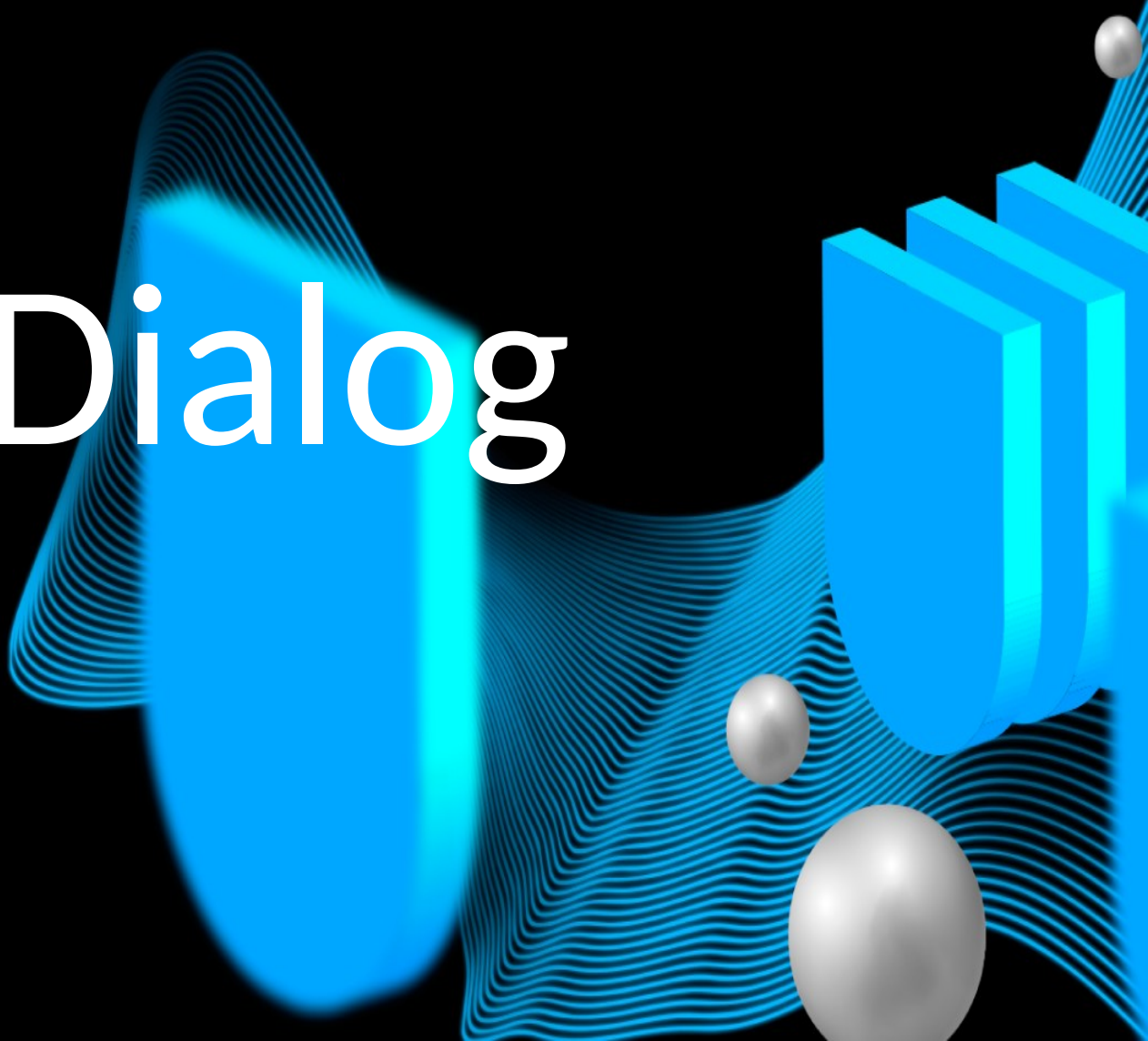

UNITAU
Universidade de Taubaté

Dialog



Dialog

Quando é necessário chamar a atenção do usuário exibindo uma informação ou solicitando uma escolha, podemos usar um **Dialog**.

A função **showDialog** é usada para exibir um widget como um popup. Usualmente é utilizado um widget da classe **Dialog** ou de uma classe derivada dessa.

Como a classe **Dialog** não tem opções sobre o conteúdo do diálogo, é mais comum o uso de widgets das classes **AlertDialog** ou **SimpleDialog**, derivadas da classe **Dialog**, que facilitam a exibição de diálogos simples e padronizados.

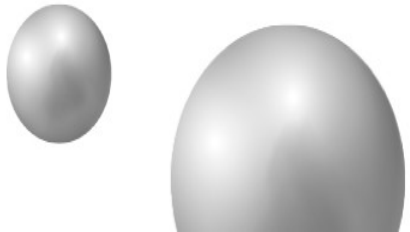
A classe **Dialog** é usada para criação de diálogos customizados. Também é possível utilizar a classe **showGeneralDialog** para customizar aspectos do popup.

Dialog

As opções mais comuns são diálogos com um único botão, apenas para exibir informações, com dois botões para confirmar uma operação, com mais de dois botões para escolher uma opção, com várias opções de escolha sem botões.

A função **showDialog** retorna um **Future**. Ao exibir um diálogo, deve-se determinar se o programa deve continuar o fluxo, sem aguardar a resposta do usuário ou se o programa irá aguardar a resposta do usuário.

A opção **barrierDismissible** da função **showDialog** determina se será possível fechar o diálogo tocando fora do diálogo (**true**) ou não (**false**).



Alerta

Crie uma nova aplicação Flutter com o nome **dialog_singlebutton**.

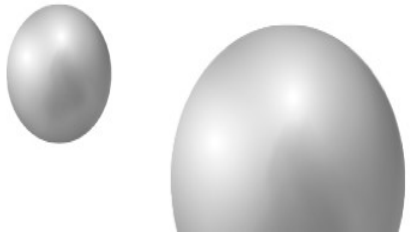
Altere o arquivo **main.dart**:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```

Alerta

Inclua a classe com a tela da aplicação:

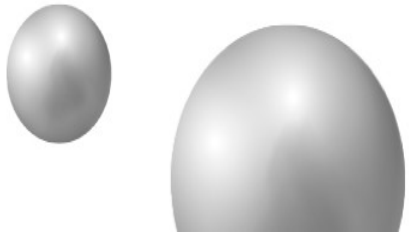
```
class Home extends StatefulWidget {

  @override
  _HomeState createState() => _HomeState();
}
```



Alerta

Inclua a classe para gerenciar o estado da tela da aplicação:

```
class _HomeState extends State<Home> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Dialog"),
      ),
      body: Center(
        child: RaisedButton(
          onPressed: () {
            _show(context);
          },
          child: const Text("Show Dialog"),
        ),
      ),
    );
  }
}
```



