DCC917A – TÓPICOS ESPECIAIS III: DESENVOLVIMENTO DE APLICATIVOS MÓVEIS

AULA 13

Carlos Bruno Oliveira Lopes

Engenheiro de Computação

Mestre em Ciência da Computação

Flutter
(Manipulação de Dados – SharedPreferences)

SharedPreferences

- Usado para armazenamento e leitura de dados persistentes.
  - Em geral, usado para armazenar e ler dados de perfil de preferência do usuário.
- Ele trabalha com armazenamento assíncrono.
- E seu armazenamento é feito em pares

  (usa a estrutura `Map<String, dynamic>`)
  - **`String`**: key
  - **`dynamic`**: dados

Flutter
(Manipulação de Dados – SharedPreferences)

SharedPreferences

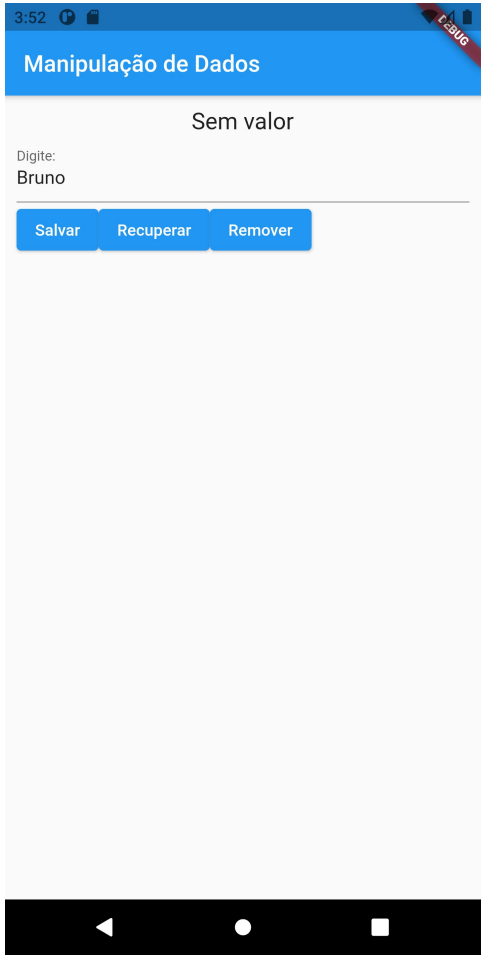
- Instalação e configuração das dependência para uso do Youtube em App:
  - Shared Preferences
  - Acesse o site: https://pub.dev/
    - Digite shared preferences no buscador e confirme a busca. Selecione Flutter e clique em shared_preferences;
    - Em seguida, selecione “Installing”, procure “depencencies:” e copie:

```
shared_preferences: ^2.0.7
```

    - Finalize a configuração do http acessando, o arquivo “pubspec.yaml” no seu projeto e adicionando a dependência.

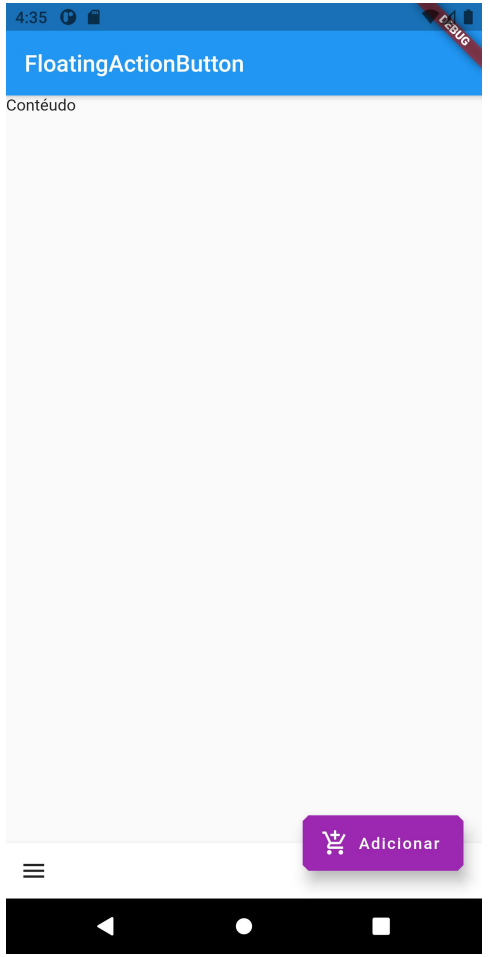
Flutter
(Manipulação de Dados – SharedPreferences)

```
_salvar() async {
  String valorDigitado = _entrada.text;
  final prefs = await SharedPreferences.getInstance();
  prefs.setString(key, valorDigitado);
  print("operação (salvar): $valorDigitado");
}

_recuperar() async{
  final prefs = await SharedPreferences.getInstance();
  setState(() {
    _dados = prefs.getString(key)?? "Sem valor";
  });
  print("operação (recuperar): $_dados");
}

_remover() async{
  final prefs = await SharedPreferences.getInstance();
  prefs.remove(key);
  print("operação (remover)");
}
```



