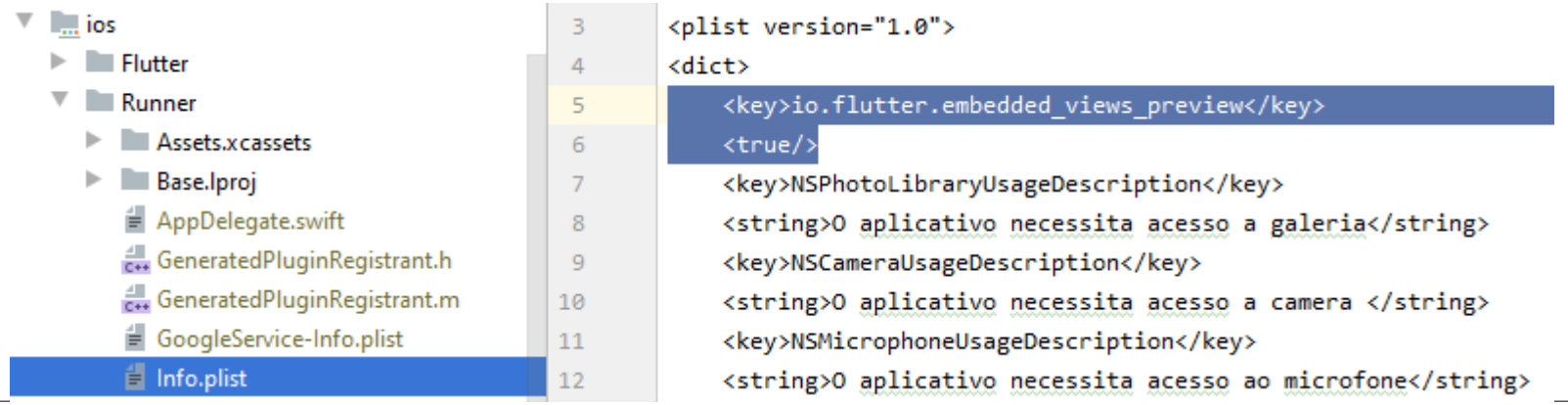
WebView e GridView

Definição

- Uma webview é um widget responsável pela exibição de páginas web dentro do aplicativo.
- Para utilizá-lo precisaremos do plugin webview flutter que pode ser obtido em:

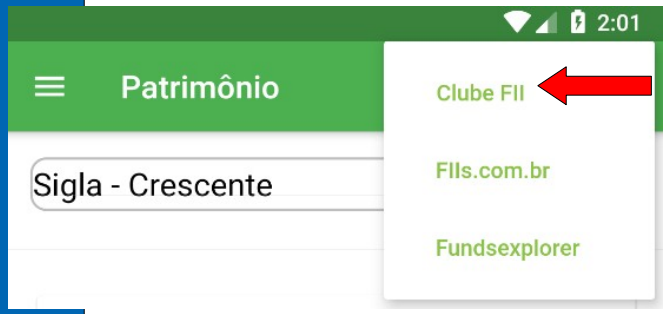
  https://pub.dev/packages/webview_flutter

- No iOS será necessário permitir visualizações incorporadas (já fizemos isso ao usar Mapas), o que significa fazer uma pequena alteração no arquivo **Info.plist**:



TelaWebViewFundos

- No FIIs Plan utilizamos uma WebView para exibir alguns sites que tratam do assunto de fundos imobiliários.