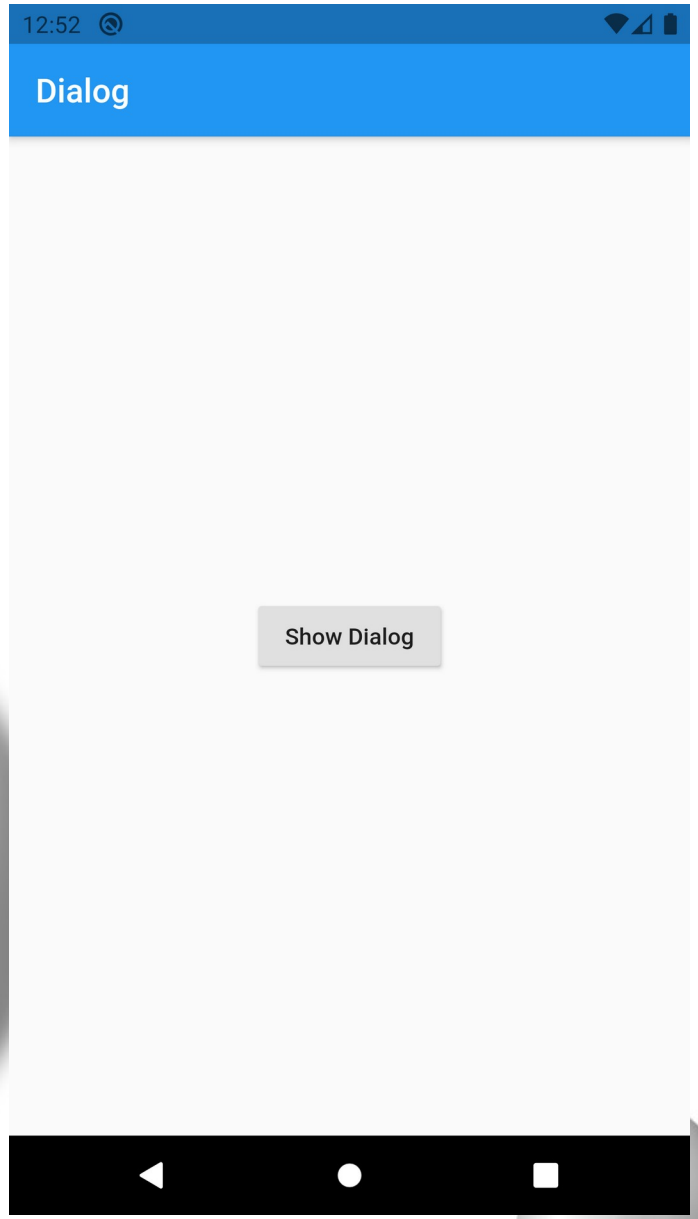
Alerta

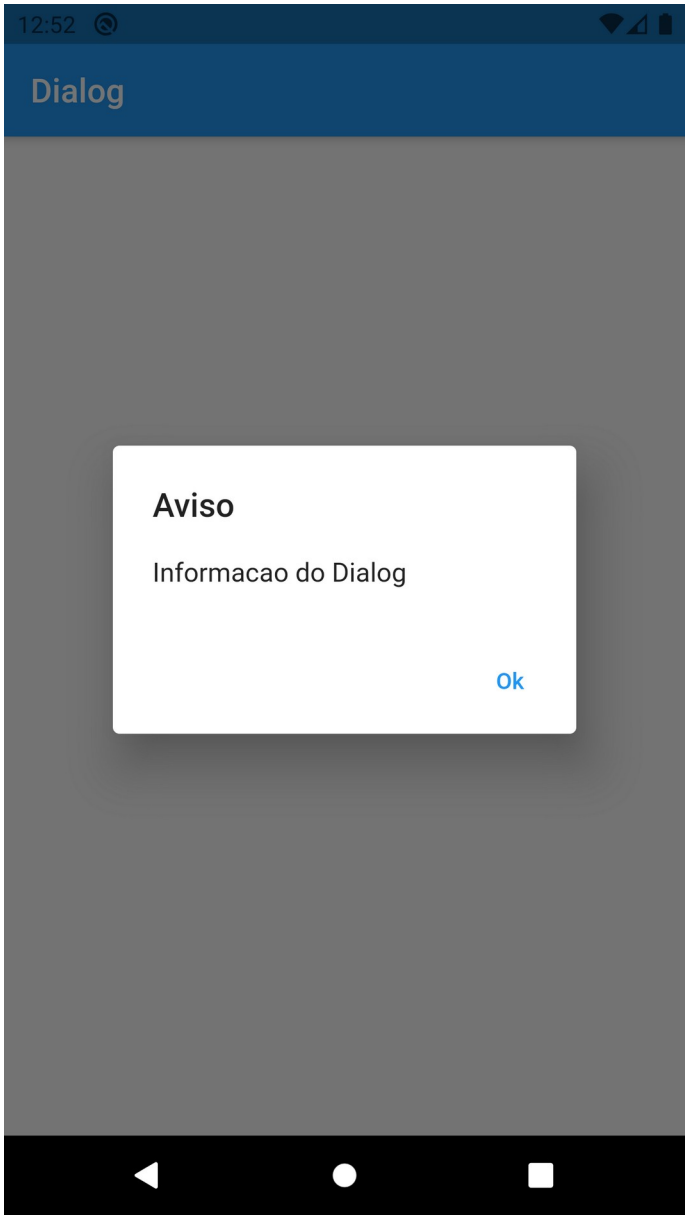
Veremos um diálogo que não interrompe o fluxo da aplicação e permite fechar a janela tocando fora do diálogo.

Inclua a função para exibir o diálogo:

```
Future<void> _show(BuildContext context) {
  return showDialog<void>(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text('Aviso'),
        content: const Text('Informacao do Dialog'),
        actions: <Widget>[
          FlatButton(
            child: Text('Ok'),
            onPressed: () => Navigator.of(context).pop(),
          ),
        ],
      );
    },
  );
}
```

Alerta


12:52
Dialog
Show Dialog


12:52
Dialog
Aviso
Informacao do Dialog
Ok

Confirmação

Crie uma nova aplicação Flutter com o nome **dialog_confirm**.

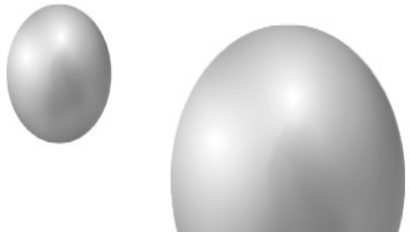
Altere o arquivo **main.dart**:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```

Confirmação

Inclua a classe com a tela da aplicação:

```
class Home extends StatefulWidget {

  @override
  _HomeState createState() => _HomeState();
}
```



Confirmação

Inclua a classe para gerenciar o estado da tela da aplicação:

```
class _HomeState extends State<Home> {
  bool _confirmado = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Dialog"),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            RaisedButton(
              onPressed: () async {
                _confirmado = await _show(context);
                setState(() {});
              },
              child: const Text("Show Dialog"),
            ),
```

Confirmação

```
            Divider(),
            Text(_confirmado?
              'Confirmado':'Nao confirmado',
              style: Theme.of(context).textTheme.display1,
            ),
          ],
        ),
      ),
    );
  }
}
```

Confirmação

Veremos um diálogo permite o usuário confirmar ou não uma operação, retornando true ou false. O diálogo interrompe o fluxo da aplicação até que o usuário seleciona uma resposta e não desaparece até que o usuário toque em um dos botões.

