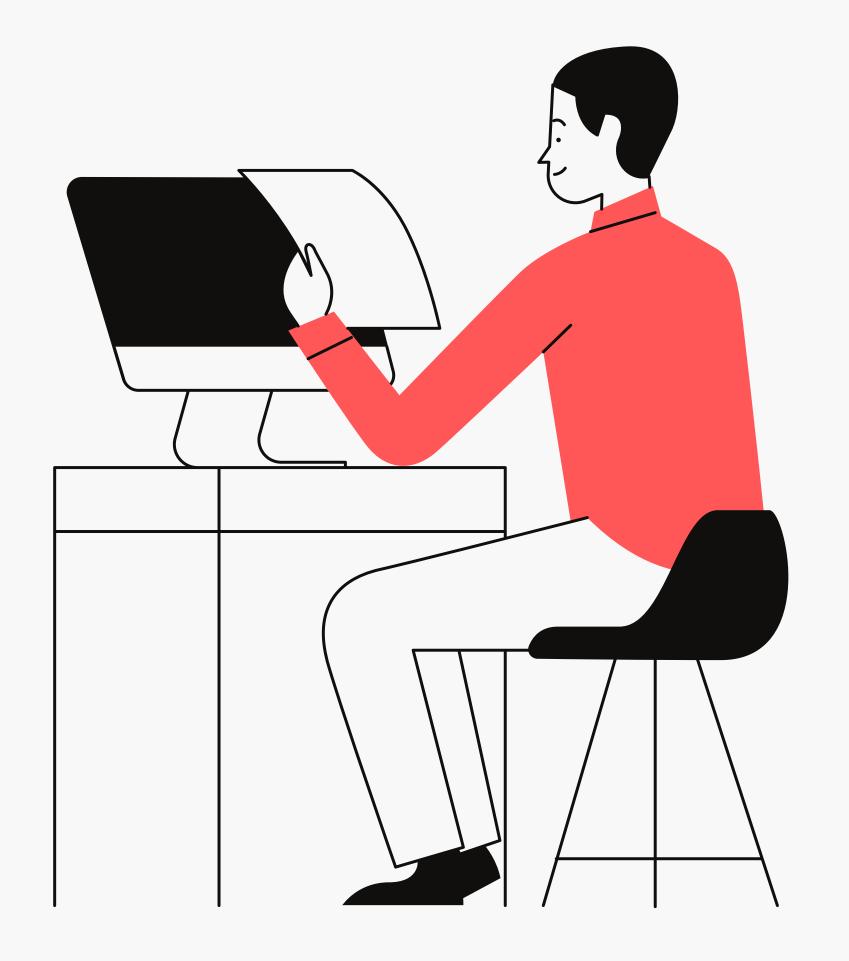
## Flutter

Curso Flutter de Verão



- Construir Container
- SizedBox e MediaQuery
- Expanded e Flexible
- Buttons

# Objetivo da aula de hoje



#### Container

Container é um widget que combina vários recursos, como posicionamento, tamanho, preenchimento, margens e decoração, em um único widget. Ele oferece muita flexibilidade e é ótimo quando você precisa personalizar diversos aspectos do seu layout em um único lugar.

## Comportamento

O contêiner tenta, em ordem: honrar o alinhamento, dimensionar-se para o filho, honrar a largura, altura e restrições, expandir para caber no pai, ser o menor possível.

```
Container(
  constraints: BoxConstraints.expand(
    height: Theme.of(context).textTheme.headlineMedium!.
  padding: const EdgeInsets.all(8.0),
  color: Colors.blue[600],
  alignment: Alignment.center,
  transform: Matrix4.rotationZ(0.1),
  child: Text('Hello World',
    style: Theme.of(context)
        .textTheme
        .headlineMedium!
        .copyWith(color: Colors.white)),
```

#### SizedBox

SizedBox é um widget mais simples, cujo único propósito é impor um tamanho fixo a um widget. É usado quando você precisa garantir que um widget tenha uma altura e largura específicas.

```
const SizedBox(
  width: 200.0,
  height: 300.0,
  child: Card(child: Text('Hello World!')),
)
```

# MediaQuery

MediaQuery é uma classe do Flutter que fornece informações sobre o ambiente atual do aplicativo, como o tamanho do dispositivo e as configurações de orientação.