TelaWebViewFundos

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

class TelaWebViewFundos extends StatefulWidget {
  String url;

  TelaWebViewFundos(this.url);

  @override
  _TelaWebViewFundosState createState() => _TelaWebViewFundosState();
}

class _TelaWebViewFundosState extends State<TelaWebViewFundos> {
  var _indice_pilha = 1;
  WebViewController _controller;
```

- Aqui vemos a importação do arquivo **webview_flutter.dart**. Essa biblioteca é fundamental para o funcionamento do **WebView**.
- Essa tela recebe a **url** como parâmetro pois utilizamos **TelaWebViewFundos** para todos os sites de fundos imobiliários.
- **_indice_pilha** é uma variável para mostrar o **CircularProgressIndicator** até o site ser carregado.
- **_controller** é um WebViewController. Um WebViewController pode ser utilizado, entre outras coisas, para fazer o **reload** da página.

TelaWebViewFundos

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      actions: <Widget>[
        IconButton(
          icon: Icon(Icons.refresh),
          onPressed: () {
            _controller.reload();
          },
        ), // IconButton
      ], // <Widget>[]
    ), // AppBar
    body: _body(),
  ); // Scaffold
}
```

- Aqui vemos o botão de reload da TelaWebViewFundos.
- O **_controller** é responsável por dar o comportamento de recarregamento do site.
- Em **_body()** obteremos **_controller** e faremos o gerenciamento entre o **CircularProgressIndicator** e a **WebView**.