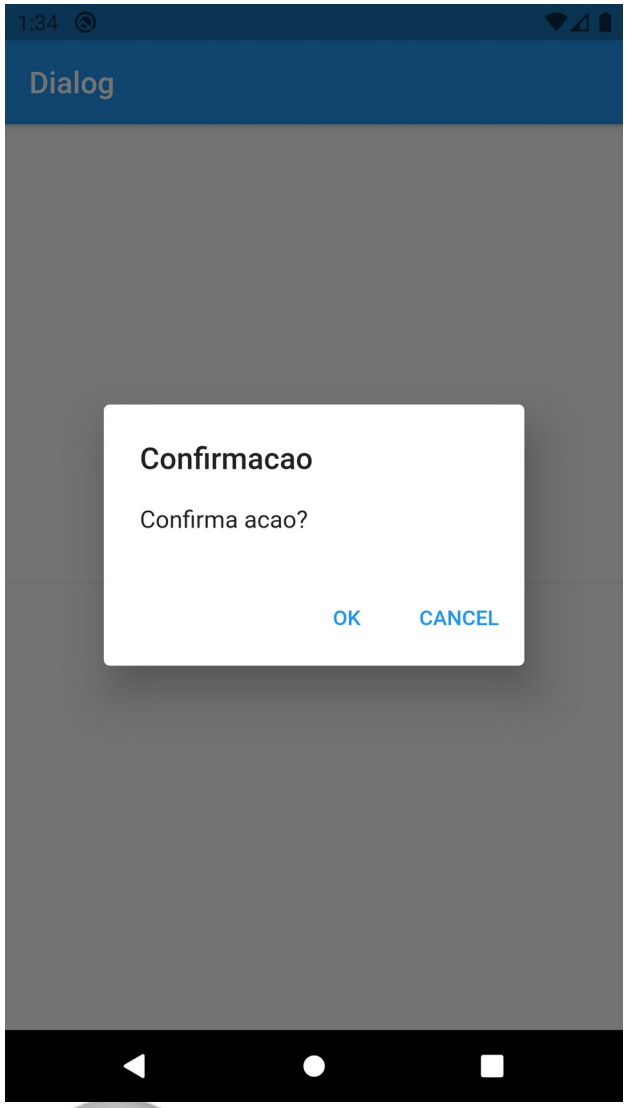
Inclua a função para exibir o diálogo:

```
Future<bool> _show(BuildContext context) async {
  return await showDialog<bool>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text('Confirmacao'),
        content: const Text('Confirma acao?'),
```

Confirmação

```
        actions: <Widget>[
          FlatButton(
            onPressed: () =>
Navigator.of(context).pop(true),
            child: const Text("OK")
          ),
          FlatButton(
            onPressed: () =>
Navigator.of(context).pop(false),
            child: const Text("CANCEL"),
          ),
        ],
      );
    },
  );
}
```

Confirmação



Opções

Crie uma nova aplicação Flutter com o nome **dialog_multibutton**.

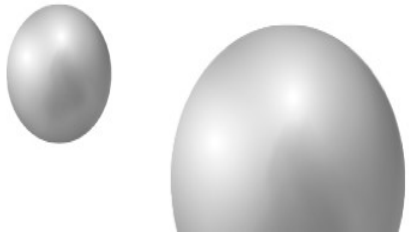
Altere o arquivo **main.dart**:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```

Opções

Inclua lista de opções e a classe com a tela da aplicação:

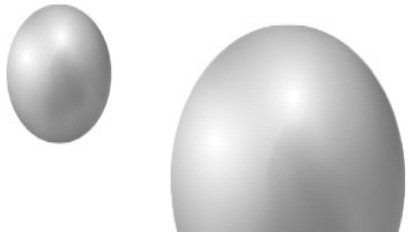
```
enum dialogAction { RESET, SAVE, CANCEL }

class Home extends StatefulWidget {

  @override
  _HomeState createState() => _HomeState();
}
```



Opções

Inclua a classe para gerenciar o estado da tela da aplicação:

```
class _HomeState extends State<Home> {
  var _action = dialogAction.RESET;
  String _actionText = 'Cancel';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Dialog"),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
```



Opções

```
        children: <Widget>[
          RaisedButton(
            onPressed: () async {
              _action = await _show(context);
              if (_action == dialogAction.RESET)
                _actionText = 'Reset';
              else if (_action == dialogAction.SAVE)
                _actionText = 'Save';
              else if (_action == dialogAction.CANCEL)
                _actionText = 'Cancel';
              setState(() {});
            },
            child: const Text("Show Dialog"),
          ),
          Divider(),
          Text(_actionText,
            style: Theme.of(context).textTheme.display1,),
        ],
      ),
    ),
  );
} }
```

Opções

Veremos um diálogo que permite selecionar entre diversas opções usando botões.

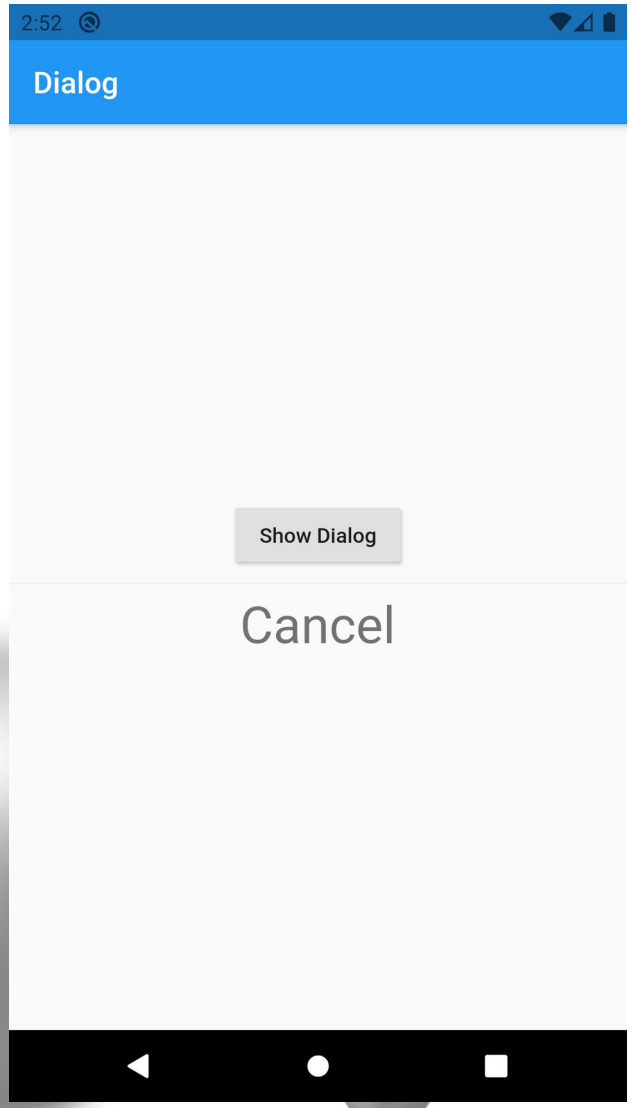
Inclua a função para exibir o diálogo:

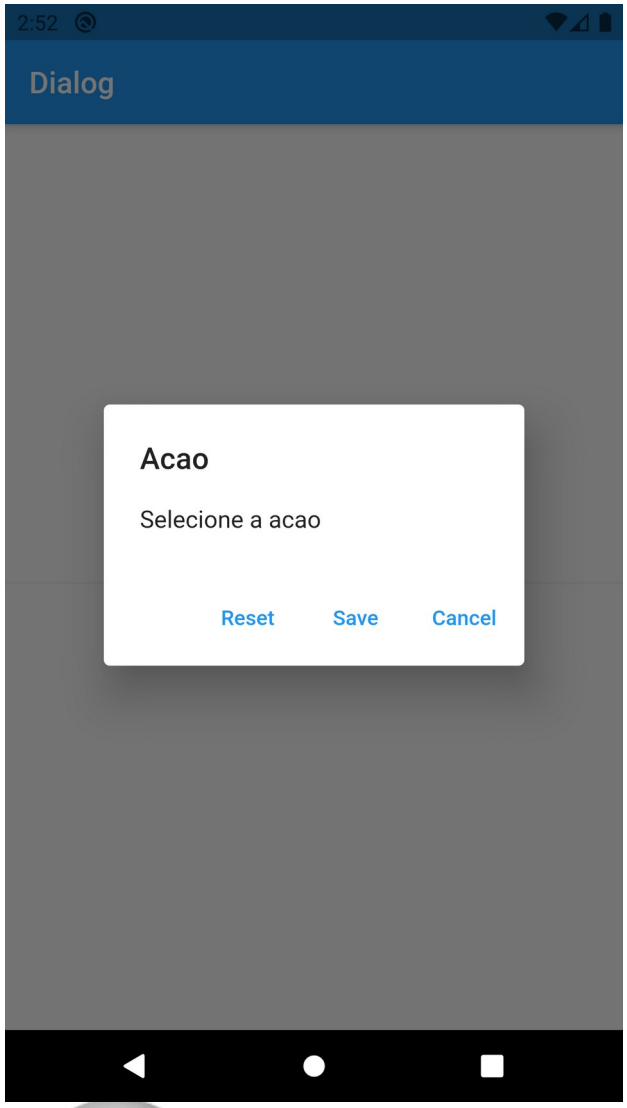
```
Future<dialogAction> _show(BuildContext context) async {
  return await showDialog<dialogAction>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text('Acao'),
        content: const Text('Selecione a acao'),
```

Opções

```
        actions: <Widget>[
          FlatButton(
            onPressed: () =>
Navigator.of(context).pop(dialogAction.RESET),
            child: const Text("Reset")
          ),
          FlatButton(
            onPressed: () =>
Navigator.of(context).pop(dialogAction.SAVE),
            child: const Text("Save"),
          ),
          FlatButton(
            onPressed: () =>
Navigator.of(context).pop(dialogAction.CANCEL),
            child: const Text("Cancel"),
          ),
        ],
      );
    },
  );
}
```

Opções


2:52
Dialog
Show Dialog
Cancel


2:52
Dialog
Acao
Selecione a acao
Reset
Save
Cancel


2:52
Dialog
Show Dialog
Save

SimpleDialog

Crie uma nova aplicação Flutter com o nome **dialog_simple**.

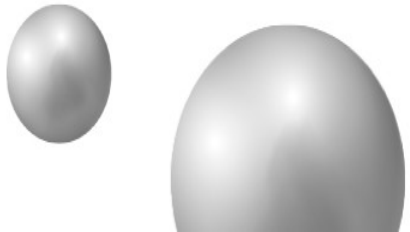
Altere o arquivo **main.dart**:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```

SimpleDialog

Inclua a lista de opções e a classe com a tela da aplicação:

```
enum dialogAction { RESET, SAVE, CANCEL }

class Home extends StatefulWidget {

  @override
  _HomeState createState() => _HomeState();
}
```



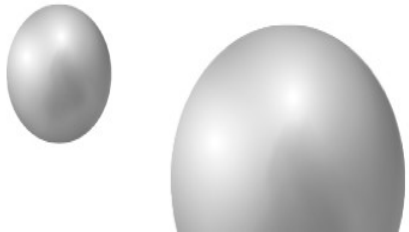
SimpleDialog

Inclua a classe para gerenciar o estado da tela da aplicação:

```
class _HomeState extends State<Home> {
  var _action = dialogAction.RESET;
  String _actionText = 'Cancel';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Dialog"),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
```

SimpleDialog

```
        children: <Widget>[
          RaisedButton(
            onPressed: () async {
              _action = await _show(context);
              if (_action == dialogAction.RESET)
                _actionText = 'Reset';
              else if (_action == dialogAction.SAVE)
                _actionText = 'Save';
              else if (_action == dialogAction.CANCEL)
                _actionText = 'Cancel';
              setState(() {});
            },
            child: const Text("Show Dialog"),
          ),
          Divider(),
          Text(_actionText,
            style: Theme.of(context).textTheme.display1,),
        ],
      ),
    ),
  );
} }
```

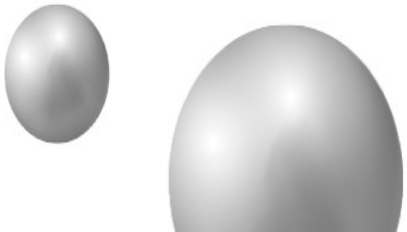


SimpleDialog

Veremos um diálogo que utiliza a classe **SimpleDialog** para permitir a seleção entre várias opções. Embora seja possível utilizar outros widgets, usualmente se utiliza **SimpleDialogOption** para representar as opções do diálogo. **SimpleDialog** não tem botões.

Inclua a função para exibir o diálogo:

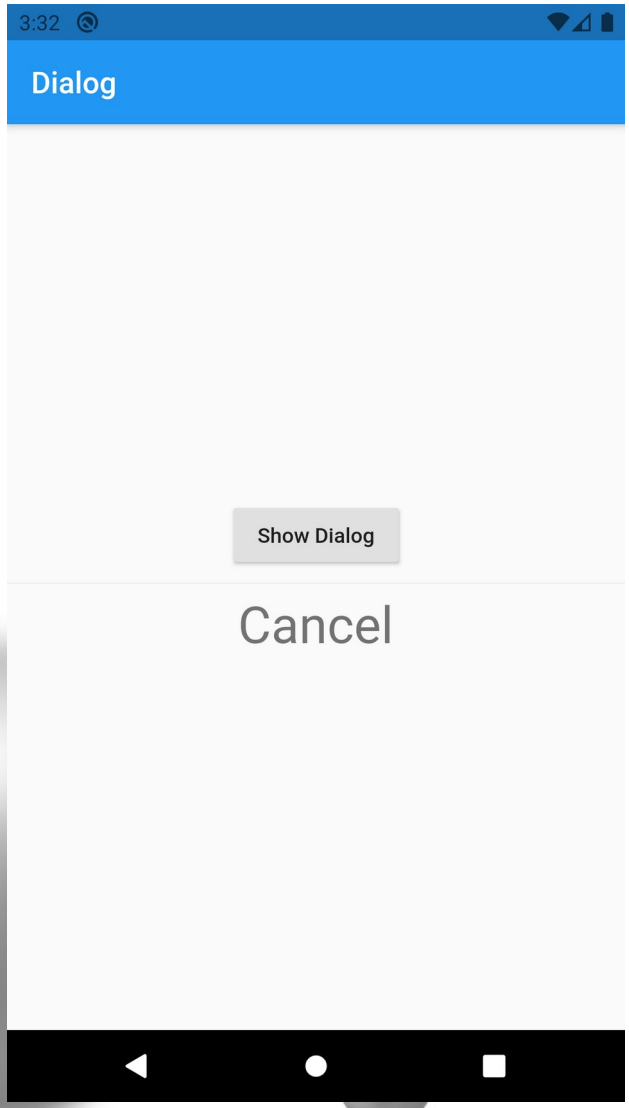
```
Future<dialogAction> _show(BuildContext context) async {
  return await showDialog<dialogAction>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return SimpleDialog(
        title: const Text('Selecione a acao'),
```

