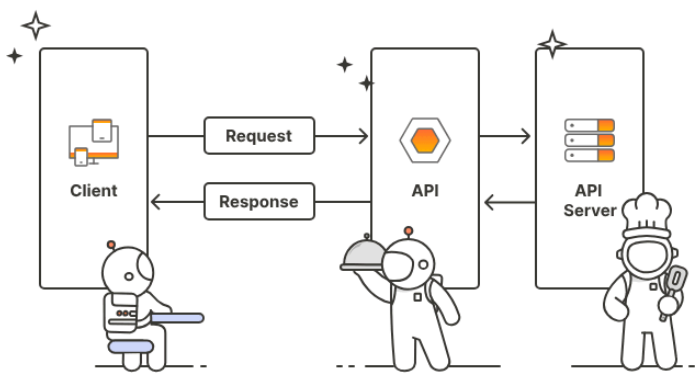
APIs

Link Notion: https://cherry-client-b8f.notion.site/APIs-34aeee9416f04ddab37c459078d6477f

O que é uma API?

Application Programming Interface são uma forma pela qual as aplicações conseguem se comunicar e utilizar recursos umas das outras. Ela atua como um intermediário, recebendo as requisições do cliente, enviando ao servidor web e retornando com a resposta.



O fluxo básico é:

1. **Requisição (request)**: você pede algo à API.
2. **Servidor processa:** ele recebe o pedido, processa e prepara os dados.
3. **Resposta (response):** a API devolve o que você pediu.

Principais conceitos

Para entender como utilizar APIs, é importante compreender alguns conceitos antes.

Como funciona a requisição?

APIs usam HTTP, que é o mesmo protocolo da web. Os métodos HTTP são como verbos que indicam o que você quer fazer.

Método	O que faz	Exemplo prático
GET	Buscar informações	Pegar a lista de produtos
POST	Criar algo novo	Criar um novo usuário
PUT	Atualizar totalmente	Atualizar todos os dados de um usuário
PATCH	Atualizar parcialmente	Alterar só o e-mail de um usuário
DELETE	Apagar recurso	Remover um produto do sistema

Como eu acesso uma API?

Para acessar uma API, nós utilizamos um **endpoint**, que é o endereço da API que você acessa para obter ou enviar informações.

Exemplos:

- https://api.loja.com/produtos → lista de todos os produtos
- https://api.loja.com/produtos/123 → dados do produto de ID 123

- https://api.loja.com/usuarios/5/pedidos → pedidos do usuário com ID 5

Cada endpoint normalmente possui:

- **URL base** (pode conter além do domínio, a versão, como /v1, /v2)
- **Caminho** que você quer acessar, como */produtos* ou */usuarios/idUsuario*
- **Parâmetros de consulta**, onde são passadas informações extras para filtrar ou detalhar o que quer receber. Começam com **?** e separam múltiplos parâmetros com **&**.

Para chamar essas APIs, é possível utilizar alguns métodos:

Navegador

Para acessar pelo navegador, é só colocar o link de acesso à API na barra de endereço e o navegador deverá mostrar o JSON de resposta ou a mensagem de erro caso a requisição não possa ser completada. Caso a API seja protegida, é necessário o uso de um token de autenticação.

Outras opções

Existem softwares especializados na manipulação de APIs, sendo os principais: **PostMan, Insomnia, Hoppscotch, cURL**, entre outros.

Como puxar dados de uma API?

Existem diferentes formas de se comunicar com APIs, sendo as mais comuns: **Fetch, Axios e React Query.**

Fetch(Nativo do JS)

É uma API nativa do navegador (e disponível no Node também) para fazer requisições HTTP.

Características:

- Retorna uma Promise.
- Precisa converter manualmente a resposta em JSON

Exemplo:

```
// Faz uma requisição HTTP para a URL especificada
fetch("https://api.exemplo.com/users")
  // Converte a resposta em JSON
  .then(res ⇒ res.json())
  // Manipula os dados obtidos
  .then(data ⇒ console.log(data))
  // Captura erros de rede (ex: sem internet)
  .catch(err ⇒ console.error(err));
```

Axios (biblioteca externa)

O Axios é uma biblioteca que facilita requisições HTTP, que tem como característica a sintaxe mais simples e legível que fetch.

```
// Importa a biblioteca axios
import axios from "axios";

// Faz uma requisição GET para a URL especificada
axios.get("https://api.exemplo.com/users")
  // A resposta já vem convertida em JSON em res.data
  .then(res ⇒ console.log(res.data))
  // Captura erros, incluindo respostas HTTP de erro (404, 500, etc)
  .catch(err ⇒ console.error(err));
```

React Query (ou TanStack Query)
O React Query é uma biblioteca que não serve só para fazer requisições, mas faz o gerenciamento de estado assíncrono.

