Listview

É um widget que permite exibir uma lista de itens com rolagem vertical. É um dos widgets mais utilizados para construir interfaces de usuário que exibem listas de dados, como listas de mensagens, lista de contatos, lista de produtos, entre outros.

É altamente personalizável, permitindo que os desenvolvedores configurem o estilo, o layout e o comportamento dos itens da lista de acordo com as necessidades do aplicativo. É possível definir um widget personalizado para cada item da lista, permitindo que os desenvolvedores exibam informações mais complexas do que apenas texto.

Listview

Definindo elemento para serem visualizados no ListView de forma estática:

```
class FormsState extends State<Forms> {
  //define conteúdo de listView
  final List<String> items = ['Item 1', 'Item 2', 'Item 3'];
```

A variável Items (do tipo List) armazena as informações que serão renderizadas no componente ListView.

Listview

Build da Classe FormState que contém o ListView:

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Sample Input ListView'),
    ),
    body: Align(
      alignment: Alignment.center,
      child: Column(
        mainAxisSize: MainAxisSize.max,
        children: [
```

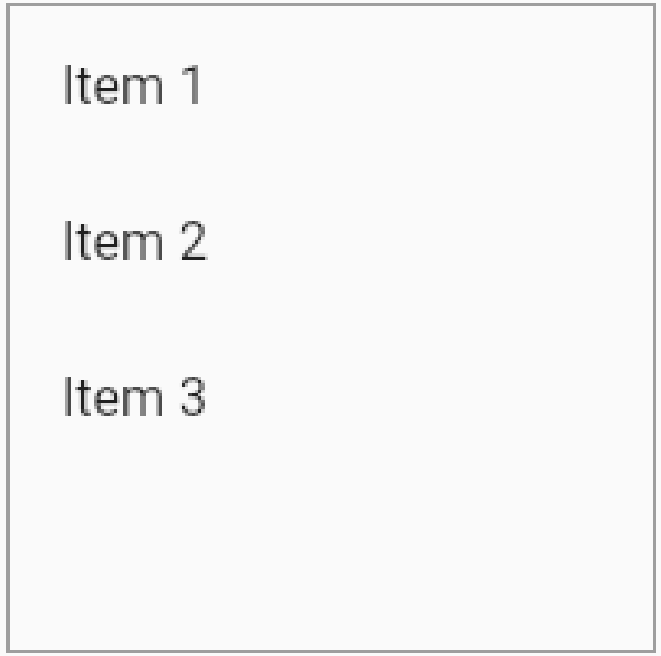
Children: Permite que os desenvolvedores especifiquem um ou mais widgets que devem ser renderizados dentro de outro widget pai.

"child" é um parâmetro comum usado em muitos widgets que aceitam apenas um único widget filho como conteúdo. Esse parâmetro permite que os desenvolvedores especifiquem o widget que deve ser renderizado dentro de outro widget pai. O parâmetro "child" é usado em widgets como Container, Card, FloatingActionButton, Icon, Image, Text, entre outros.

Listview

Composição do Layout do ListView

