Hipermídia como plataforma do estado das aplicações:** depois de acessar um recurso, o cliente REST pode descobrir todas as outras ações disponíveis no momento por meio de hiperlinks.

Essas restrições podem parecer excessivas, mas são muito mais simples do que um protocolo prescrito. Por isso, as APIs RESTful estão se tornando mais comuns do que as APIs SOAP.

Nos últimos anos, as especificações da [OpenAPI](https://www.openapis.org/) se tornaram o padrão na hora de definir APIs REST. A OpenAPI permite que desenvolvedores de todas as linguagens criem interfaces de API REST compreensíveis com o mínimo de suposições.

### **SOA ou arquitetura de microsserviços**

As duas abordagens de arquitetura que mais usam APIs remotas são a arquitetura orientada a serviços (SOA) e a arquitetura de [arquitetura de microsserviços](https://www.redhat.com/pt-br/topics/microservices). A SOA, a mais antiga das duas abordagens, começou como um aprimoramento sobre as aplicações monolíticas. Considerando que um único aplicativo monolítico faz tudo, algumas funções podem ser fornecidas por aplicações diferentes levemente acopladas por meio de um padrão de integração, como um barramento de serviços corporativos (ESB).

Embora a SOA seja, de forma geral, mais simples que uma arquitetura monolítica, ela traz o risco de mudanças em cascata por todo o ambiente caso as interações entre os componentes não sejam compreendidas claramente. Essa complexidade adicional traz de volta alguns dos problemas que a SOA foi criada para solucionar.

As arquiteturas de microsserviços são parecidas com os padrões SOA no que diz respeito à utilização de serviços levemente acoplados. No entanto, elas vão além no rompimento com as arquiteturas tradicionais. Na arquitetura de microsserviços, os serviços usam um framework de mensageria comum, como as APIs RESTful. As APIs RESTful são usadas na comunicação entre serviços sem transações de conversão de dados complicadas ou camadas de integração adicionais. O uso de APIs RESTful permite – e até estimula – a entrega mais rápida de novos recursos e atualizações. Cada serviço é independente. Um serviço pode ser substituído, aprimorado ou descartado sem afetar nenhum outro na arquitetura. Com essa arquitetura leve, é possível otimizar os serviços distribuídos ou de cloud e oferecer suporte à escalabilidade dinâmica para serviços individuais.