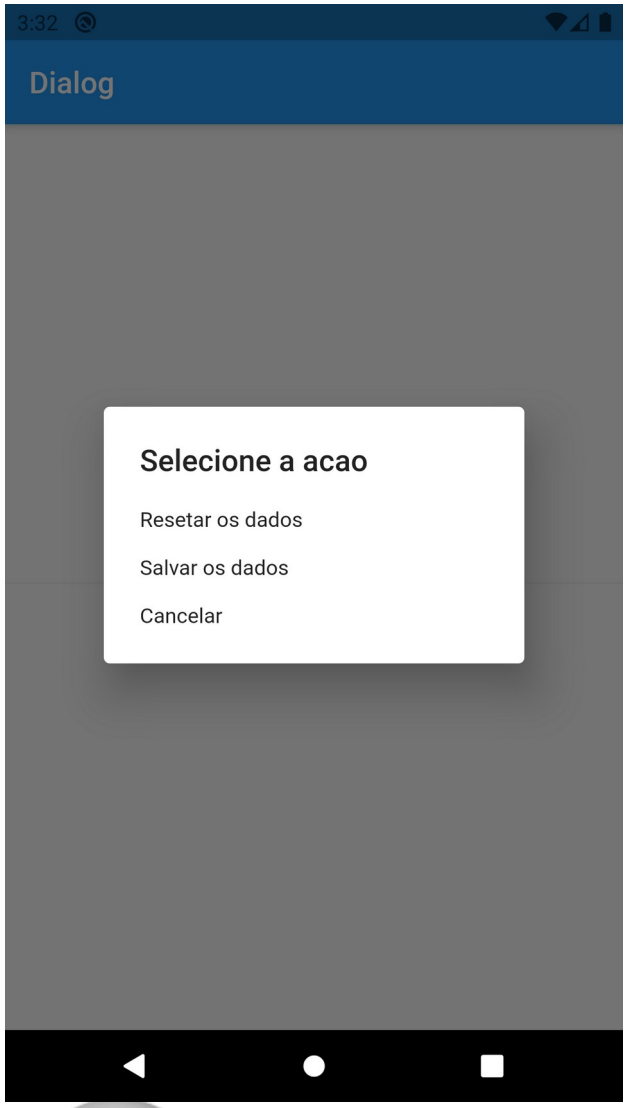


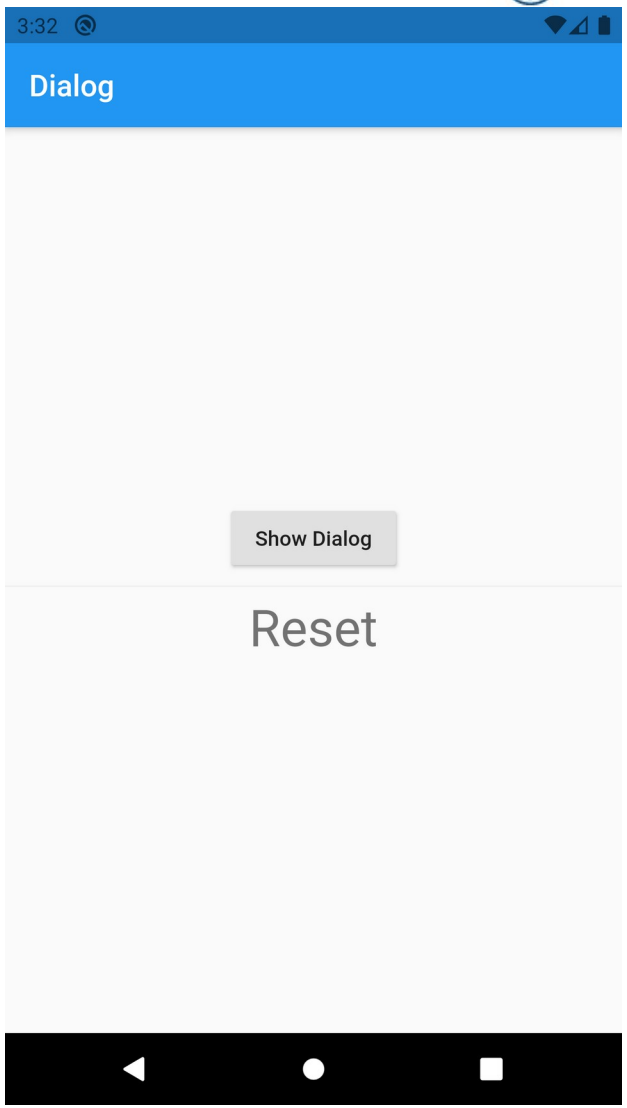
SimpleDialog

```
        children: <Widget>[
          SimpleDialogOption(
            onPressed: () => Navigator.pop(context,
dialogAction.RESET),
            child: const Text('Resetar os dados'),
          ),
          SimpleDialogOption(
            onPressed: () => Navigator.pop(context,
dialogAction.SAVE),
            child: const Text('Salvar os dados'),
          ),
          SimpleDialogOption(
            onPressed: () => Navigator.pop(context,
dialogAction.CANCEL),
            child: const Text('Cancelar'),
          ),
        ],
      );
    },
  );
}
```

SimpleDialog







Entrada de Dados

Crie uma nova aplicação Flutter com o nome **dialog_input**.

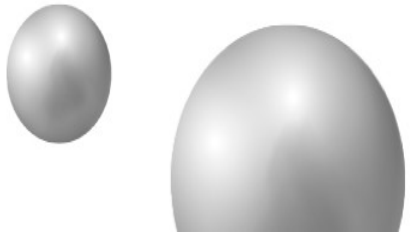
Altere o arquivo **main.dart**:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```

Entrada de Dados

Inclua a classe com a tela da aplicação:

```
class Home extends StatefulWidget {

  @override
  _HomeState createState() => _HomeState();
}
```



Entrada de Dados

Inclua a classe para gerenciar o estado da tela da aplicação:

```
class _HomeState extends State<Home> {
  String _input = "";

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Dialog"),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
```

Entrada de Dados

```
          children: <Widget>[
            RaisedButton(
              onPressed: () async {
                String _retorno = await _show(context);
                if ( _retorno != null ) {
                  _input = _retorno;
                  setState(() {});
                }
              },
              child: const Text("Show Dialog"),
            ),
            Divider(),
            Text(_input,
              style: Theme.of(context).textTheme.display1,
            ),
          ],
        ),
      ),
    );
  }
}
```

Entrada de Dados

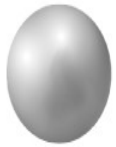
Veremos um diálogo para entrada de dados que permite que o usuário digite um texto.

