



Persistência de Dados

Parte 1

Profa. Ivre Marjorie
(ivre@pucminas.br)

O que vamos aprender nessa aula?

- Nessa aula vamos usar o **Shared Preference** para armazenar os dados no dispositivo móvel

Introdução

- **Persistir os dados**, significa armazená-los, salvá-los em arquivo ou em banco de dados, de tal forma, que é possível recuperar esses dados quando o APP for executado novamente
- Em aplicativos para dispositivos móveis é possível persistir os dados no Shared Preference, outros arquivos, banco de dados no próprio dispositivo móvel ou na nuvem

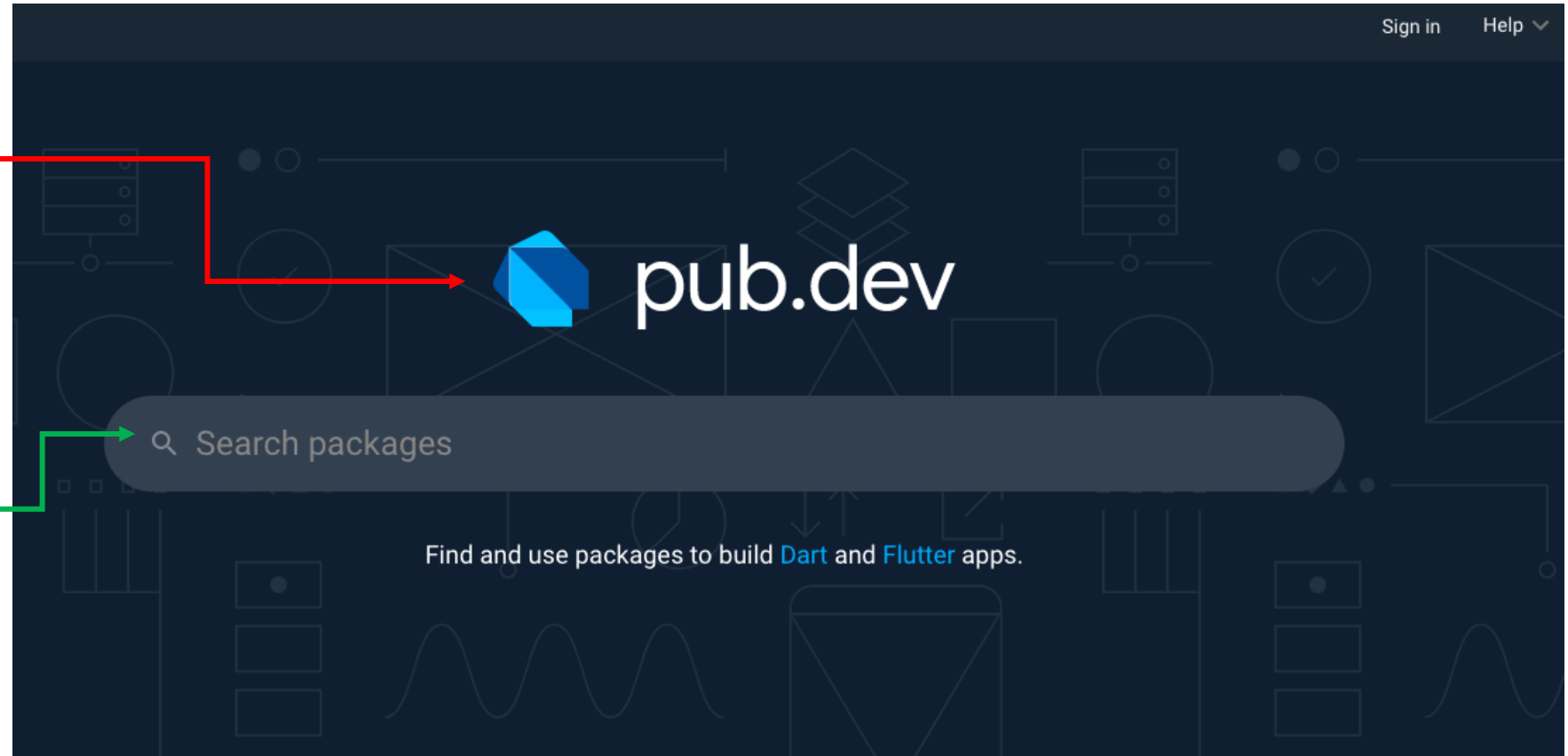
Introdução

- **Shared Preference** (Preferências compartilhadas): é uma das formas de armazenamento de dados no próprio dispositivo móvel e serve para armazenar e recuperar pequenas quantidades de dados
- É usado para armazenar pequenas quantidades de dados que estão associados ao aplicativo, sendo assim, quando o usuário desinstalar o APP o arquivo é excluído

Pacotes

Nesse site encontramos diversos recursos que podemos utilizar com o flutter através de pacotes

É possível realizar um busca. A tela principal traz os mais utilizados



Pacotes

- Ao escolher um recurso, a pagina apresenta uma série de coisas, tais como exemplo, instalação (que basicamente, será uma dependência que iremos copiar) e versões com o que mudou

shared_preferences 0.5.12

Published Sep 23, 2020 • flutter.dev

FLUTTER ANDROID IOS WEB

1.67K

Readme Changelog Example Installing Versions Scores

