VOLTAR

# Projeto de um App com armazenamento interno - ToDo List

Nessa aula vamos aprender a criar uma aplicação que pode até ser bastante usual e que armazena dados internamente, para que sejam persistidos, ou seja, mantidos, mesmo se a aplicação for encerrada.

NESTE TÓPICO

Marcar tó

## O que vamos desenvolver?

Nessa aula vamos desenvolver um aplicativo bastante simples, mas muito interessante e muito útil que é um simples aplicativo de "ToDo list", ou seja, um app para gerenciarmos nossas listas de tarefas.

Este aplicativo armazenará seus dados internamente, em uma estrutura de arquivos, mas utilizando um arquivo em formato JSON, conforme já exploramos no aplicativo de clima; a única diferença aqui é que usaremos um arquivo guardado gerenciado pela própria aplicação.

Neste aplicativo vamos incluir novas funcionalidades muito legais como o ? dismissible? (dispensável) que permite arrastarmos um item da lista para o lado para deletá-lo.

O vídeo abaixo mostra como será o aplicativo desenvolvido com os recursos muito interessantes que iremos implementar nessa aula.

Apresentação do App que vamos desenvolver

Esperamos que você esteja bastante empolgado(a) para iniciar o desenvolvimento deste app muito legal. Vamos começar.

Uma observação importante: Nessa aula é muito importante você assistir os vídeos na ordem, pois muitos conceitos são explicados em cada um dos vídeos e a construção deste aplicativo é incremental, ou seja, ocorre em vários passos ordenados.

## Criação do projeto e definição dos temas

Vamos começar a criar o projeto e definir os temas (claro e escuro) do aplicativo. Para isso, nada melhor do que um vídeo explicando o passo a passo disso e, por isso, o vídeo abaixo descreve como o fazer.

Criação do projeto e definição dos temas

É muito importante aprendermos bem como definir temas em nossas aplicações pois muitas empresas utilizam este conceito, especialmente para temas "light" (claros) e "dark" (escuros). O próprio WhatsApp possui estes

temas.

Agora que já criamos nosso projeto e os temas, vamos começar a desenvolvê-lo, de fato.

## Criando o corpo da aplicação

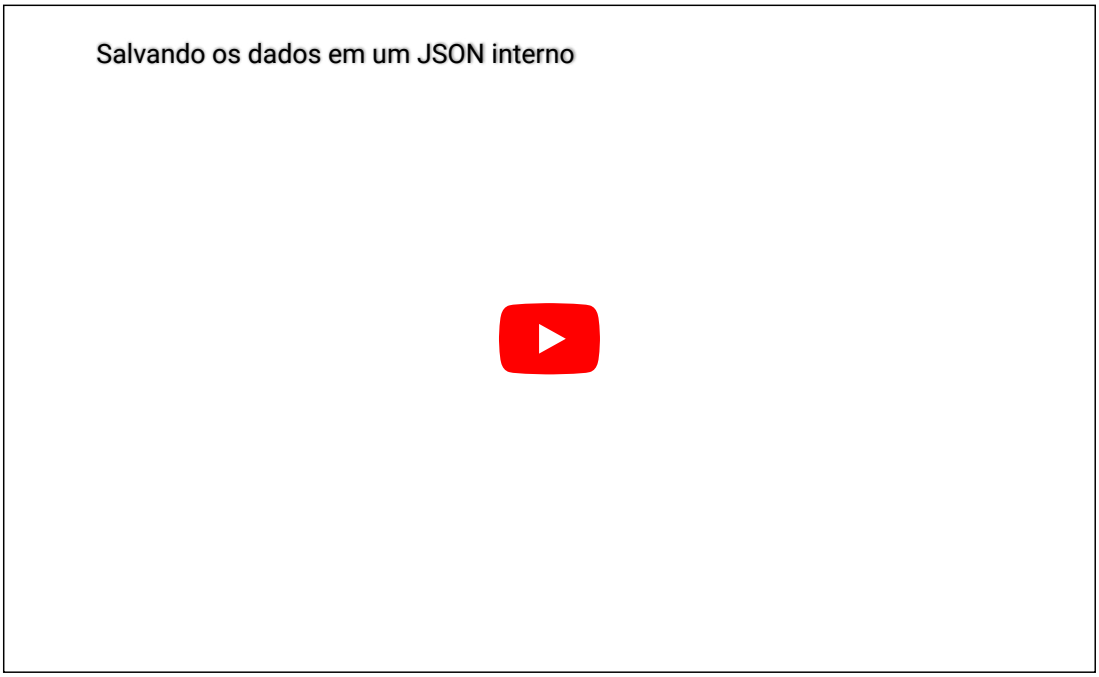
Agora que já temos o projeto criado, podemos começar a criar o corpo da aplicação, especialmente o campo de texto para acrescentar uma nova tarefa ao sistema.