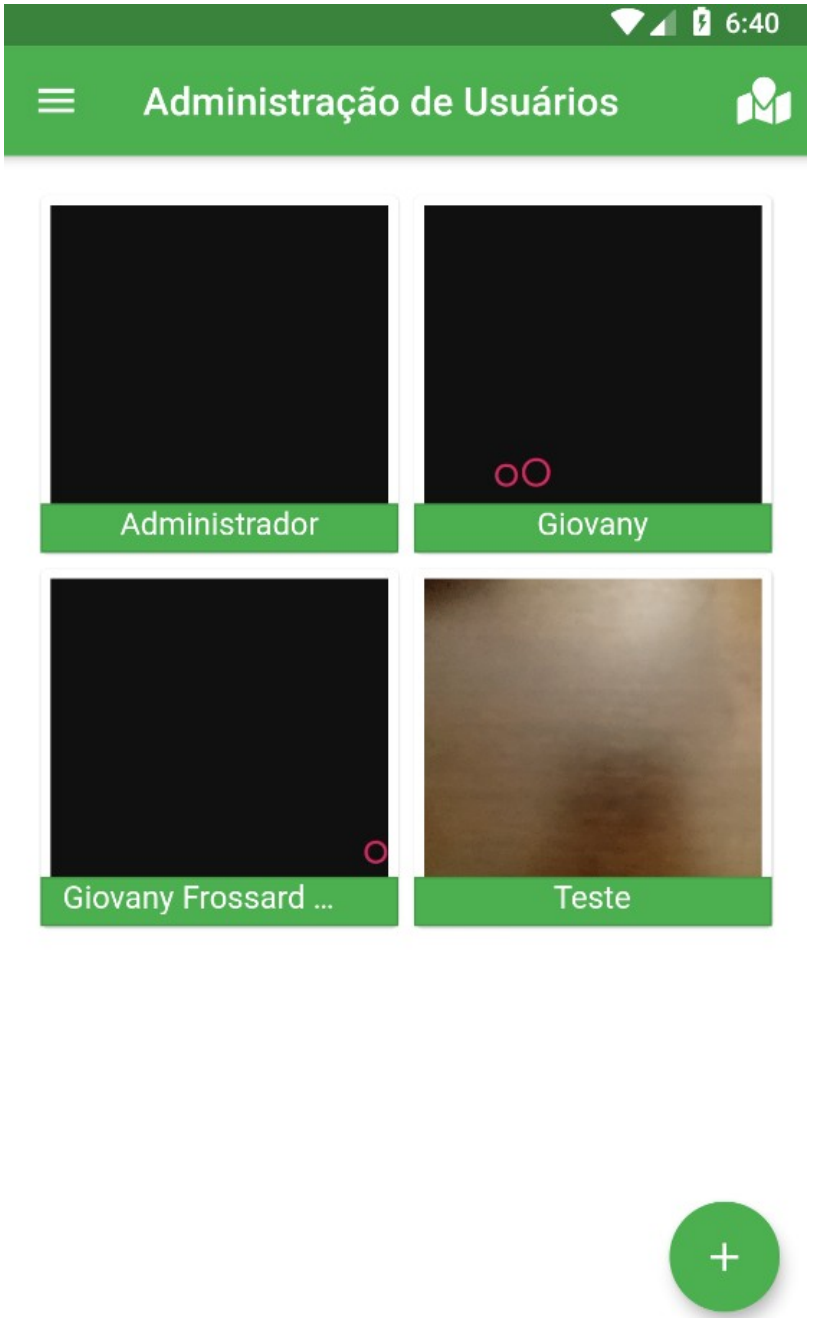
TelaWebViewFundos

- **IndexedStack** é uma **Stack** que troca quem está em vísivel através do parâmetro **index**.
- Inicialmente **_indice_pilha** é 1 (quadro verde), dessa forma o que aparece é o **CircularProgressIndicator**.
- **initialUrl** contém a o endereço para o qual a WebView deve ir.
- Em **onWebViewCreated** obtemos o **controller** para poder fazer o reload.
- Em **onPageFinished** (terminou o carregamento da página) mudamos **_indice_pilha** para 0 para mostrar a WebView (quadro vermelho).
- **javascriptMode** determina o que pode ser usado de JavaScript na página navegada (url).
- **navigationDelegate** controla se foi clicado em algum link da página e o que fazer quando isso acontecer. Nesse caso, foi permitida a navegação.

```
_body() {
  return IndexedStack(
    index: this._indice_pilha,
    children: <Widget>[
      Column(
        children: <Widget>[
          Expanded(
            child: WebView(
              initialUrl: this.widget.url,
              onWebViewCreated: (controller) {
                _controller = controller;
              },
              // Habilita o JavaScript
              javascriptMode: JavascriptMode.unrestricted,
              // Permite navegar na página
              navigationDelegate: (request){
                return NavigationDecision.navigate;
              },
              onPageFinished: (value){
                setState(() {
                  this._indice_pilha = 0;
                });
              },
            ), // WebView
          ), // Expanded
        ], // <Widget>[]
      ), // Column
      Container(
        color: Colors.white,
        child: Center(
          child: CircularProgressIndicator(),
        ), // Center
      ) // Container
    ], // <Widget>[]
  ); // IndexedStack
}
```

GridView

- O GridView é um widget para apresentar listagens de elementos (como um ListView) mas de forma bidimensional.
- No App FIIs Plan ele foi utilizado na TelaAdministracaoUsuario para listar os usuários cadastrados no aplicativo.