Shared preferences plugin

pub v0.5.12

Wraps platform-specific persistent storage for simple data (NSUserDefaults on iOS and macOS, SharedPreferences on Android, etc.). Data may be persisted to disk asynchronously, and there is no guarantee that writes will be persisted to disk after returning, so this plugin must not be used for storing critical data.

Please set your constraints

Backward compatibility

The plugin has reached a new version with 0.5.y+z. Please use the new version to allow a smoother ecosystem transition. <https://github.com/flutter/flutter>

Readme Changelog Example **Installing** Versions Scores

Use this package as a library

1. Depend on it

Add this to your package's pubspec.yaml file:

```
dependencies:  
  shared_preferences: ^0.5.12
```

Exemplo

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

void main() {
  runApp(MaterialApp(
    home: Scaffold(
      body: Center(
        child: RaisedButton(
          onPressed: _incrementCounter,
          child: Text('Increment Counter'),
        ),
      ),
    ),
  ));
}

_incrementCounter() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  int counter = (prefs.getInt('counter') ?? 0) + 1;
  print('Pressed $counter times.');
```

await prefs.setInt('counter', counter);

```
}
```

Shared Preference - Instalação

Para usar o Shared Preference
precisamos incluir uma
dependência no arquivo
pubspec.yaml

```
environment:  
  sdk: ">=2.7.0 <3.0.0"  
  
