



# **INTRODUÇÃO AO ARMAZENAMENTO DE DADOS**

EM FLUTTER



- *O armazenamento de dados é uma parte essencial no desenvolvimento de aplicativos Flutter. Existem várias opções disponíveis para armazenar dados em um aplicativo, dependendo das necessidades específicas.*

*Vou introduzir algumas das principais opções de armazenamento de dados em Flutter:*

## SHARED PREFERENCES:

- O *Shared Preferences* é uma forma simples e leve de armazenar dados chave-valor. É ideal para armazenar pequenas quantidades de dados, como configurações de usuário, preferências ou informações de autenticação. Os dados são persistidos localmente e podem ser acessados rapidamente. No entanto, não é adequado para armazenar grandes volumes de dados estruturados.

## ARQUIVOS LOCAIS:

- O *Flutter* oferece suporte ao acesso ao sistema de arquivos do dispositivo, permitindo que você armazene e recupere arquivos no armazenamento local. Isso é útil para armazenar dados estruturados, como arquivos de banco de dados SQLite, arquivos JSON ou qualquer outro tipo de arquivo. Você pode usar a biblioteca *path\_provider* para obter o caminho adequado para armazenamento local em diferentes plataformas.

## BANCO DE DADOS SQLITE:

- O SQLite é um banco de dados relacional leve e embutido que pode ser usado para armazenar dados estruturados em um aplicativo Flutter. Ele oferece recursos avançados, como consultas SQL e indexação de dados, tornando-o adequado para aplicativos que exigem operações de banco de dados mais complexas. A biblioteca sqflite é amplamente usada para interagir com o SQLite em Flutter.

## ARMAZ. EM NUVEM:

- Se você precisa armazenar dados em um local centralizado e acessível em vários dispositivos, pode considerar o armazenamento em nuvem. Existem várias opções disponíveis, como o Firebase Realtime Database, o Cloud Firestore ou serviços de armazenamento de objetos, como o Amazon S3 ou o Google Cloud Storage. Esses serviços oferecem recursos de sincronização em tempo real e escalabilidade para aplicativos Flutter.

- ALÉM DESSAS OPÇÕES, TAMBÉM EXISTEM BIBLIOTECAS ADICIONAIS QUE FORNECEM SOLUÇÕES DE GERENCIAMENTO DE ESTADO E PERSISTÊNCIA, COMO O HIVE, QUE É UM BANCO DE DADOS DE CHAVE-VALOR RÁPIDO E FÁCIL DE USAR, E O MOOR, QUE É UM GERADOR DE CÓDIGO QUE FACILITA A COMUNICAÇÃO COM BANCOS DE DADOS SQLITE.

- **A ESCOLHA DO MÉTODO DE ARMAZENAMENTO DE DADOS DEPENDE DAS NECESSIDADES ESPECÍFICAS DO SEU APLICATIVO, COMO O TAMANHO DOS DADOS, A COMPLEXIDADE DAS OPERAÇÕES DE BANCO DE DADOS E OS REQUISITOS DE SINCRONIZAÇÃO DE DADOS. É IMPORTANTE CONSIDERAR A SEGURANÇA, O DESEMPENHO E A ESCALABILIDADE AO SELECIONAR A OPÇÃO DE ARMAZENAMENTO MAIS ADEQUADA PARA O SEU APLICATIVO FLUTTER.**

# **CHAMADAS DE API PARA OBTENÇÃO DE DADOS EXTERNOS**

***EM FLUTTER, VOCÊ PODE REALIZAR  
CHAMADAS DE API PARA OBTENÇÃO DE  
DADOS EXTERNOS USANDO A BIBLIOTECA  
HTTP.***

***AQUI ESTÁ UM EXEMPLO BÁSICO DE COMO  
FAZER ISSO:***

**1- CERTIFIQUE-SE DE TER ADICIONADO A DEPENDÊNCIA HTTP NO ARQUIVO PUBSPEC.YAML DO SEU PROJETO FLUTTER:**

```
dependencies:  
  flutter:  
    sdk: flutter  
  http: ^0.13.3
```

**2- Importe a biblioteca http no arquivo Dart onde você deseja realizar as chamadas de API:**

```
import 'package:http/http.dart' as http;
```

**3- USE A FUNÇÃO HTTP.GET() PARA FAZER UMA CHAMADA GET PARA A URL DA API DESEJADA. AQUI ESTÁ UM EXEMPLO BÁSICO QUE OBTÉM DADOS DE UMA API E EXIBE A RESPOSTA NO CONSOLE:**

```
void fetchData() async {  
    var url = Uri.parse('https://api.example.com/data');  
    var response = await http.get(url);  
  
    if (response.statusCode == 200) {  
        print(response.body); // Exibe a resposta no console  
    } else {  
        print('Erro na requisição. Código de status: ${response.statusCode}');  
    }  
}
```



**4- VOCÊ PODE CHAMAR A FUNÇÃO `FETCHDATA()` QUANDO NECESSÁRIO PARA OBTER OS DADOS DA API. LEMBRE-SE DE LIDAR ADEQUADAMENTE COM OS ERROS E EXCEÇÕES QUE POSSAM OCORRER DURANTE A CHAMADA DE API.**

**ESTE É APENAS UM EXEMPLO BÁSICO DE COMO REALIZAR CHAMADAS DE API EM FLUTTER USANDO A BIBLIOTECA `HTTP`. É IMPORTANTE NOTAR QUE EXISTEM OUTRAS BIBLIOTECAS E ABORDAGENS DISPONÍVEIS PARA TRABALHAR COM CHAMADAS DE API EM FLUTTER, COMO O USO DE PACOTES COMO `DIO` OU A CRIAÇÃO DE CLASSES E MODELOS PERSONALIZADOS PARA FACILITAR A SERIALIZAÇÃO E DESSERIALIZAÇÃO DE DADOS.**

**RECOMENDA-SE CONSULTAR A DOCUMENTAÇÃO DESSAS BIBLIOTECAS E EXPLORAR EXEMPLOS MAIS AVANÇADOS PARA OBTER MAIS INFORMAÇÕES.**

# TRATAMENTO DE ERROS E FEEDBACK AO USUÁRIO

NO PROCESSO DE CHAMADAS DE API EM FLUTTER, É IMPORTANTE REALIZAR O TRATAMENTO DE ERROS E FORNECER FEEDBACK ADEQUADO AO USUÁRIO PARA GARANTIR UMA EXPERIÊNCIA DE USO MELHOR.