Características:

- Pode usar fetch, axios ou qualquer outra forma de obter dados.
- O diferencial está em:
  - Cache de dados da API (evita refetch desnecessário).
  - Refetch automático quando o usuário volta para a aba ou a internet reconecta.
  - Mutations (para POST, PUT, DELETE) com optimistic updates.
  - Controle de estados (loading, error, success) prontos para usar.
  - Revalidação automática (dados sempre frescos).

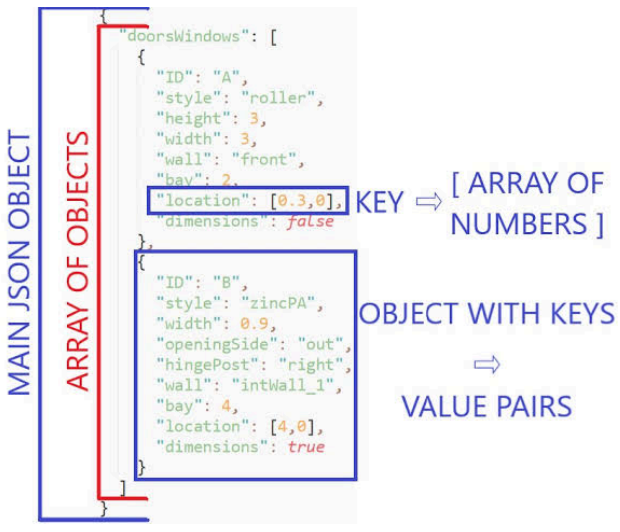
```jsx
// Importa o hook useQuery da biblioteca React Query
import { useQuery } from "@tanstack/react-query";
// Importa axios para realizar a requisição HTTP
import axios from "axios";

function Users() {
  // useQuery gerencia o estado assíncrono da API
  const { data, error, isLoading } = useQuery({
    queryKey: ["users"], // Chave única para o cache desta query
    queryFn: () =>
      // Função que retorna os dados da API usando axios
      axios.get("https://api.exemplo.com/users").then(res => res.data)
  });

  // Enquanto os dados estão carregando
  if (isLoading) return <p>Carregando...</p>;
  // Se ocorrer algum erro
  if (error) return <p>Erro: {error.message}</p>;

  // Renderiza a lista de usuários
  return (
    <ul>
      {data.map(user => <li key={user.id}>{user.name}</li>)}
    </ul>
  );
}
```

Qual o formato de resposta?



A maioria das APIs modernas usa **JSON (JavaScript Object Notation)**, pois é um formato legível por humanos e fácil de usar em JavaScript, sendo representado por dados em objetos e listas (arrays).

E quando a API é protegida?

Nem toda API é pública. Algumas exigem identificação para garantir segurança:

- **API Key** → uma “chave” secreta que você envia junto da requisição.
- **Bearer Token / JWT** → um token temporário que prova que você tem permissão.
- **OAuth** → login com serviços de terceiros (Google, Facebook) para liberar acesso.

Como saber se deu certo?

Quando você faz uma requisição, o servidor retorna um código de status que indica o resultado:

Código	Significado	Exemplo de uso / descrição
200	Sucesso	A API devolveu os dados corretamente
201	Criado	Um novo recurso foi criado com sucesso
204	Sem conteúdo	A operação foi bem-sucedida, mas não há dados para retornar
400	Erro do cliente	A requisição estava incorreta (dados faltando ou inválidos)
401	Não autorizado	Falta autenticação ou token inválido
403	Proibido	Você não tem permissão para acessar o recurso
404	Não encontrado	O recurso solicitado não existe
409	Conflito	Tentativa de criar ou atualizar um recurso que já existe
500	Erro do servidor	Problema interno na API, algo deu errado no servidor
503	Serviço indisponível	O servidor está temporariamente fora do ar

Não sei construir APIs e agora?

Existem sites e bibliotecas que podem auxiliar quem precisa utilizar APIs de forma temporária:

MockAPI

Cria API diretamente pelo navegador gerando endpoints de acesso.

https://mockapi.io/

FakerJS

Biblioteca que gera massa de dados fake para a fase de testes e desenvolvimento da aplicação.

https://fakerjs.dev/

JSON placeholder

Retorna dados fake para teste e prototipação.

https://jsonplaceholder.typicode.com/

Referências

https://developer.mozilla.org/pt-BR/docs/Learn_web_development/Extensions/Client-side_APIs/Introduction

https://ijaycent.hashnode.dev/getting-started-with-application-programming-interface-api

https://ijaycent.hashnode.dev/public-apis-developers-can-use-in-their-projects

https://technologyadvice.com/blog/information-technology/how-to-use-an-api/

https://github.com/public-apis/public-apis