# **API E INTEGRAÇÕES**

API é a sigla em inglês para Application Programming Interface, ou interface de programação de aplicações. As interfaces de programação de aplicativos (APIs) são conjuntos de ferramentas, definições e protocolos para a criação de aplicações de software.

Resumidamente, API nada mais é do que uma forma de comunicação entre sistemas em que um deles fornece informações e serviços que podem ser utilizados pelo outro, sem a necessidade de algum dos sistemas conhecer detalhes de implementação do software.

# **TIPOS DE API**

Existem diversos tipos de APIs, cada uma com diferentes níveis de acesso e controle. A API privada é usada internamente pelas empresas, oferecendo um controle direto sobre seu uso. Em contraste, a API pública está disponível para todos, permitindo que terceiros desenvolvam novas aplicações e impulsionem a inovação. Além disso, há APIs compartilhadas com parceiros de negócios específicos, que podem gerar receitas adicionais sem comprometer a qualidade do serviço. As APIs remotas ou web são desenvolvidas para interagir através de redes de comunicação, acessando recursos fora do dispositivo solicitante. Embora nem todas as APIs remotas sejam necessariamente da web, as APIs web são comumente remotas. Elas normalmente utilizam o protocolo HTTP para enviar solicitações e definem a estrutura das respostas, que geralmente são formatadas em XML ou JSON.

# **API SOAP E API REST**

Com o aumento do uso de APIs na web, surgiram normas de protocolo para padronizar a troca de informações. Duas delas são o Simple Object Access Protocol (SOAP) e a Representational State Transfer (REST). As APIs que seguem o padrão SOAP utilizam XML para mensagens e recebem solicitações via HTTP ou SMTP, facilitando a comunicação entre diferentes aplicações. Por outro lado, as APIs RESTful, seguindo os princípios da arquitetura REST, não têm um padrão oficial definido, mas são consideradas RESTful se atenderem às seis restrições de arquitetura estabelecidas por Roy Fielding.

* **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.

# **FERRAMENTAS**

# **SWAGGER**

Ele é de código aberto e possibilita aos desenvolvedores e equipes criar, documentar e utilizar APIs em RESTful.

O Swagger é capaz de ajudar desenvolvedores de APIs REST a:

* Modelar APIs;
* Gerar documentação automatizada da API;
* Gerar códigos do cliente e do servidor, suportando diversas linguagens de programação;
* Testar funcionalidades da API.

Para executar essas atividades, o Swagger requer a definição da OpenAPI, que é uma linguagem usada para descrever os contratos das APIs REST. A OpenAPI estabelece um formato JSON com campos consistentes, permitindo que o usuário descreva os recursos, os modelos de dados, as URIs, os tipos de conteúdo e os métodos HTTP aceitos.

Além da OpenAPI, o Swagger dispõe um conjunto de ferramentas e as principais são:

* Swagger Editor
* Swagger UI
* Swagger Codegen

O módulo de UI permite que desenvolvedores interajam intuitivamente com as APIs em sandbox. Enquanto isso, o Editor é uma ferramenta online que permite uma documentação mais interativa, utilizando YAML. Uma vantagem do Editor é sua coleção de modelos de documentos que oferecem uma base para aqueles que não querem começar do zero na documentação.

A documentação para APIs pode ser feita de três formas:

1. **Manualmente:** o desenvolvedor escreve livremente as especificações da API e as publica posteriormente em seu próprio servidor.
2. **Automaticamente**: simultaneamente ao desenvolvimento da API é gerada a documentação.
3. **Codegen**: converte todas as anotações contidas no código fonte das APIs REST em documentação.

# **API BLUEPRINT**

É uma linguagem de descrição e especificação de APIs que utiliza Markdown, simplificando a edição para usuários não familiarizados com códigos. A maioria de suas ferramentas são de código aberto, visando agilizar o processo de criação e prototipagem de APIs, assim como documentar e testar APIs já em uso de forma eficiente.

A documentação nesse caso mostra:

* Atributos listados com suas descrições
* Exemplos de solicitações
* Respostas desses exemplos

No caso do Blueprint, duas são as ferramentas integradas a ele que se destacam:

1. **Dredd**

A ferramenta possibilita a realização de testes no serviço de back-end com base na documentação da API, o que ajuda a resolver problemas de desatualização dessa documentação. Além disso, ela oferece suporte para uma variedade de linguagens, incluindo PHP, Python, Ruby, Perl, Node.js e Go.

1. **Drakov**

Facilita o início e teste de serviços simulados, onde as solicitações e respostas são ajustadas de acordo com a documentação da API, funcionando como um ambiente de testes.

# **RAML**

O RAML oferece opções de documentação de APIs com ferramentas open source, como o Console da API e o HTML RAML 2. A rápida geração de documentação é facilitada por analisadores disponíveis em vários idiomas, permitindo a criação de documentos personalizados e scripts interativos. O formato base do RAML é YAML, mas também é compatível com outros padrões como JSON e é neutro em relação a ferramentas de linguagem, como Java, Javascript, .Net, PHP, Python e Ruby. No entanto, o uso do RAML pode ter desvantagens, especialmente se a API não seguir os princípios RESTful ou se não for baseada em HTTP. Além disso, o YAML é voltado para desenvolvedores, o que pode dificultar para equipes sem conhecimentos técnicos criarem APIs ou documentações sem o uso de ferramentas de design adequadas.

**REFERÊNCIAS**

O que é API?. **Red Hat**,2023. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces>. Acesso em: 28 fev. 2024.

RUIZ, David. Conheça as 3 principais ferramentas de documentação de APIs. **API Playbook**, 2020. Disponível em: <https://www.apiplaybook.com/p/conheca-as-3-principais-ferramentas-de-documentacao-de-apis>. Acesso em: 28 fev. 2024.