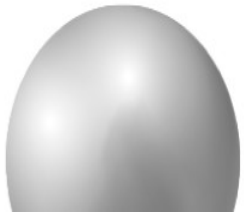
Inclua a função para exibir o diálogo:

```
Future<String> _show(BuildContext context) async {
  String _entrada;
  return await showDialog<String>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text('Entre com um texto'),
```

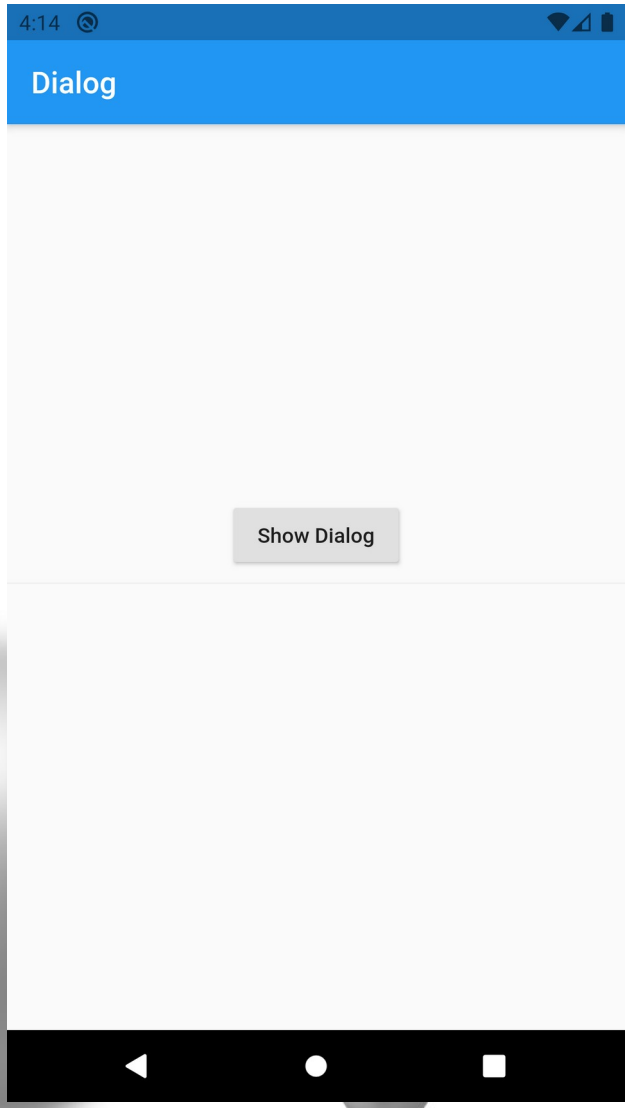
Entrada de Dados

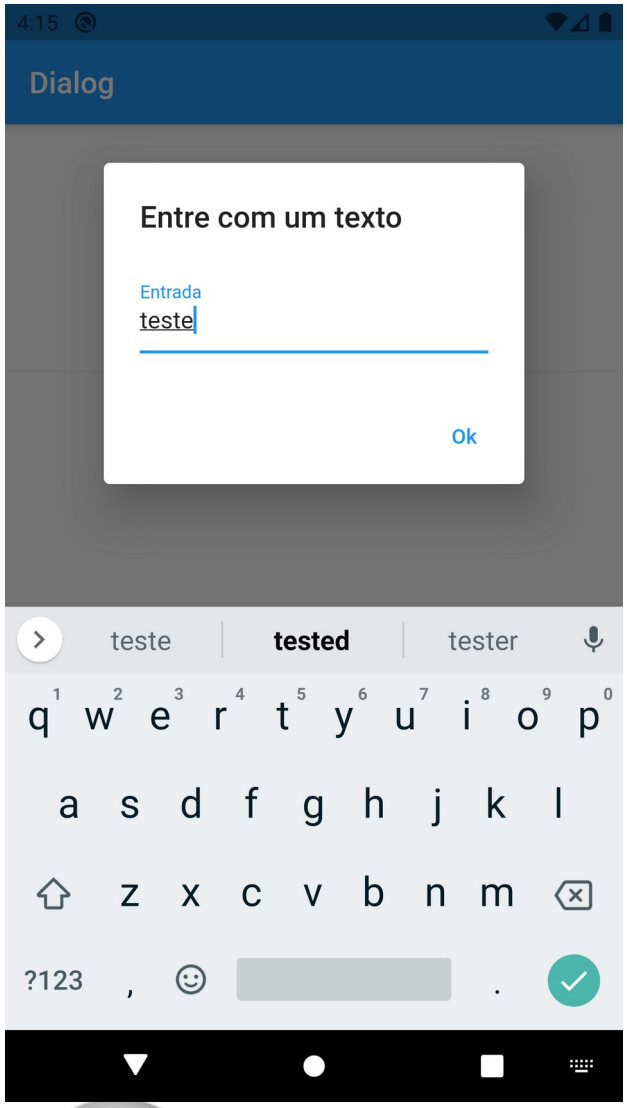
```
        content: Row(
          children: <Widget>[
            Expanded(
              child: TextField(autofocus: true,
                decoration: InputDecoration(
                  labelText: 'Entrada', ),
                onChanged: (value) => _entrada = value,
              )
            )
          ],
        ),
        actions: <Widget>[
          FlatButton(
            child: Text('Ok'),
            onPressed: () {
              Navigator.of(context).pop(_entrada); },
          ),
        ],
      );
    },
  );
}
```





Entrada de Dados





Dialog Customizado

Crie uma nova aplicação Flutter com o nome dialog_custom.

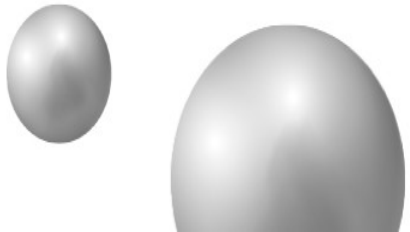
Altere o arquivo main.dart:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```

Dialog Customizado

Inclua a classe com a tela da aplicação:

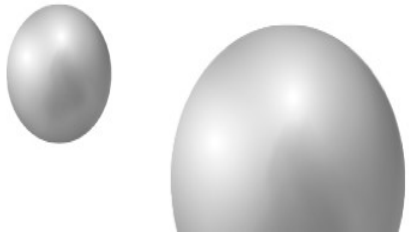
```
class Home extends StatefulWidget {

  @override
  _HomeState createState() => _HomeState();
}
```



Dialog Customizado

Inclua a classe para gerenciar o estado da tela da aplicação:

```
class _HomeState extends State<Home> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Dialog"),
      ),
      body: Center(
        child: RaisedButton(
          onPressed: () {
            _show(context);
          },
          child: const Text("Show Dialog"),
        ),
      ),
    );
  }
}
```



Dialog Customizado

Veremos um diálogo customizado com a classe Dialog. É necessário criar todos os elementos do diálogo.