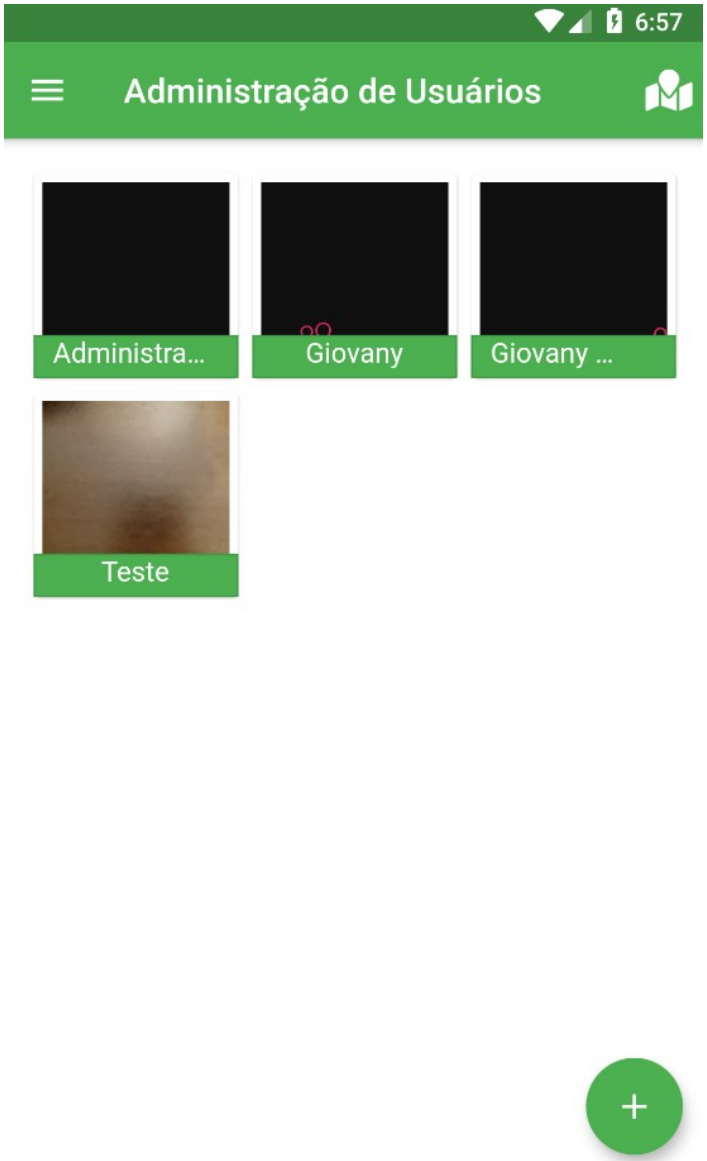


Método _gridView

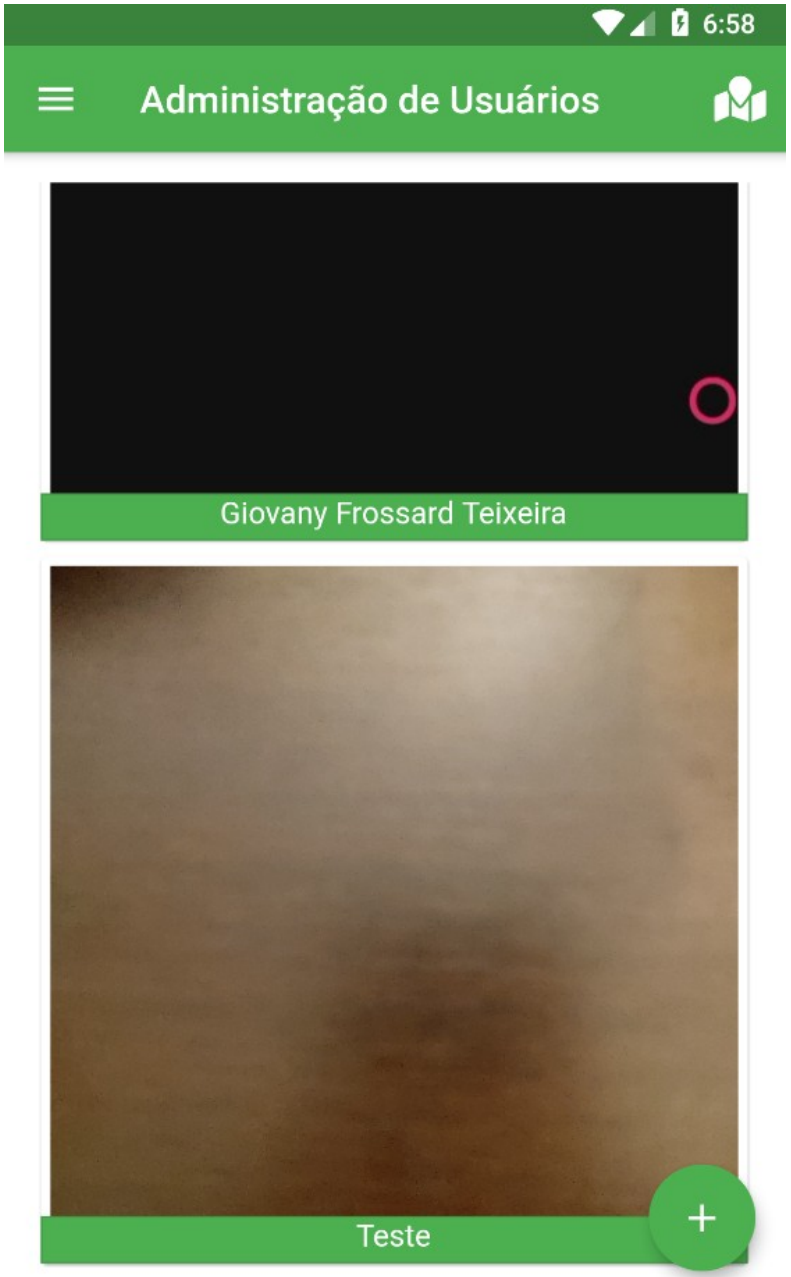
```
_gridView() {
  return Container(
    padding: EdgeInsets.all(16),
    child: GridView.builder(
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 2),
      itemCount: _controle.usuarios != null ? _controle.usuarios.length : 0,
      itemBuilder: (context, index) {
        Usuario usuario = _controle.usuarios[index];
        return GestureDetector(
          onTap: (){
            _showDialog(index, usuario);
          },
          child: CardUsuario(usuario),
        ); // GestureDetector
      },
    ), // GridView.builder
  ); // Container
}
```