Para isso, o vídeo abaixo mostra como o fazer, explicando vários conceitos importantes desta etapa.


## Salvando os dados em um JSON interno

Agora que já criamos o campo e o botão para acrescentar uma nova tarefa à nossa lista, vamos começar a desenvolver a lógica de gerar um JSON e o gravar internamente.

Para isso, assista o vídeo abaixo que mostra como realizar este procedimento programaticamente. Você poderá observar que o processo é bastante simples.




## Criando o Widget para os "ToDos"

Como sabemos, cada tarefa no sistema é representada por um item da lista. Cada um destes itens precisa ser "montado" para exibição.

O vídeo abaixo mostra como o fazer criando-se um wodget (ferramenta) própria para isso, programaticamente e de forma muito simples.

O legal deste processo é que usaremos bastante em nossos próximos projetos a criação de Widgets personalizados por nós mesmos.


## Desfazendo a ação de deleção de tarefa

Um recurso muito útil e interessante que vamos implementar em nós aplicação é da ação de desfazer a deleção de uma tarefa. Este é um recurso muito útil pois muitas vezes o usuário pode deletar sem querer um item e se

arrepender logo em seguida.

Para implementar este recurso necessário utilizarmos um "widget" que seja dispensável (dismissible, no Flutter). O vídeo abaixo mostra como fazer isso:

Implementação do "desfazer" da deleção de uma tarefa

## Carregando a lista de tarefas do sistema de arquivos

Agora que já temos a listagem de tarefas, precisamos implementar um detalhe muito importante em relação ao aplicativo: O carregamento do arquivo JSON na inicialização.

Para isso precisamos implementar o método "initState" que é executado assim que o Widget é construído em memória.