dependencies:  
  shared_preferences: ^0.5.12  
  flutter:  
    sdk: flutter
```

[Readme](#) [Changelog](#) [Example](#) [Installing](#) [Versions](#) [Scores](#)

Use this package as a library

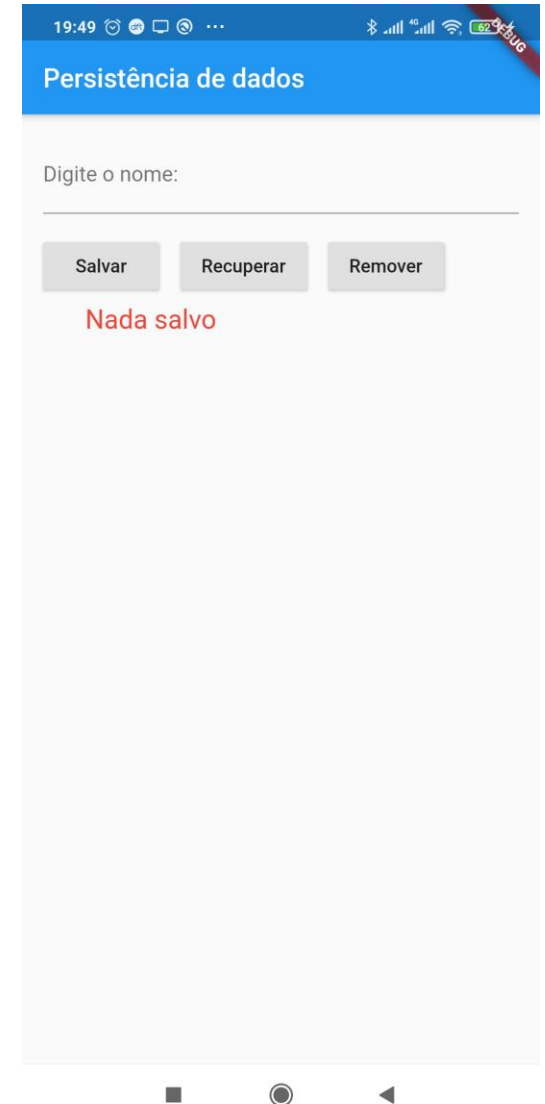
1. Depend on it

Add this to your package's pubspec.yaml file:

```
dependencies:  
  shared_preferences: ^0.5.12
```

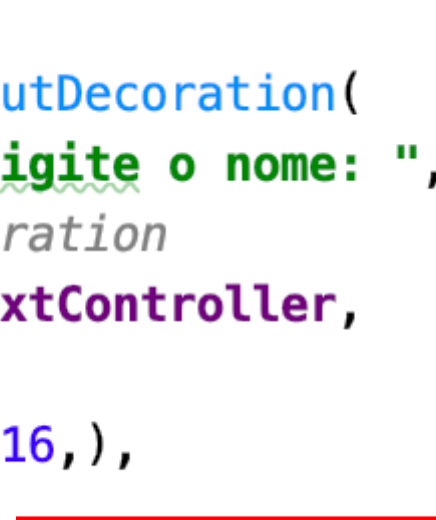

Shared Preference - Exemplo

- Para o nosso exemplo, vamos criar uma interface com um TextField, três botões (Salvar, Recuperar e Remover) e um campo texto
- Para montar essa interface, foram usados uma coluna (Column) e dentro dela duas linhas (Row), observe o código no slide a seguir, e inclusive os alinhamentos



Shared Preference - Exemplo

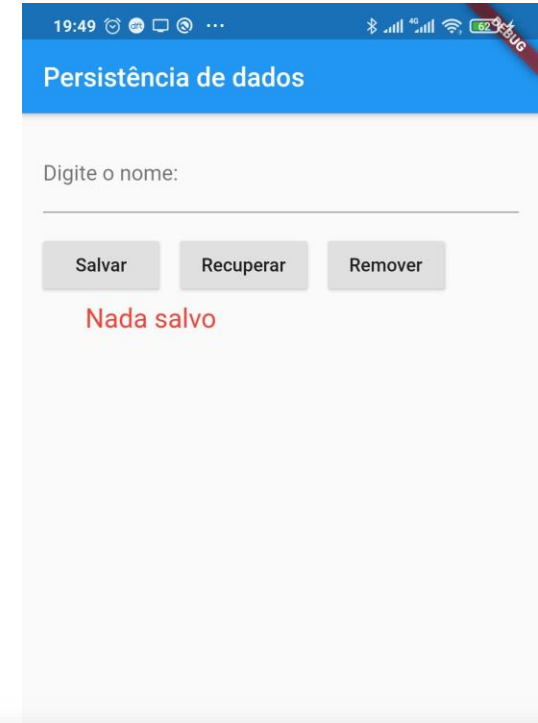
```
-body: Container(  
  padding: EdgeInsets.all(16),  
  child: Column(  
    children: <Widget>[  
      TextField(  
        decoration: InputDecoration(  
          labelText: "Digite o nome: ",  
        ), // InputDecoration  
        controller: _textController,  
      ), // TextField  
      SizedBox(height: 16,),  
      Row(...), // Row  
      Row(...) // Row  
    ], // <Widget>[]  
  ), // Column  
) // Container
```