```
const Divider(),// linha divisória
Container(
  width: 200,// commprimento
  height: 200,// altura
  decoration: BoxDecoration(
    border: Border.all(// borda ao redor
      color: Colors.grey,// cor da borda
      width: 1.0,// espessura da linha
    ),
  ),
```



Container: Define uma área que irá conter o ListView. Acima as propriedades e respetivas configurações de altura, comprimento e decoração.

Listview

ListView inserido no Layout

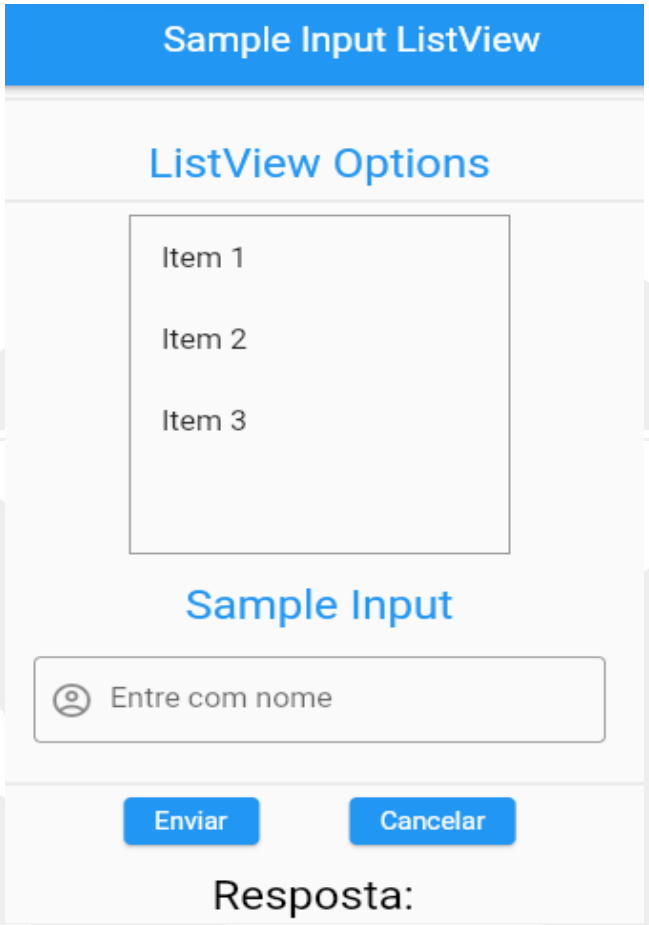
```
child: ListView.builder(
  shrinkWrap: true, // if scroll
  itemCount: items.length,//total de itens
  itemBuilder: (BuildContext context, int index) {
    return ListTile(
      //leading: Image(image: items[index]),
      title: Text(items[index]),
      //subtitle: Text(items[index]),
      onTap: () {
        // Handle tap on the item
        changeSelectedItem('Item selecionado: ${index + 1}');
      },
    );
  },
),
```

clique no item

onTap: evento que identifica que um item foi selecionado. Este listView tem propriedades pré-definidas, como Leading (imagem que aparece ao lado do item do ListView), Title (Título do item) e subTitle (Sub título do item). ListBuilde renderiza o Widget.

Listview

ListView em execução:



O exemplo, ainda contém funções específicas que permitem verificar se o nome e um item foram informados.

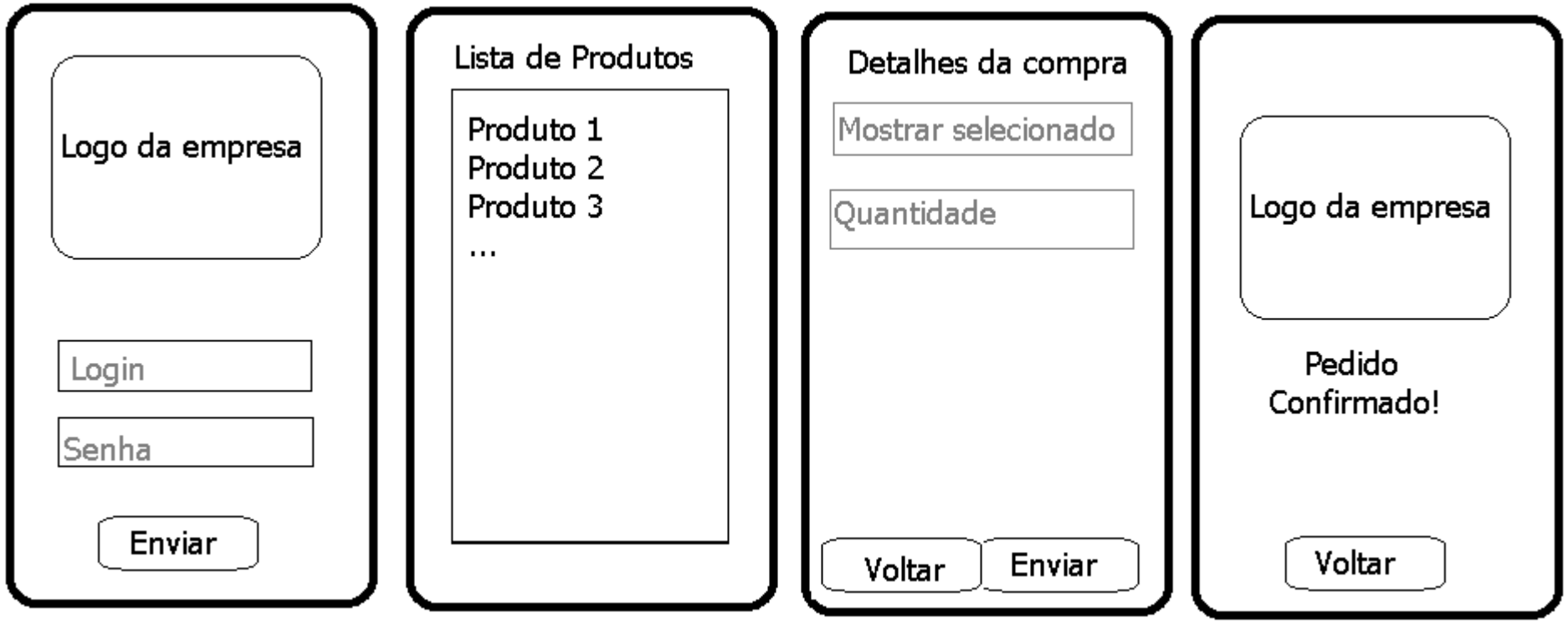
```
// obtém item selecionado e armazena
void changeSelectedItem(String e) {
  setState(() {
    _selectItem = e;
    _result = _selectItem;
  });
}