O MediaQuery pode ser usado em conjunto com o **SizedBox** para garantir que o tamanho do widget seja adaptado com base nas informações do ambiente.

```
SizedBox(
  width: MediaQuery.of(context).size.width,
  height: MediaQuery.of(context).size.height * 0.50,
  child: Text("Hello World")
)
```

#### Flexible

O Flexible é um widget usado dentro de um Flex (Row/Column) para definir como um filho específico deve ocupar o espaço disponível. O Flexible recebe um parâmetro chamado flex, que é um valor inteiro que determina a proporção de espaço que o widget filho ocupará em relação aos demais filhos flexíveis dentro do mesmo Flex.

# Expanded

O widget "Expanded" é um widget flexível usado para para preencher o espaço restante dentro do "Flex" após a colocação de outros filhos.

Quando você coloca um filho dentro de um "Expanded", esse filho tentará ocupar todo o espaço disponível no eixo principal do "Flex" (horizontal em "Row" e vertical em "Column")

```
Row(
  children: [
    Flexible(
      flex: 2,
      child: Container(
        color: Colors.blue,
        height: 50,
    Expanded(
      flex: 1,
      child: Container(
        color: Colors.green,
        height: 50,
      ),
```

#### Button

**ElevatedButton**: botão com um efeito de elevação (sombra) quando pressionado.

FloatingActionButton: botão circular flutuante colocado geralmente na parte.

**TextButton**: exibe texto simples, geralmente sem borda e elevação.

**IconButton**: contém apenas um ícone, sem texto e sem elevação.

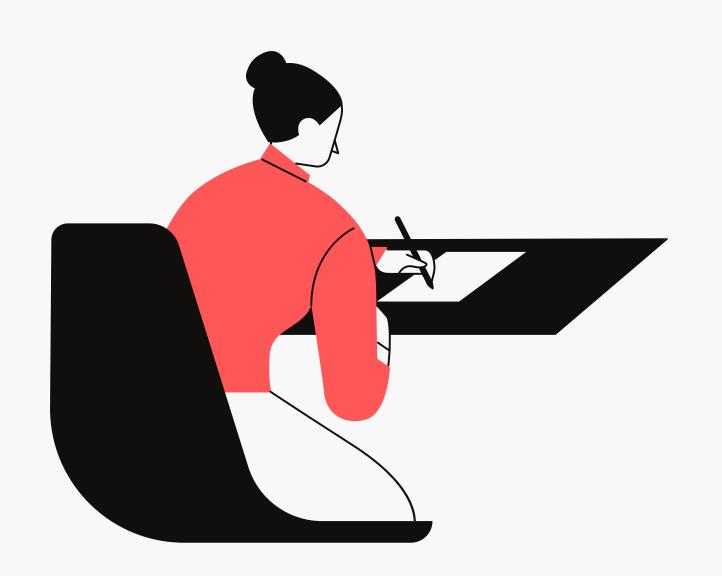
```
ElevatedButton(
  onPressed: () {},
  onLongPress: () {},
  onFocusChange: (value) {},
  onHover: (value) {},
  style: ElevatedButton.styleFrom(
    foregroundColor: Colors.white,
    backgroundColor: Colors.cyan[300],
   minimumSize: const Size(88, 36),
    padding: const EdgeInsets.symmetric(horizontal: 16),
    shape: const RoundedRectangleBorder(
      borderRadius: BorderRadius.all(Radius.circular(2)),
  child: const Text('Elevated Button'),
```

```
TextButton(
  child: const Text(
    "Text Button",
    style: TextStyle(
      color: Colors.green,
      fontSize: 18
  onPressed: () {},
```

```
///WithOut Extended FloatingActionButton
FloatingActionButton(
 backgroundColor: Colors.yellow,
 foregroundColor: Colors.white,
 onPressed: () {},
 child: const Icon(Icons.person,color: Colors.black,),
/// With Extended FloatingActionButton
FloatingActionButton.extended(
 backgroundColor: Colors.yellow,
 foregroundColor: Colors.white,
 onPressed: () {},
  icon: const Icon(Icons.person,color: Colors.black,),
  label: const Text("Add",style: TextStyle(color: Colors.black),),
```

```
IconButton(
  icon: const Icon(Icons.flutter_dash),
  iconSize: 50,
  color: Colors.blue[500],
  tooltip: 'IconButton',
  onPressed: () {
  },
```

