**Aula XXII - DevAppII**

**API REST**

**HTTP Métodos**

**HTTP Dependências**

**Consumindo API REST em Flutter**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fontes e Referências**

[Fetch data from the internet](https://flutter.dev/docs/cookbook/networking/fetch-data)

[Aprendendo a consumir uma API no Flutter em 1 minuto.](https://blog.flutterando.com.br/aprendendo-a-consumir-uma-api-no-flutter-em-1-minuto-b03c2f93f9af)

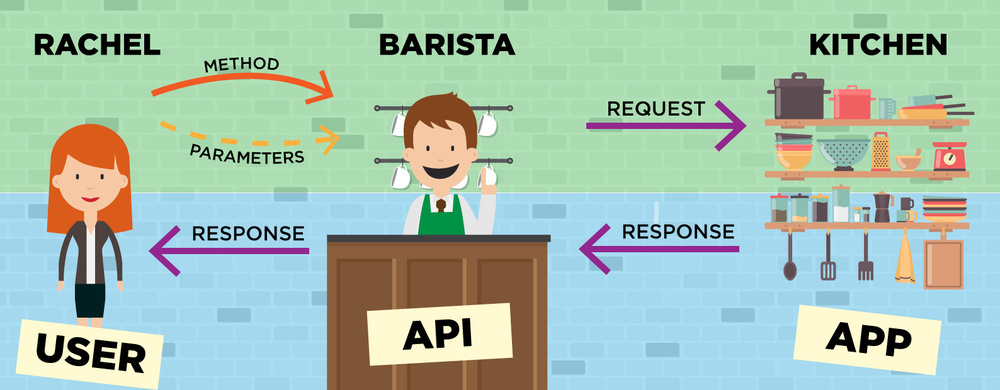
[Consuming APIs in Flutter | Engineering Education (EngEd) Program](https://www.section.io/engineering-education/consuming-apis-in-flutter/)

[JSONPlaceholder - Free Fake REST API](https://jsonplaceholder.typicode.com/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Afinal, o que é uma API REST?**

**API:** vem do acrônimo API que provém do inglês **Application Programming Interface (Em português, significa Interface de Programação de Aplicações)**, trata-se de um conjunto de rotinas e padrões estabelecidos e documentados por uma aplicação A, para que outras aplicações consigam utilizar as funcionalidades desta aplicação A, sem precisar conhecer detalhes da implementação do software. Gostou? Desta forma, entendemos que as APIs permitem uma interoperabilidade entre aplicações. **Em outras palavras, a comunicação entre aplicações e entre os usuários.**



*Fonte img:* [*https://programathor.com.b*](https://programathor.com.b)*r*

Resumindo: Cada página na internet é armazenada em algum lugar em um servidor remoto. Um servidor remoto não é tão místico, afinal, é apenas uma parte de um computador localizado remotamente, otimizado para processar solicitações. Para colocar as coisas em perspectiva, podemos criar um servidor no notebook capaz de servir um site inteiro na Web.

Quando digitamos uma URL no navegador, uma solicitação é enviada para o servidor remoto onde este site está armazenado. Quando o navegador recebe a resposta, ele interpreta o código e exibe a página. Simples, portanto, para o navegador, também conhecido como cliente, o servidor da do site é uma API!! Isso significa que toda vez que visitamos uma página na Web, interagimos com a API de alguns servidores remotos.

**API REST?**

**API REST, também chamada de API RESTful,** é uma interface de programação de aplicações (API ou API web) que está em conformidade com as restrições do estilo de arquitetura REST, permitindo a interação com serviços web RESTful. **REST é a sigla em inglês para transferência representacional de estado.**

Então, é uma **Application Programming Interface (Interface de Programa de Aplicativo - API) que usa requisições HTTP para obter ou enviar dados entre sistemas.** É um tipo de comunicação entre clientes e servidores. A forma mais básica de uma API REST é uma URL que o cliente usa para fazer uma solicitação ao servidor. Ao receber uma solicitação bem-sucedida, o servidor verifica o endpoint da URL, faz algum processamento e envia os dados ou recursos solicitados de volta ao cliente.