Flutter (FloatingActionButton)

```
floatingActionButtonLocation: FloatingActionButtonLocation.endDocked,
  // floatingActionButton: FloatingActionButton(
  //   child: Icon(Icons.add),
  //   // backgroundColor: Colors.purple,
  //   // foregroundColor: Colors.white,
  //   elevation: 10,
  //   // mini: true,
  //   onPressed: (){},
  // ),
  floatingActionButton: FloatingActionButton.extended(
    icon: Icon(Icons.add_shopping_cart),
    label: Text("Adicionar"),
    backgroundColor: Colors.purple,
    foregroundColor: Colors.white,
    elevation: 10,
    shape: BeveledRectangleBorder(
      borderRadius: BorderRadius.circular(5),
      ),
    // mini: true,
    onPressed: (),
  ),
```



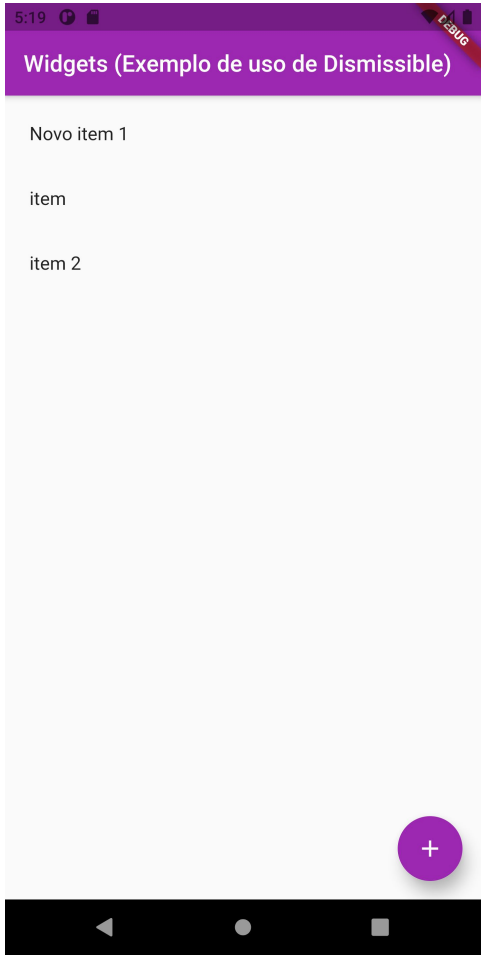
Flutter (Dismissible)

Dismissible

- Widget que permite que permite ser descartado arrastando-o em uma direção indicada;
  - Ela faz com que widget filha deslize para fora da tela
- Parâmetros requiridos:
  - `Key key`: usado como identificador para Widgets, Elements e SemanticNodes.
  - `Widget child`: usado para inserir um filho na widget Dismissible.

Flutter (Dismissible)

```
Dismissible(
  background: Container(
    color: Colors.green,
    padding: EdgeInsets.all(16),
    child: Row(
      children: [Icon(Icons.edit, color: Colors.white,),],
    ),
  ),
  secondaryBackground: Container(color: Colors.red,
   ...
  ),
  // direction: DismissDirection.vertical,
  onDismissed: (direction) {
    if(direction == DismissDirection.startToEnd){
      print("direção: startToEnd");
    } else { print("direção: endToStart");}
      setState(() { _listaString.removeAt(index);});
  },
  key: Key(_listaString[index]),
    child: ListTile(title: Text(_listaString[index]),
  ),
);
```



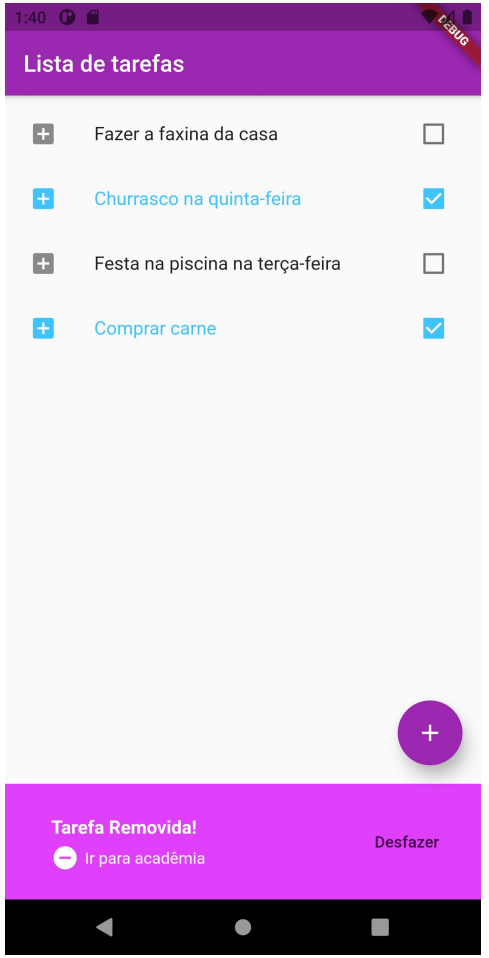
Flutter (SnackBar)