- Na **TelaAdministracaoUsuario** o método responsável por "desenhar" a listagem de itens é o **_gridView**.
- No parâmetro **gridDelegate** definimos que utilizaremos 2 colunas para exibir os itens da listagem.
- **itemCount** determina a quantidade de itens da listagem (assim como no ListView.builder).
- **itemBuilder** retorna o widget que "desenha" a linha (da mesma forma que o ListView.builder).
- O uso do **GestureDetector** é para capturar o clique sobre o widget **CardUsuario** criado.
- O método setado em **onTap** mostra uma caixa de dialogo para definir o que será feito com o usuário (Alterar, Excluir, Cancelar) quando o **CardUsuario** for clicado.

```dart
_gridView() {
  return Container(
    padding: EdgeInsets.all(16),
    child: GridView.builder(
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 3),
      itemCount: _controle.usuarios != null ? _controle.usuarios.length : 0,
      itemBuilder: (context, index) {
        Usuario usuario = _controle.usuarios[index];
        return GestureDetector(
          onTap: (){
            _showDialog(index, usuario);
          },
          child: CardUsuario(usuario),
        ); // GestureDetector
      },
    ), // GridView.builder
  ); // Container
}
```



```dart
_gridView() {
  return Container(
    padding: EdgeInsets.all(16),
    child: GridView.builder(
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 1),
      itemCount: _controle.usuarios != null ? _controle.usuarios.length : 0,
      itemBuilder: (context, index) {
        Usuario usuario = _controle.usuarios[index];
        return GestureDetector(
          onTap: (){
            _showDialog(index, usuario);
          },
          child: CardUsuario(usuario),
        ); // GestureDetector
      },
    ), // GridView.builder
  ); // Container
}
```




```
_gridView() {
  return Container(
    padding: EdgeInsets.all(16),
    child: GridView.builder(
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 4),
      itemCount: _controle.usuarios != null ? _controle.usuarios.length : 0,
      itemBuilder: (context, index) {
        Usuario usuario = _controle.usuarios[index];
        return GestureDetector(
          onTap: (){
            _showDialog(index, usuario);
          },
          child: CardUsuario(usuario),
        ); // GestureDetector
      },
    ), // GridView.builder
  ); // Container
}
```

Dúvidas?