```
Row(  
  mainAxisAlignment: MainAxisAlignment.center,  
  children: <Widget>[  
    RaisedButton(...), // RaisedButton  
    SizedBox(width: 16,),  
    RaisedButton(...), // RaisedButton  
    SizedBox(width: 16,),  
    RaisedButton(...), // RaisedButton  
  ], // <Widget>[]  
) // Row
```

Shared Preference - Exemplo

```
Row(  
  mainAxisAlignment: MainAxisAlignment.center,  
  children: <Widget>[  
    RaisedButton(  
      child: Text("Salvar"),  
      onPressed: () {  
        print("Clicou no salvar");  
      }  
    ), // RaisedButton  
    SizedBox(width: 16,),  
    RaisedButton(...), // RaisedButton  
    SizedBox(width: 16,),  
    RaisedButton(...), // RaisedButton  
  ], // <Widget>[]  
) , // Row
```



```
Console  
I/flutter (18017): Clicou no salvar  
I/flutter (18017): Clicou no Recuperar  
I/flutter (18017): Clicou no Remover  
I/flutter (18017): Clicou no salvar
```

Shared Preference - Exemplo

Método `_salvarDados()`

```
_salvarDados() async{  
  String valorDigitado = _textController.text;  
  final prefs = await SharedPreferences.getInstance();  
  await prefs.setString("nome", valorDigitado);  
  print("Operação salvar: $valorDigitado");  
}
```

async: o método será assíncrono, ou seja, ao salvar os dados não necessariamente serão salvos de forma instantânea. Esse processo será executado de forma paralela

```
RaisedButton(  
  child: Text("Salvar"),  
  onPressed: _salvarDados,  
), // RaisedButton
```

Para recuperar a instância do Shared Preference também pode demorar um pouco, por isso, vamos usar **await** (de forma assíncrona)
Await: "Aguarde", usada esperar que a função seja executada e termine

Shared Preference - Exemplo

Método `_salvarDados()`

```
prefs.setString(key, value)
```

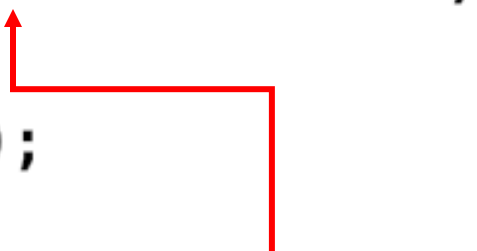
Key: será a chave que iremos utilizar tanto para salvar o dado, quanto para recuperar, podemos usar qualquer nome

Value: valor que será salvo

Shared Preference - Exemplo

Método _recuperarDados()

```
_recuperarDados() async{  
  final prefs = await SharedPreferences.getInstance();  
  setState(() {  
    _textoSalvo = prefs.getString("nome") ?? "Sem valor";  
  });  
  print("Operação recuperar: $_textoSalvo");  
}
```



```
RaisedButton(  
  child: Text("Recuperar"),  
  onPressed: _recuperarDados,  
), // RaisedButton
```

Caso o getString
seja nulo, o
_textoSalvo receberá
a string "Sem Valor"

Shared Preference - Exemplo

Método _recuperarDados()