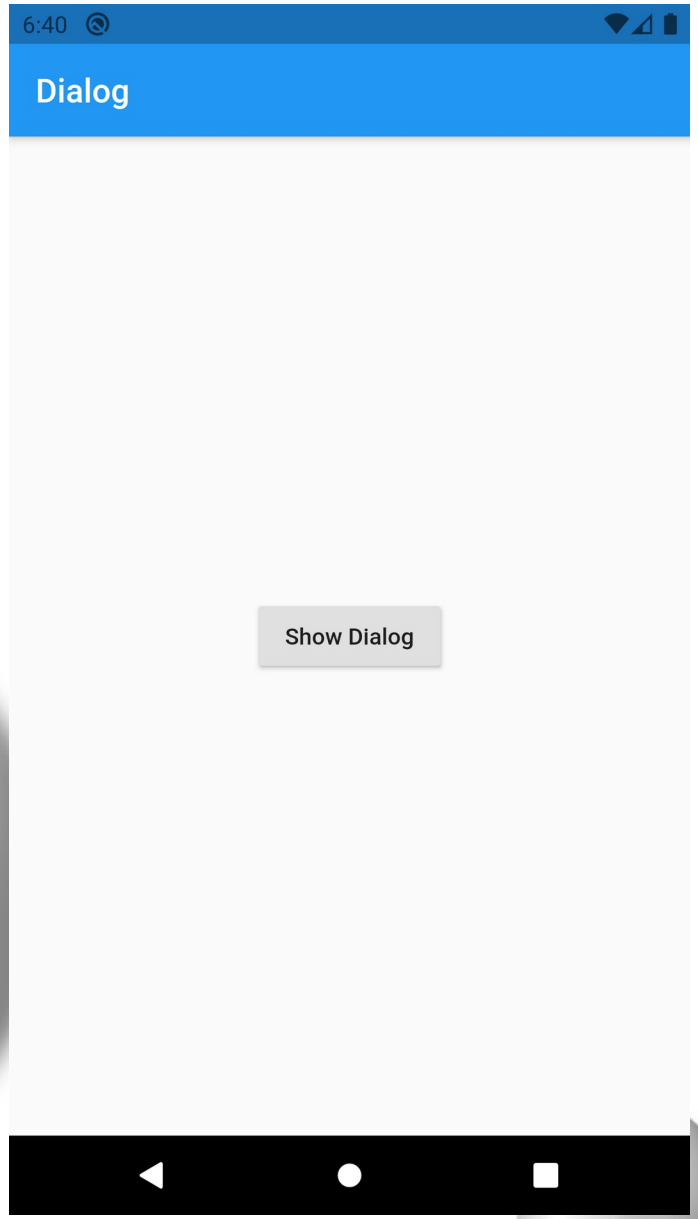
Inclua a função para exibir o diálogo:

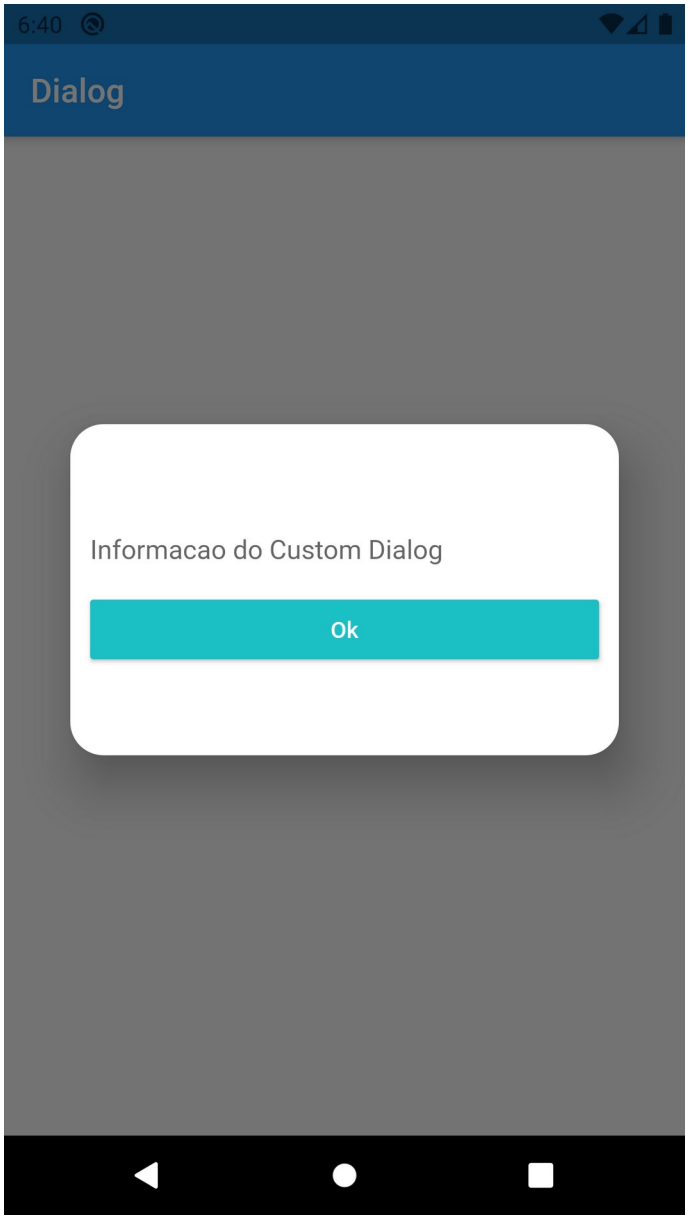
```
Future<void> _show(BuildContext context) {
  return showDialog(
    context: context,
    builder: (BuildContext context) {
      return Dialog(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(20.0),
        ),
        child: Container(height: 200,
          child: Padding(
            padding: const EdgeInsets.all(12.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              crossAxisAlignment: CrossAxisAlignment.start,
```

Dialog Customizado

```
                children: [
                  TextField(
                    decoration: InputDecoration(
                      border: InputBorder.none,
                      hintText: 'Informacao do Custom Dialog'
                    ), ),
                  SizedBox(width: 320.0,
                    child: RaisedButton(
                      onPressed: ()=>
Navigator.of(context).pop(),
                      child: Text("Ok",
                        style: TextStyle(color: Colors.white),
                      ),
                      color: const Color(0xFF1BC0C5),
                    ), )
                ],
              ),
            ),
          ),
        );
      },
    ); }
```

Dialog Customizado





Dialogs Úteis

Dada a frequente necessidade de exibir uma informação ou uma mensagem erro e de solicitar a confirmação de uma operação, criaremos diálogos para essas funções que serão utilizados nas demais aplicações.

Dialogs Úteis

Crie uma nova aplicação Flutter com o nome **dialog_utils**.

Altere o arquivo **main.dart**:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```

Dialogs Úteis

Crie o subdiretório lib/pages e o arquivo home.dart nesse subdiretório.

```
import 'package:flutter/material.dart';
import '../utils/confirm.dart';
import '../utils/showalert.dart';
import '../utils/showerror.dart';

class Home extends StatefulWidget {

  @override
  _HomeState createState() => _HomeState();
}
```

Dialogs Úteis

Inclua a classe para gerenciar o estado da tela da aplicação:

```
class _HomeState extends State<Home> {
  bool _confirmado = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Dialog"), ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            RaisedButton(
              onPressed: () async {
                await showAlert(context,
                    "Informacoes para o usuario"
                );
              },
              child: const Text("Alerta"),
            ),
```

Dialogs Úteis

```
Divider(),
RaisedButton(
  onPressed: () async {
    await showError(context,
        "Mensagem de erro"
    );
  },
  child: const Text("Erro"),
),
Divider(),
RaisedButton(
  onPressed: () async {
    _confirmado = await confirm(context,
      "Confirmar acao?"
    );
    setState(() {});
  },
  child: const Text("Confirmacao"),
),
Divider(),
```

Dialogs Úteis

```
            Text(_confirmado?
              'Confirmado':'Nao confirmado',
              style: Theme.of(context).textTheme.display1,
            ),
          ],
        ),
      ),
    );
  }
}
```

Dialogs Úteis

Crie o subdiretório lib/utils e o arquivo showalert.dart com o diálogo para exibir uma informação para o usuário nesse subdiretório.