## Exercício

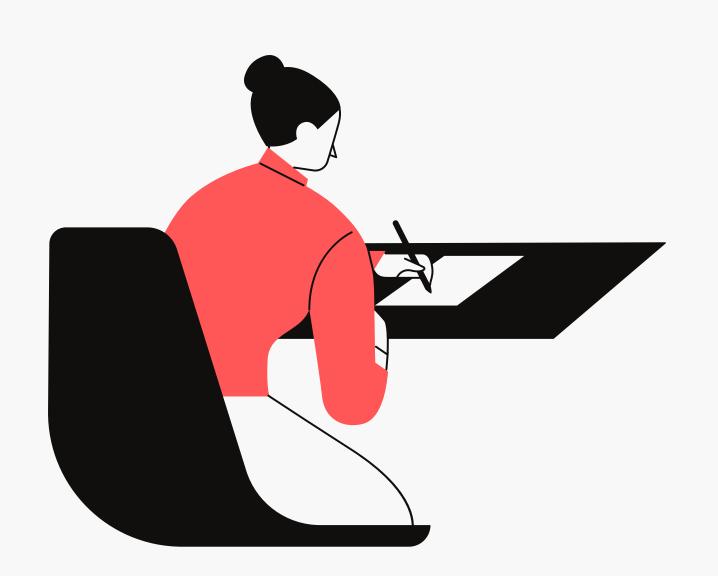


Crie um contêiner personalizado no centro da tela do usuário.

Esse contêiner deve ser personalizado e ter as configurações de posicionamento, tamanho, preenchimento, margens e decoração.

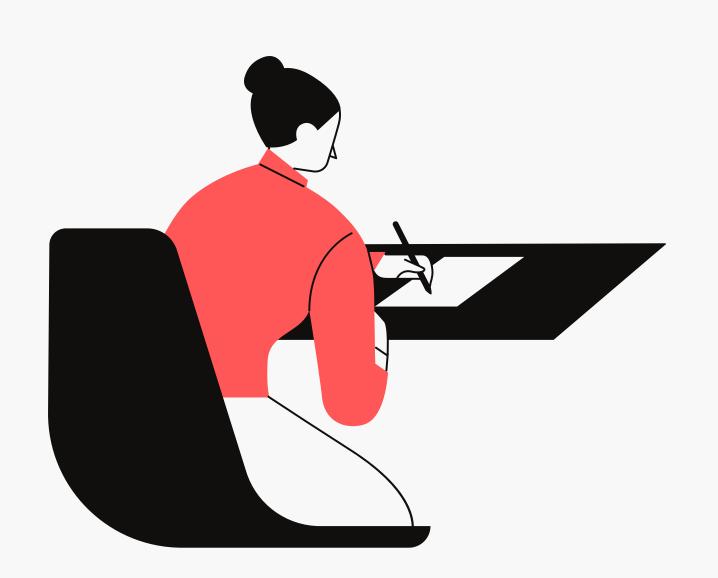
Seja criativo!

## Exercício



Utilizando o código da aula anterior: torne a interface do usuário mais responsiva, de forma a se adaptar a qualquer tamanho de tela do usuário com o uso do MediaQuery. Utilize os widgets Expanded ou Flexible para proporcionar uma distribuição proporcional dos elementos na interface do usuário.

### Exercício



Adicione mais 2 categorias com pratos ao seu mapa. Posteriormente construa Containers personalizados para apresentar de maneira criativa cada um desses pratos separados pelas categorias determinadas.