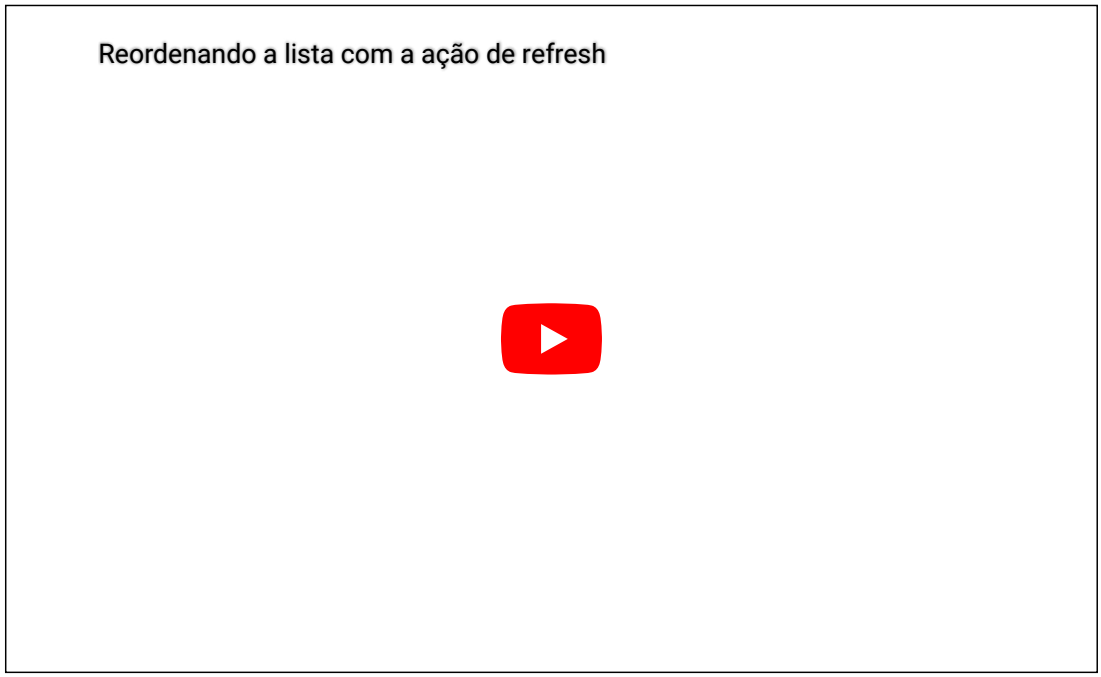
Para implementarmos isso, então, veja o vídeo abaixo.

Carregamento da lista ao abrir a aplicação

## Reordenação da lista

Prontinho, nossa aplicação está quase pronta e está ficando maravilhosa, mas ainda precisamos implementar um detalhe que a deixa ainda mais poderosa: A reordenação da lista ao ser "recarregada", ou seja, quando o usuário "puxar" a lista para baixo.

Para isso, veja o vídeo abaixo:




## E onde está o arquivo JSON?

É muito importante sabermos onde o arquivo JSON foi guardado e sim, podemos acessá-lo dentro do celular.

Para isso, veja o vídeo abaixo.

Como você pode ter notado, é possível acessar os arquivos das aplicações facilmente. É muito comum usarmos este processo para armazenarmos referencias do usuário, configurações definidas pelo usuário etc. E é por isso que temos que tomar muito cuidado com os dados que são armazenados neste tipo de arquivo, pois podem ser lidos por outros não autorizados e, às vezes, mal intencionados. Lembre-se de nunca armazenar senhas em arquivos como este.

Abaixo você pode também acessar o código-fonte da aplicação.

Arquivo main.dart:

```
import 'package:flutter/material.dart';
import 'package:todo_app2/screens/home.dart';
import 'package:todo_app2/theme/theme.dart';

void main() {
  runApp(MaterialApp(
    debugShowCheckedModeBanner: false,
    title: "Todo List",
    home: Home(),
    themeMode: ThemeMode.system,
    theme: lightTheme(),
    darkTheme: darkTheme(),
  ));
}
```

Arquivo "theme.dart":

```dart
import 'package:flutter/material.dart';

ThemeData darkTheme() {
  return ThemeData(
      brightness: Brightness.dark,
      primaryColor: Colors.black,
      accentColor: Colors.orange,
      appBarTheme: AppBarTheme(color: Colors.grey),
      scaffoldBackgroundColor: Colors.grey[800],
      iconTheme: IconThemeData(color: Colors.white),
      hintColor: Colors.orange,
      floatingActionButtonTheme:
          FloatingActionButtonThemeData(backgroundColor: Colors.white70),
      textTheme: TextTheme(
          headline4: TextStyle(fontSize: 23.0, fontWeight: FontWeight.bold),
          headline5: TextStyle(fontSize: 23.0, fontWeight: FontWeight.w200)));
}

ThemeData lightTheme() {
  return ThemeData(
      brightness: Brightness.light,
      primaryColor: Colors.purple,
      accentColor: Colors.orange,
      appBarTheme: AppBarTheme(color: Colors.deepPurple),
      scaffoldBackgroundColor: Colors.white,
      iconTheme: IconThemeData(color: Colors.white),
      hintColor: Colors.orange,
      floatingActionButtonTheme:
          FloatingActionButtonThemeData(backgroundColor: Colors.orange),
      textTheme: TextTheme(
          headline4: TextStyle(fontSize: 23.0, fontWeight: FontWeight.bold),
          headline5: TextStyle(fontSize: 23.0, fontWeight: FontWeight.w200)));
}
```