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

void showAlert(BuildContext context, String _msg) async {
  return await showDialog<void>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(20.0),
        ),
```

Dialogs Úteis

```
        title: Row(
            children: [
              Icon(Icons.info,color:
Colors.yellow[600],size: 48.0,),
              Expanded(
                child: Text(
                  'Alerta',
                  style: TextStyle(fontWeight:
FontWeight.bold),
                  textAlign: TextAlign.center,
                ),
              ),
            ]
        ),
        content: Text(_msg),
```

Dialogs Úteis

```
        actions: <Widget>[
          FlatButton(
            child: Text('Ok'),
            onPressed: () => Navigator.of(context).pop(),
          ),
        ],
      );
    },
  );
}
```

Dialogs Úteis

Crie o arquivo **showerror.dart** com o diálogo para exibir uma mensagem de erro para o usuário no subdiretório **lib/utils**.

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

void showError(BuildContext context, String _msg) async {
  return await showDialog<void>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(20.0),
        ),
```

Dialogs Úteis

```
        title: Row(
            children: [
              Icon(Icons.error,color: Colors.red,size:
48.0,),
              Expanded(
                child: Text(
                  'Erro',
                  style: TextStyle(fontWeight:
FontWeight.bold),
                  textAlign: TextAlign.center,
                ),
              ),
            ]
        ),
        content: Text(_msg),
```

Dialogs Úteis

```
        actions: <Widget>[
          FlatButton(
            child: Text('Ok'),
            onPressed: () => Navigator.of(context).pop(),
          ),
        ],
      );
    },
  );
}
```

Dialogs Úteis

Crie o arquivo **confirm.dart** com o diálogo para solicitar a confirmação de uma operação para o usuário no subdiretório **lib/utils**.

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

Future<bool> confirm(BuildContext context, String _msg)
async {
  return await showDialog(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(20.0),
        ),
```

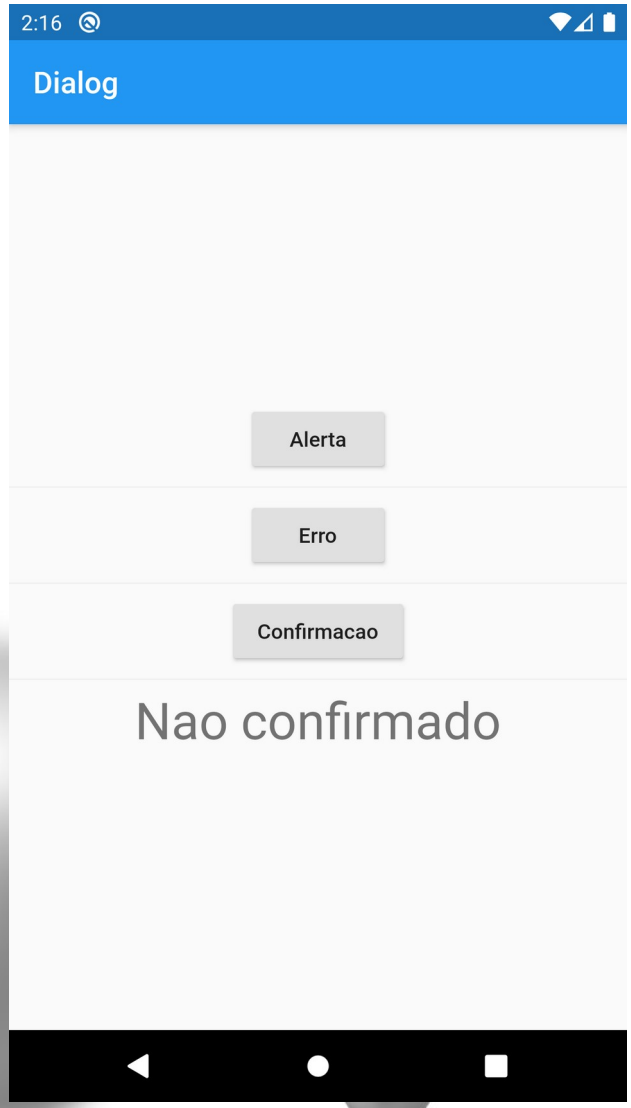
Dialogs Úteis

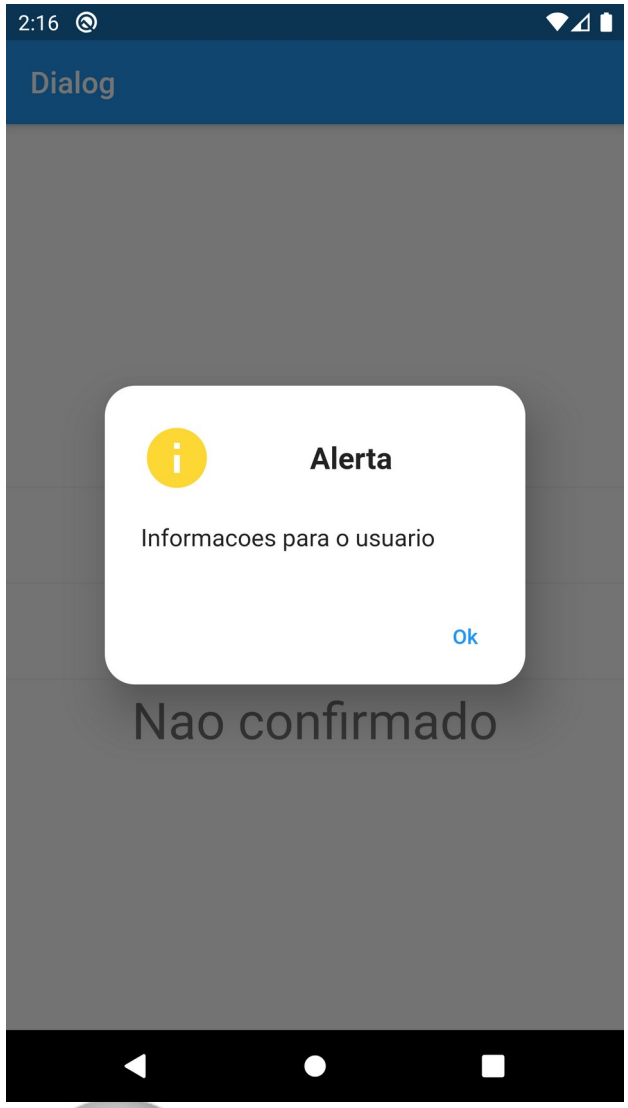
```
        title: Row(
          children: [
            Icon(Icons.help,color: Colors.blue,size: 48.0,),
            Expanded(
              child: Text('Confirmacao',
                style: TextStyle(fontWeight:
FontWeight.bold),
                textAlign: TextAlign.center,
              ),
            ),
          ]
        ),
        content: Text(_msg),
```

Dialogs Úteis

```
        actions: <Widget>[
          FlatButton(child: const Text("OK"),
            onPressed: () =>
Navigator.of(context).pop(true),
          ),
          FlatButton(child: const Text("CANCEL"),
            onPressed: () =>
Navigator.of(context).pop(false),
          ),
        ],
      );
    },
  );
}
```

Dialogs Úteis





Dialogs Úteis