// botão de envio
void _enviar() {
  //obtém informações do usuário
  String nome = _nome.text;
  String selecionado = _selectItem;

// define resposta
setState(() {
  if (nome == "" || selecionado == "") {
    changeTextColor(Colors.red);
    if (nome == "") {
      _result = "Campo nome obrigatório";
    }
    else
    if (selecionado == "") {
      _result = "Campo Item obrigatório";
    }
```

Listview

Atividade: Implementar um aplicativo em Flutter que permita gerenciar as informações de um carrinho de compras (básico). O Layout deve seguir o modelo abaixo:



Atenção:Utilize os exemplos anteriores (vistos em aula) para auxiliar no desenvolvimento do aplicativo solicitado. Não é necessário armazenar as informações e nem acumular produtos selecionados.

Listview

Material de apoio disponível em:

https://docs.google.com/document/d/1cL7QcstqxB4IfstC7OnkDGgizEBGwxDKjNi-O7lPSC8/edit?usp=sharing

Customizando Listview

```
List<dynamic> _data = [
{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  }
},
  ... restante das informações do vetor
];
```

A demonstração da customização do ListView irá utilizar lista dinâmica definida ao lado. Basicamente é uma estrutura JSON.

Customizando Listview

```
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'HTTP Request Example',
    home: Scaffold(
      appBar: AppBar(
        title: const Text('HTTP Request Example'),
      ),
      body: ListView.builder(
        itemCount: _data.length,
        itemBuilder: (BuildContext context, int index) {
          final item = _data[index];
          return Padding(
            padding: const EdgeInsets.all(8.0),
            child: Container(
              decoration: BoxDecoration(
                color: Colors.grey[200],
                borderRadius: BorderRadius.circular(8.0),