E finalmente o arquivo "home.dart":

```
import 'dart:convert';
import 'dart:io';
import 'package:flutter/cupertino.dart';
import 'package:path_provider/path_provider.dart';

import 'package:flutter/material.dart';

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> {
  List _toDoList = [];
  final _toDoController = TextEditingController();
  int _indeceUltimoRemovido;
  Map<String, dynamic> _ultimoRemovido;


  @override
  void initState() {
    super.initState();
    _lerDados().then((value) {
      setState(() {
        _toDoList = json.decode(value);
      });
    });
  }

  Future<String> _lerDados() async {
    try {
      final arquivo = await _abreArquivo();
      return arquivo.readAsString();
    } catch (e) {
      return null;
    }
  }

  void _adicionaTarefa() {
    setState(() {
      Map<String, dynamic> novoTodo = Map();
      novoTodo["titulo"] = _toDoController.text;
      novoTodo["realizado"] = false;
      _toDoController.text = "";
      _toDoList.add(novoTodo);
      _salvarDados();
    });
  }

  Future<File> _salvarDados() async {
    String dados = json.encode(_toDoList);
    final file = await _abreArquivo();
    return file.writeAsString(dados);
  }

  Future<File> _abreArquivo() async {
    final diretorio = await getApplicationDocumentsDirectory();
    return File("${diretorio.path}/todoList.json");
  }

  Future<Null> _recarregaLista() async {
    await Future.delayed(Duration(seconds: 1));
    setState(() {
      _toDoList.sort((a, b) {
        if (a["realizado"] && !b["realizado"])
          return 1;
        if (!a["realizado"] && b["realizado"])
          return -1;
        return 0;
      });
      _salvarDados();
```

```
    });
    return null;
  }


  Widget widgetTarefa(BuildContext context, int index) {
    return Dismissible(
      key: Key(DateTime.now().microsecondsSinceEpoch.toString()),
      background: Container(
        color: Colors.red,
        child: Align(
          alignment: Alignment(0.85, 0.0),
          child: Icon(Icons.delete_sweep_outlined, color: Colors.white,),
        ),
      ),
      direction: DismissDirection.endToStart,
      child: CheckboxListTile(
        title: Text(_toDoList[index]["titulo"]),
        value: _toDoList[index]["realizado"],
        secondary: CircleAvatar(
          child: Icon(
            _toDoList[index]["realizado"] ? Icons.check : Icons.error,
            color: Theme.of(context).iconTheme.color,
          ),
          backgroundColor: Theme.of(context).primaryColor,
        ),
        onChanged: (value) {
          setState(() {
            _toDoList[index]["realizado"] = value;
            _salvarDados();
          });
        },
        checkColor: Theme.of(context).primaryColor,
        activeColor: Theme.of(context).secondaryHeaderColor,
      ),
      onDismissed: (direction) {
        setState(() {
          _ultimoRemovido = Map.from(_toDoList[index]);
          _indeceUltimoRemovido = index;
          _toDoList.removeAt(index);
          _salvarDados();
        });

        final snack = SnackBar(
            content: Text("Tarefa \"${_ultimoRemovido["titulo"]}\" apagada!"),
          action: SnackBarAction(
            label: "Desfazer",
            onPressed: () {
              setState(() {
                _toDoList.insert(_indeceUltimoRemovido, _ultimoRemovido);
                _salvarDados();
              });
            },
          ),
          duration: Duration(seconds: 4),
        );
        Scaffold.of(context).removeCurrentSnackBar();
        Scaffold.of(context).showSnackBar(snack);
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("ToDo List"),
        centerTitle: true,
      ),
```

```
      body: Builder(
        builder: (context) => Column(
          children: [
            Container(
              padding: EdgeInsets.fromLTRB(17.0, 1.0, 7.0, 1.0),
              child: Row(
                children: [
                  Expanded(
                      child: TextField(
                    controller: _toDoController,
                    maxLength: 50,
                    decoration: InputDecoration(labelText: "Nova tarefa"),
                  )),
                  Container(
                    height: 45.0,
                    width: 45.0,
                    child: FloatingActionButton(
                      child: Icon(Icons.save),
                      onPressed: () {
                        if (_toDoController.text.isEmpty) {
                          final alerta = SnackBar(
                            content: Text('Não pode ser vazia!'),
                            duration: Duration(seconds: 4),
                            action: SnackBarAction(
                              label: 'Ok',
                              onPressed: () {
                                Scaffold.of(context).removeCurrentSnackBar();
                              },
                            ),
                          );

                          Scaffold.of(context).removeCurrentSnackBar();
                          Scaffold.of(context).showSnackBar(alerta);
                        } else {
                          _adicionaTarefa();
                        }
                      },
                    ),
                  )
                ],
              ),
            ),
            Padding(padding: (EdgeInsets.only(top: 10.0))),
            Expanded(
              child: RefreshIndicator(
                onRefresh: _recarregaLista,
                child: ListView.builder(
                  itemBuilder: widgetTarefa,
                  itemCount: _toDoList.length,
                  padding: EdgeInsets.only(top: 10.0),
                ),
              ),
            )
          ],
        ),
      ),
    );
  }
}
```