**AQUI ESTÃO ALGUMAS PRÁTICAS COMUNS PARA LIDAR COM ERROS E FORNECER FEEDBACK AO USUÁRIO:**

1 - VERIFICAR O CÓDIGO DE STATUS DA RESPOSTA: AO RECEBER A RESPOSTA DA CHAMADA DE API, VOCÊ PODE VERIFICAR O CÓDIGO DE STATUS HTTP PARA DETERMINAR SE A SOLICITAÇÃO FOI BEM-SUCEDIDA OU SE OCORREU ALGUM ERRO. OS CÓDIGOS DE STATUS COMUNS SÃO 2XX PARA SUCESSO, 4XX PARA ERROS DO CLIENTE E 5XX PARA ERROS DO SERVIDOR.

2- Tratar erros de conexão: É importante lidar com erros de conexão, como falhas na rede ou problemas de conectividade. Você pode envolver a chamada de API em um bloco try-catch para capturar exceções e fornecer uma mensagem de erro adequada ao usuário.

```
try {  
    var response = await http.get(url);  
    // Processar a resposta da API  
} catch (e) {  
    print('Erro na conexão: $e');  
    // Exibir mensagem de erro para o usuário  
}
```

**3- EXIBIR MENSAGENS DE ERRO AO USUÁRIO: QUANDO OCORRER UM ERRO DURANTE A CHAMADA DE API, VOCÊ PODE EXIBIR UMA MENSAGEM DE ERRO RELEVANTE PARA O USUÁRIO. ISSO PODE SER FEITO POR MEIO DE UM DIÁLOGO, UM SNACKBAR, UM TOAST OU QUALQUER OUTRO COMPONENTE DE INTERFACE DO USUÁRIO ADEQUADO.**

```
ScaffoldMessenger.of(context).showSnackBar(  
  SnackBar(  
    content: Text('Ocorreu um erro na chamada de API.'),  
  ),  
);
```

**4- TRATAR ERROS ESPECÍFICOS DA API: ALÉM DOS ERROS DE CONEXÃO, A API PODE RETORNAR ERROS ESPECÍFICOS COM INFORMAÇÕES ÚTEIS. POR EXEMPLO, UMA RESPOSTA COM CÓDIGO DE STATUS 400 PODE INDICAR QUE OS DADOS DA SOLICITAÇÃO ESTÃO INCORRETOS. VOCÊ PODE ANALISAR A RESPOSTA DA API E TRATAR ESSES ERROS ESPECÍFICOS DE ACORDO.**

```
if (response.statusCode == 400) {  
    var errorResponse = json.decode(response.body);  
    var errorMessage = errorResponse['message'];  
    // Exibir mensagem de erro específica para o usuário  
}
```

5 -FORNECER FEEDBACK DE CARREGAMENTO: DURANTE A CHAMADA DE API, É COMUM EXIBIR UM INDICADOR DE CARREGAMENTO PARA INFORMAR AO USUÁRIO QUE A SOLICITAÇÃO ESTÁ EM ANDAMENTO. ISSO PODE SER FEITO USANDO UM CIRCULARPROGRESSINDICATOR OU OUTRO COMPONENTE VISUAL ADEQUADO.

ESSAS SÃO APENAS ALGUMAS PRÁTICAS BÁSICAS PARA LIDAR COM ERROS E FORNECER FEEDBACK AO USUÁRIO DURANTE CHAMADAS DE API EM FLUTTER. VOCÊ PODE ADAPTAR ESSAS PRÁTICAS DE ACORDO COM OS REQUISITOS DO SEU APLICATIVO E A LÓGICA ESPECÍFICA DAS CHAMADAS DE API QUE ESTÁ IMPLEMENTANDO.

# FONTES

[HTTPS://WWW.DEVMEDIA.COM.BR/GUIA/FLUTTER/40713](https://www.devmedia.com.br/guia/flutter/40713)

[HTTPS://QASTACK.COM.BR/PROGRAMMING/41369633/HOW-TO-SAVE-TO-LOCAL-STORAGE-USING-FLUTTER](https://qastack.com.br/programming/41369633/how-to-save-to-local-storage-using-flutter)

[HTTPS://BR.ATSIT.IN/ARCHIVES/105497](https://br.atsit.in/archives/105497)

[HTTPS://WWW.ALURA.COM.BR/ARTIGOS/OBTER-DADOS-INTERNET-FLUTTER-USANDO-HTTP](https://www.alura.com.br/artigos/obter-dados-internet-flutter-usando-http)

[HTTPS://ICHI.PRO/PT/MANEIRA-FACIL-DE-CRIAR-UM-BOM-TRATAMENTO-DE-ERROS-NO-FLUTTER-COM-DARTZ-74802447729083](https://ichi.pro/pt/maneira-facil-de-criar-um-bom-tratamento-de-erros-no-flutter-com-dartz-74802447729083)