# Tutorial: Integração com APIs no Flutter

## Objetivo

Nesta etapa, vamos revisar como integrar o Flutter com APIs REST, utilizando o pacote `http`. Vamos realizar requisições GET e POST, manipular dados em JSON, e exibir esses dados em widgets do Flutter.

## Configuração do Projeto

Para começar, adicione o pacote `http` no arquivo `pubspec.yaml` do projeto Flutter:  
  
```yaml  
dependencies:  
 flutter:  
 sdk: flutter  
 http: ^0.13.3  
```  
  
Depois disso, é necessário importar a biblioteca `http` no código e configurar o projeto para realizar requisições.

## Requisição GET

Vamos realizar uma requisição GET para consumir dados de uma API pública.  
  
Exemplo de código:  
  
```dart  
import 'package:flutter/material.dart';  
import 'package:http/http.dart' as http;  
import 'dart:convert';  
  
void main() => runApp(MyApp());  
  
class MyApp extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 home: HomeScreen(),  
 );  
 }  
}  
  
class HomeScreen extends StatefulWidget {  
 @override  
 \_HomeScreenState createState() => \_HomeScreenState();  
}  
  
class \_HomeScreenState extends State<HomeScreen> {  
 String data = 'Loading...';  
  
 Future<void> fetchData() async {  
 final response = await http.get(Uri.parse('https://jsonplaceholder.typicode.com/todos/1'));  
  
 if (response.statusCode == 200) {  
 var jsonResponse = jsonDecode(response.body);  
 setState(() {  
 data = jsonResponse['title'];  
 });  
 } else {  
 setState(() {  
 data = 'Erro ao carregar dados';  
 });  
 }  
 }  
  
 @override  
 void initState() {  
 super.initState();  
 fetchData();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(title: Text('Requisição GET')),  
 body: Center(child: Text(data)),  
 );  
 }  
}  
```  
  
Explicação: Este código faz uma requisição GET para uma API pública, decodifica o JSON e exibe um valor específico.

## Requisição POST

Vamos agora realizar uma requisição POST para enviar dados a uma API.  
  
Exemplo de código:  
  
```dart  
Future<void> sendData() async {  
 final response = await http.post(  
 Uri.parse('https://jsonplaceholder.typicode.com/posts'),  
 headers: {'Content-Type': 'application/json; charset=UTF-8'},  
 body: jsonEncode({  
 'title': 'foo',  
 'body': 'bar',  
 'userId': 1,  
 }),  
 );  
  
 if (response.statusCode == 201) {  
 print('Dados enviados com sucesso');  
 } else {  
 print('Falha ao enviar dados');  
 }  
}  
```  
  
Explicação: Esse código mostra como enviar dados usando POST. O `jsonEncode` é usado para converter o mapa em JSON antes de enviar.

## Manipulação de JSON

Para manipular dados JSON, utilizamos o pacote `dart:convert`. Isso é essencial ao consumir dados de APIs que retornam JSON.  
  
Exemplo: Para converter JSON em objeto e vice-versa:  
```dart  
var jsonString = '{"name": "Alice", "age": 25}';  
var user = jsonDecode(jsonString);  
print(user['name']); // Alice  
```

## Exibição de Dados em Widgets

Para exibir dados recebidos de uma API, podemos utilizar widgets como ListView, Text, entre outros.  
  
Exemplo de exibição em ListView:  
  
```dart  
List<String> items = [];  
  
Future<void> fetchItems() async {  
 final response = await http.get(Uri.parse('https://jsonplaceholder.typicode.com/todos'));  
 if (response.statusCode == 200) {  
 List jsonResponse = jsonDecode(response.body);  
 setState(() {  
 items = jsonResponse.map((item) => item['title'].toString()).toList();  
 });  
 }  
}  
  
@override  
Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(title: Text('Lista de Tarefas')),  
 body: ListView.builder(  
 itemCount: items.length,  
 itemBuilder: (context, index) {  
 return ListTile(title: Text(items[index]));  
 },  
 ),  
 );  
}  
```  
  
Neste exemplo, utilizamos um `ListView.builder` para exibir uma lista de dados recebidos via API.