**Métodos HTTP**

**Existem quatro métodos HTTP principais que podemos usar nas APIs REST.**

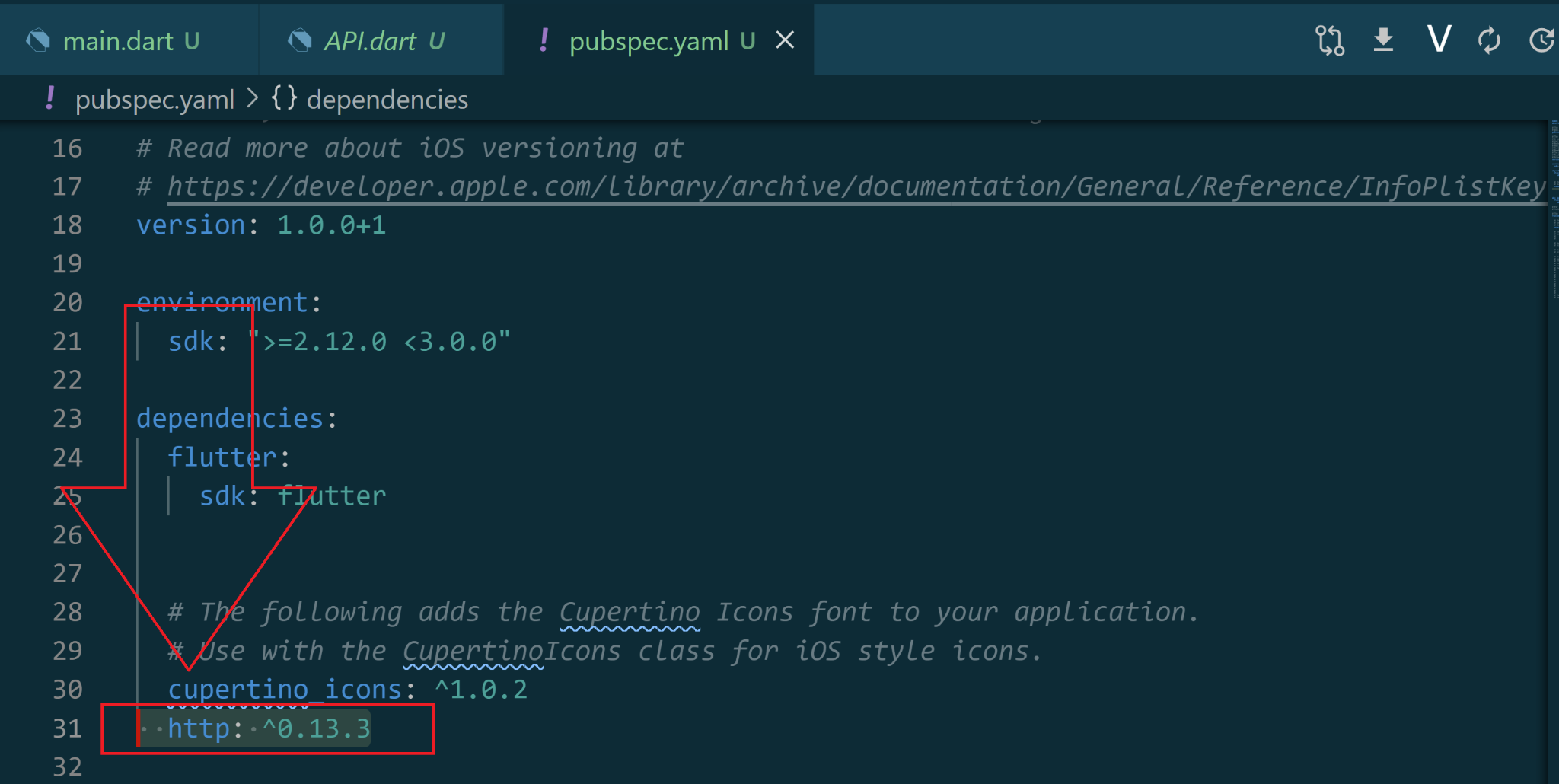
1. **GET:** Solicita uma representação do recurso especificado. As solicitações que usam GET somente recuperam dados.
2. **POST:** Envia dados para o recurso especificado. Você usa esse método para enviar dados ao servidor.
3. **DELETE :** Exclui o recurso especificado
4. **PUT:** Substitui todas as representações atuais do recurso de destino pelo conteúdo carregado.

**Prática e Dependências do HTTP**

Para este modelo, iremos usar o [{JSON} Placeholder](https://jsonplaceholder.typicode.com/), que nada mais é que uma API falsa gratuita para teste e prototipagem, aliás, bem útil e interessante para quando precisamos criar testes de aplicação.

Primeiro vamos criar um projeto do zero em Flutter no VsCode ou Android Studio …, como preferirem. Após criamos o projeto, vamos direto para o nosso arquivo **pubspec.yaml,** para declararmos uma dependência de HTPP!

Logo abaixo da referência do cupertino …



**Observação!!** Se você está dev no VsCode, clique sobre o código e salve (CRTL S) para que ele exporte as novas extensões do Dart com as dependências que precisamos. Se você está dev no Android Studio, clique sobre o **link pub get,** ou ainda, no modo terminal de ambas as IDEs, digite: **flutter pub get**

**Vamos para a main!**

Observem as importações da nossa main:



**Lib dart:convert**

Ref: [dart:convert library](https://api.dart.dev/stable/2.13.4/dart-convert/dart-convert-library.html)

Codificadores e decodificadores para conversão entre diferentes representações de dados, incluindo JSON e UTF-8. Além de conversores para representações de dados comuns, esta biblioteca fornece suporte para a implementação de conversores de uma forma que os torna fáceis de encadear e usar com fluxos, e usamos como a linha 1 na imagem acima.

Dois conversores normalmente usados ​​são as instâncias de nível superior de **JsonCodec e Utf8Codec** , denominados json e utf8 , respectivamente e estes “Srs.” já conhecemos de longa data, desde os estudos de HTML.

* **JSON é um formato de texto simples para representar coleções e objetos estruturados.** O codificador / decodificador JSON se transforma entre strings e estruturas de objetos, como listas e mapas, usando o formato JSON.
* **UTF-8 é uma codificação comum de largura variável que pode representar todos os caracteres no conjunto de caracteres Unicode**. O codificador / decodificador UTF-8 se transforma entre Strings e bytes.