              ),
              child: Padding(
                padding: const EdgeInsets.all(16.0),
                child: Column(
```

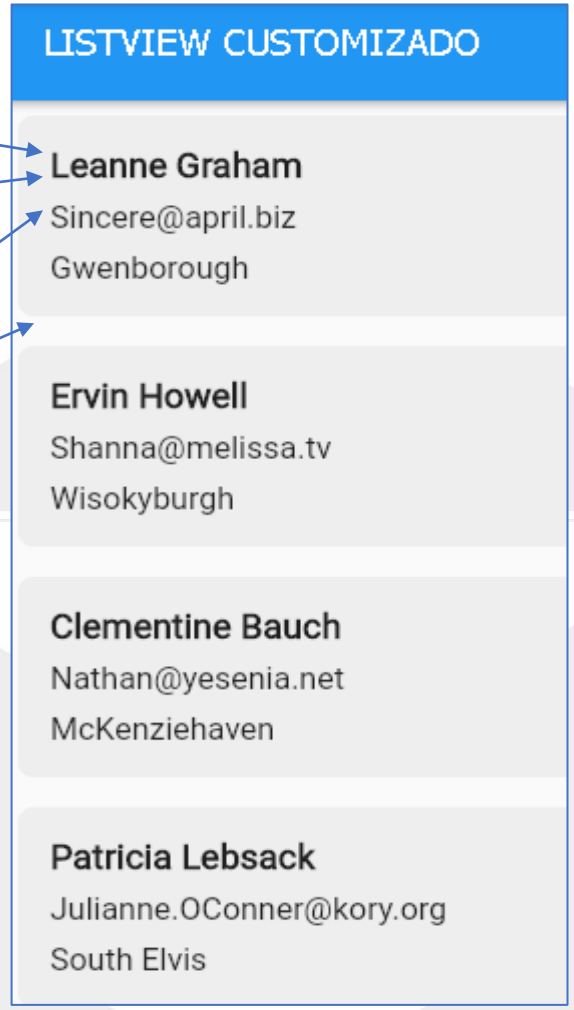
ListView Builder

Acesso individual (item)

Item: o atributo item recebe um a um as informações que estão definidas na lista dinâmica (representadas por uma estrutura JSON).

Customizando Listview

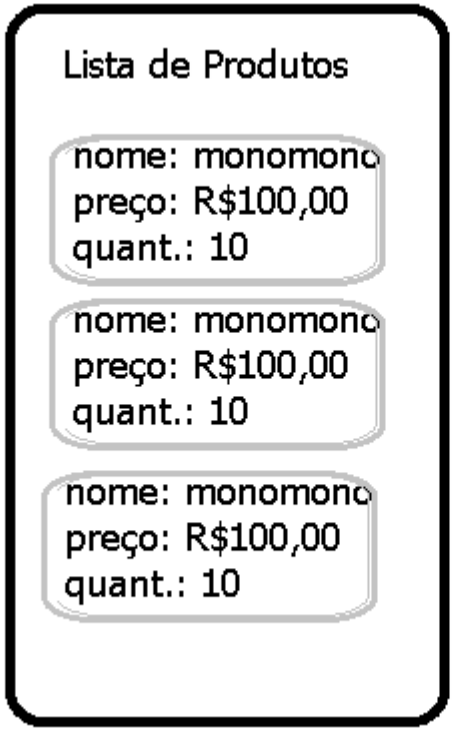
```
child: Padding(
  padding: const EdgeInsets.all(16.0),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Text(
        item['name'],
        style: const TextStyle(
          fontSize: 18.0,
          fontWeight: FontWeight.bold,
        ),
      ),
      const SizedBox(height: 8.0),
      Text(
        item['email'],
        style: const TextStyle(
          fontSize: 16.0,
        ),
      ),
```

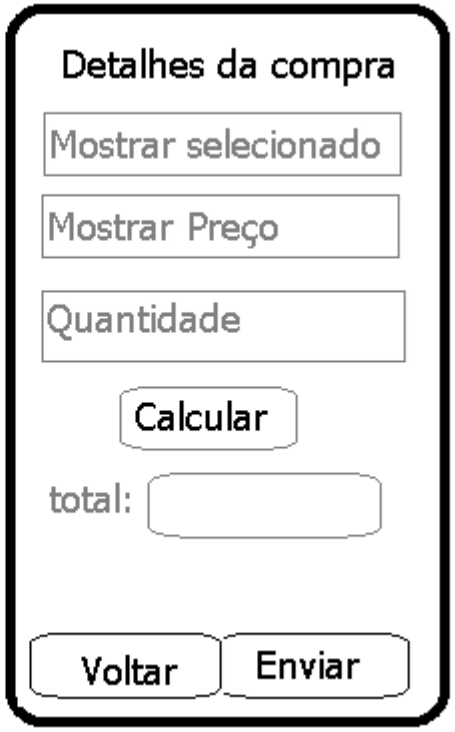


Atenção:
CrossAxisAlignment organiza os textos na vertical, inciando pelo início do Widget Column que por sua vez possui vários filhos, definidos por Children (Widget por Widget com respectivo conteúdo e configuração)

Listview

Atividade: Implementar um outro aplicativo (aproveitar o que foi desenvolvido no anterior)em Flutter que permita gerenciar as informações de um carrinho de compras (básico). O Layout deve seguir o modelo abaixo:





Atenção: A quantidade, solicitada na tela 3, diz respeito a quantidade que o usuário deseja adquirir do produto. O botão calcular deve mostrar o total a ser pago pelo produto, em função da quantidade informada.

* Não é necessário fazer controle de estoque e um produto por vez (não acumula produtos).

Listview

Material de apoio disponível em:

https://docs.google.com/document/d/14GaNpyoSd_IhPpCvuOqdizDxWASwltWZeCDo2Dd_ahw/edit?usp=sharing