E, por fim mas não menos importante, se você precisar fazer o download do projeto completo, clique no link abaixo:

MATERIAL COMPLEMENTAR

(Https://Img.Uninove.Br/Static/

0/0/0/0/0/0/9/0/0/8/9/9008969/Todo-App2.7z)

## Quiz

Exercício Final

Projeto de um App com armazenamento interno - ToDo List

INICIAR

# Referências


DART. **Dart documentation**. Site. Disponível em: https://dart.dev/. Acesso em: 08 dez. 2020.

Material Design. **Material Design documentation**. Site. Disponível em: https://material.io/. Acesso em: 08 dez. 2020.

Flutter. **Flutter docs.** Site disponível em: https://flutter.dev. Acesso em: 09 dez.2020.

WINDMILL, Eric. **Flutter in action**. Nova Iorque: Manning publications, 2020. *E-book.* Disponível em: https://learning.oreilly.com/library/view/flutter-in-action/9781617296147/. Acesso em: 08 dez. 2020.

SINHA, Sanjib . **Quick start guide to Dart programming**: create high performance applications for the web and mobile. Lompoc, CA, EUA: Apress, 2019. *E-book*. Disponível em: https://learning.oreilly.com/library/view/quick-start-guide/9781484255629/. Acesso em: 09 dez. 2020.

ALESSANDRIA, Simone. **Flutter projects**: a practical, Project-based guide to building real-words cross-platform mobile applications and games. Birmingham, Reino Unido: Packt Publishing, 2020. *E-book*. Disponível em: https://learning.oreilly.com/library/view/flutter-projects/9781838647773/. Acesso em: 09 dez. 2020.

ZACCAGNINO, Carmine. **Programming Flutter**. [s.l.]: The Pragmatic Bookshelf, 2020. *E-book*. Disponível em: https://learning.oreilly.com/library/view/programming-flutter/9781680507621/. Acesso em: 09 dez. 2020

Avalie este tópico

ANTERIOR
Projeto de um app consumindo JSON de uma API real

Índice

PRÓXIMO
Integração do App com o Firebase