SnackBar

- Uma widget “leve” que exibe uma mensagem com um opção de ação que mostrada brevemente na parte inferior da tela.

Flutter (SnackBar)

```
snackbar = SnackBar(
  backgroundColor: Colors.purpleAccent,
  duration: Duration(seconds: 5),
  content: ListTile(
  title: Text(...),
  subtitle: Container(...),
  action: SnackBarAction(
    textColor: Color(0xFF3F0044),
    label: "Desfazer",
    onPressed: () {
      setState(() {
        _listaDeTarefas.insert(index, _tarefaRemovida);
        _atualizarDados();
      });
      print("Ultima ação foi desfeita!");
    },
  ),
);
```



Flutter
(Manipulação de Dados – path_provider)

Path Provider

- Plugin Flutter para obter locais comumente usados em sistemas de arquivos anfitriã, como diretórios de arquivo temporários e dados de Apps.
  - Suporte para Android, IOS, Linux, macOS e Windows.

Flutter
(Manipulação de Dados – path_provider)

Path Provider

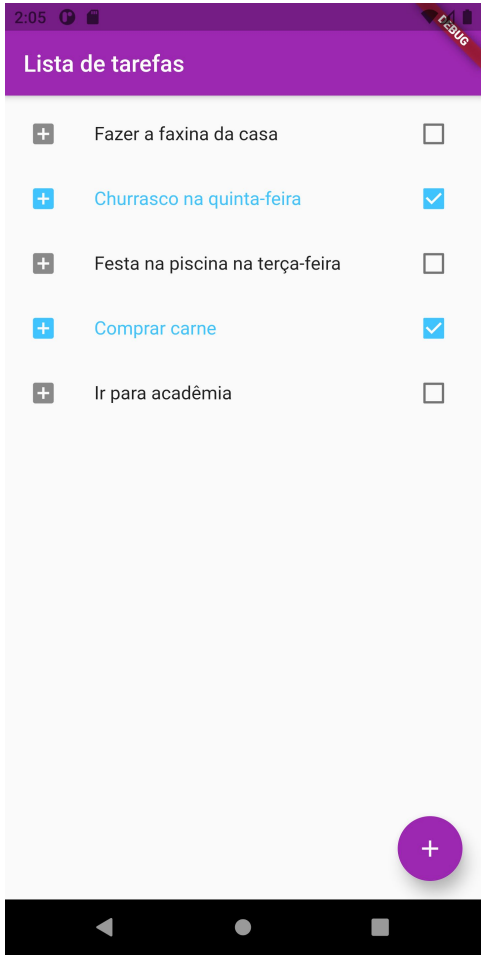
- Instalação e configuração das dependência para uso do Youtube em App:
  - Shared Preferences
  - Acesse o site: https://pub.dev/
    - Digite path provider no buscador e confirme a busca. Selecione Flutter e clique em path_provider;
    - Em seguida, selecione “Installing”, procure “depencencies:” e copie:

```
path_provider: ^2.0.3
```

    - Finalize a configuração do http acessando, o arquivo “pubspec.yaml” no seu projeto e adicionando a dependência.

Flutter [App – Lista de Tarefas] (Manipulação de Dados – path_provider)

```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.purple,
      title: Text("Lista de tarefas"),),
    body: ListView.builder(
      padding: EdgeInsets.all(5),
      itemCount: _listaDeTarefas.length,
      itemBuilder: _builderItem,),
    floatingActionButton: FloatingActionButton(
      child: Icon(Icons.add),
      backgroundColor: Colors.purple,
      foregroundColor: Colors.white,
      elevation: 10,
      onPressed: () { showDialog(context: context,
          builder: (context) => _builderDialogAddTask());},
    ),
  );
}
```



Flutter (Exercícios)

1. Adicionando funcionalidade na App Lista de Tarefas:
   - Torne o ícone da lista um botão de ação.
     - Quando clicar nesse botão deve abrir uma tela editável para inserir informações extras sobre a atividade.
   - Adicione a ação onTap sobre o texto da lista.
     - Quando clicar nesse botão deve-se exibir as informações complementares.