```
setState(() {  
  _textoSalvo = prefs.getString("nome") ?? "Sem valor";  
});
```

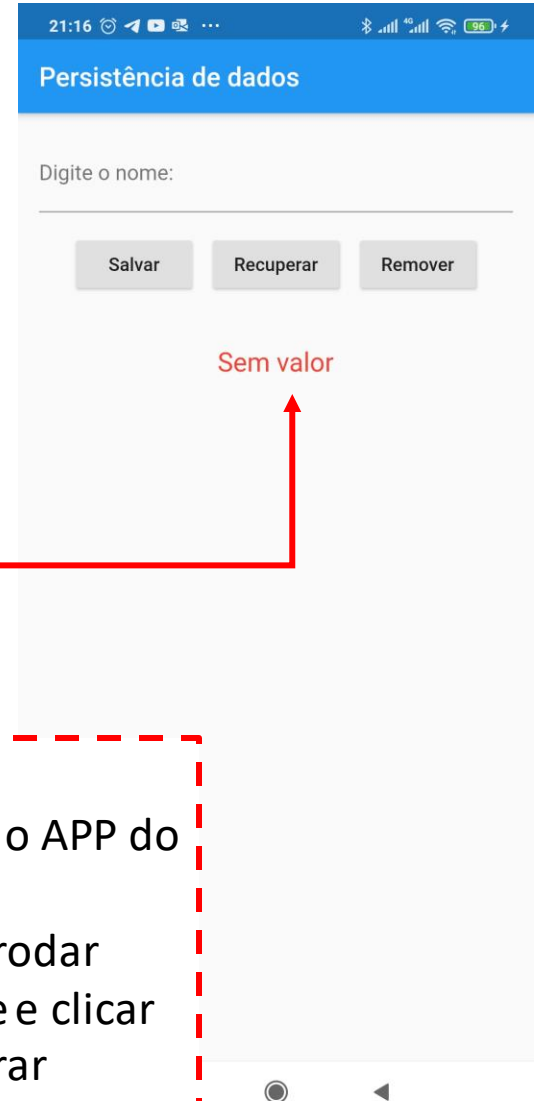
1 - Operador ??

`expressao_1 ?? expressao_2`

`x = y ?? z`

- Avalia `expressao_1` e caso for `NULL` atribui a `expressao_2`
- Atribui `z` a `x` se `y` for `NULL`

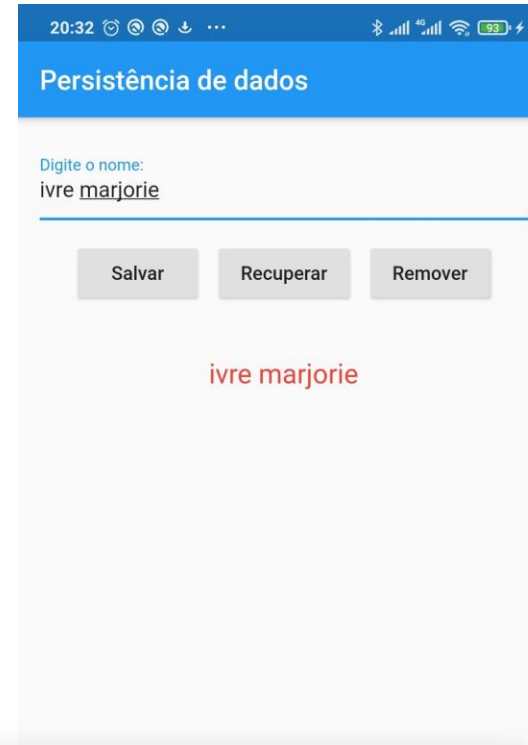
Fonte: <http://www.macoratti.net/>



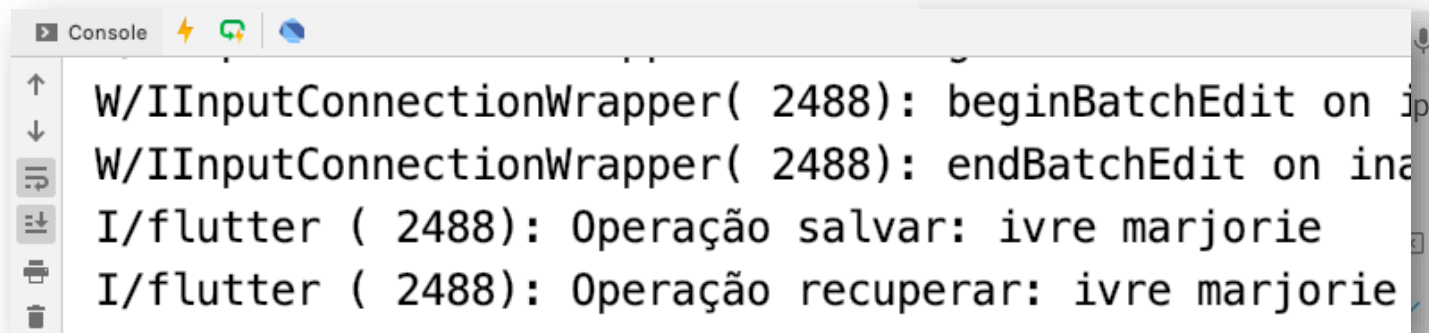
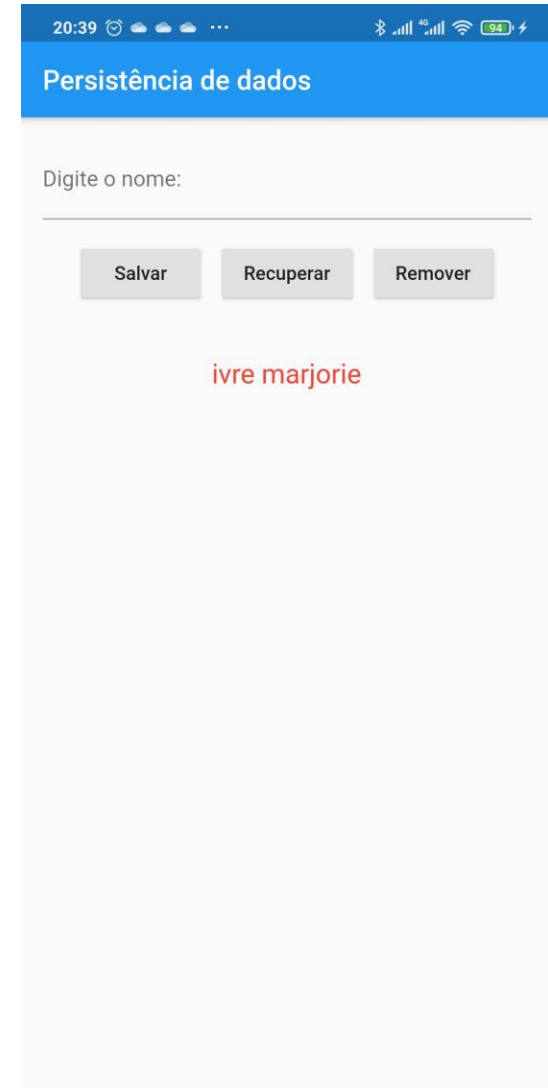
Shared Preference - Exemplo

- Para testar o método para recuperar dados, feche o APP (no celular ou no emulador), em seguida, abra novamente e clique em Recuperar como mostrado na Tela 2.

Tela 1



Tela 2



Shared Preference - Exemplo

Método `_removerDados()`

```
_removerDados() async{  
  final prefs = await SharedPreferences.getInstance();  
  prefs.remove("nome");  
  print("Operação remover");  
}
```

```
RaisedButton(  
  child: Text("Remover"),  
  onPressed: _removerDados,  
), // RaisedButton
```

remove(key): passando o
nome da chave que será
removida

Referências Bibliográficas

- Curso da Udemy – Flutter Essencial do professor Ricardo Lecheta.
- Curso da Udemy - Desenvolvimento Android e IOS com Flutter 2020 – Crie 15 Apps do professor Jamilton Damasceno.
- Site para Pacotes - <http://pub.dartlang.org/>
 - https://pub.dev/packages/shared_preferences
- Site Flutter – flutter.dev
 - <https://flutter.dev/docs/get-started/flutter-for/android-devs>