Desenvolver Interface Gráfica Para Dispositivos Móveis

Aula 14

Professor: Henrique Augusto Maltauro

Fecomércio PR
Sesc Senac IFPD
Senac

Flutter/Dart

- **Row**

O Row é uma classe/widget do Flutter que implementa uma lista de elementos, permitindo que eles sejam apresentados um do lado do outro.

Flutter/Dart

- **Row**

As principais propriedades do Row são:

➔ **children**

- Que recebe uma lista de Widgets para serem apresentados em uma lista horizontal.

Flutter/Dart

- **Row**

➔ **mainAxisAlignment**

- Que recebe um MainAxisAlignment para definir o alinhamento dos componentes no eixo principal.

➔ **crossAxisAlignment**

- Que recebe um CrossAxisAlignment para definir o alinhamento dos componentes no eixo transversal.

Flutter/Dart

- **Row**

➔ **textDirection**

- Que recebe um TextDirection? para definir a disposição horizontal dos componentes.

➔ **verticalDirection**

- Que recebe um VerticalDirection para definir a disposição vertical dos componentes.

Flutter/Dart

- **Row**

```
Row(
    mainAxisAlignment: MainAxisAlignment,
    crossAxisAlignment: CrossAxisAlignment,
    textDirection: TextDirection?,
    verticalDirection: VerticalDirection,
    children: Widget[]
)  // Row
```

Flutter/Dart

- **Column**

Algumas outras propriedades do Column são:

➔ **mainAxisAlignment**

- Que recebe um MainAxisAlignment para definir o alinhamento dos componentes no eixo principal.

➔ **crossAxisAlignment**

- Que recebe um CrossAxisAlignment para definir o alinhamento dos componentes no eixo transversal.

Flutter/Dart

- **Column**

➔ **textDirection**

- Que recebe um TextDirection? para definir a disposição horizontal dos componentes.

➔ **verticalDirection**

- Que recebe um VerticalDirection para definir a disposição vertical dos componentes.

Flutter/Dart

- **Column**

```
Column(
    mainAxisAlignment: MainAxisAlignment,
    crossAxisAlignment: CrossAxisAlignment,
    textDirection: TextDirection?,
    verticalDirection: VerticalDirection
)  // Column
```

Flutter/Dart

- **MainAxisAlignment**

O MainAxisAlignment é um enum do Flutter, que define como os componentes devem ser dispostos ao longo do eixo principal de layout.

Ele não é classificado como um componente, ou seja, ele sempre vai precisar estar relacionado a algum outro componente.

Flutter/Dart

- **MainAxisAlignment**

Os valores de enum do MainAxisAlignment são:

➔ **start**

- Que define que os elementos serão posicionados no início do eixo principal.
- Se esse valor for usado em uma direção horizontal, o TextDirection deve definir se o começo é na direita ou na esquerda.
- Se esse valor for usado em uma direção vertical, o VerticalDirection deve definir se o começo é em cima ou em baixo.

Flutter/Dart

- **MainAxisAlignment**

➔ **end**

- Que define que os elementos serão posicionados no fim do eixo principal.
- Se esse valor for usado em uma direção horizontal, o TextDirection deve definir se o fim é na direita ou na esquerda.
- Se esse valor for usado em uma direção vertical, o VerticalDirection deve definir se o fim é em cima ou em baixo.

Flutter/Dart

- **MainAxisAlignment**

➔ **center**

- Que define que os elementos serão posicionados no centro do eixo principal.

➔ **spaceBetween**

- Que define que os elementos serão posicionados no eixo principal com um espaçamento igual entre eles.

Flutter/Dart

- **MainAxisAlignment**

➔ **spaceAround**

- Que define que os elementos serão posicionados no eixo principal com um espaçamento igual entre eles, mas com metade desse espaço ficando nas bordas do alinhamento.

➔ **spaceEvenly**

- Que define que os elementos serão posicionados no eixo principal com um espaçamento igual entre eles, e com esse mesmo espaçamento nas bordas do alinhamento.

Flutter/Dart

- **MainAxisAlignment**

```
MainAxisAlignment.start;
MainAxisAlignment.end;
MainAxisAlignment.center;
MainAxisAlignment.spaceBetween;
MainAxisAlignment.spaceAround;
MainAxisAlignment.spaceEvenly;
```

Flutter/Dart

- **CrossAxisAlignment**

O CrossAxisAlignment é um enum do Flutter, que define como os componentes devem ser dispostos ao longo do eixo transversal de layout.

Ele não é classificado como um componente, ou seja, ele sempre vai precisar estar relacionado a algum outro componente.

Flutter/Dart

- **CrossAxisAlignment**

Os valores de enum do CrossAxisAlignment são:

➔ **start**

- Que define que os elementos serão posicionados no início do eixo transversal.
- Se esse valor for usado em uma direção horizontal, o TextDirection deve definir se o começo é na direita ou na esquerda.
- Se esse valor for usado em uma direção vertical, o VerticalDirection deve definir se o começo é em cima ou em baixo.

Flutter/Dart

- **CrossAxisAlignment**

➔ **end**

- Que define que os elementos serão posicionados no fim do eixo transversal.
- Se esse valor for usado em uma direção horizontal, o TextDirection deve definir se o fim é na direita ou na esquerda.
- Se esse valor for usado em uma direção vertical, o VerticalDirection deve definir se o fim é em cima ou em baixo.

Flutter/Dart

- **CrossAxisAlignment**

➔ **center**

- Que define que os elementos serão posicionados no centro do eixo transversal.

➔ **stretch**

- Que define que os elementos ocuparam todo o espaço do eixo transversal.

Flutter/Dart

- **CrossAxisAlignment**

➔ **baseline**

- Que define que os elementos serão posicionados ao longo do eixo transversal, respeitando a sua linha de base.

Flutter/Dart

- **CrossAxisAlignment**

```
CrossAxisAlignment.start;
CrossAxisAlignment.end;
CrossAxisAlignment.center;
CrossAxisAlignment.stretch;
CrossAxisAlignment.baseline;
```

Flutter/Dart

- **TextDirection**

O TextDirection é um enum do Flutter, que define a direção que o texto flui horizontalmente.

Ele não é classificado como um componente, ou seja, ele sempre vai precisar estar relacionado a algum outro componente.

Flutter/Dart

- **TextDirection**

Os valores de enum do TextDirection são:

- **rtl**
  - Que define que o texto flui do direita para a esquerda.
- **ltr**
  - Que define que o texto flui do esquerda para a direita.

Flutter/Dart

- **TextDirection**

```
TextDirection.rtl;
TextDirection.ltr;
```

Flutter/Dart

- **VerticalDirection**

O VerticalDirection é um enum do Flutter, que define a direção que os elementos fluem verticalmente.

Ele não é classificado como um componente, ou seja, ele sempre vai precisar estar relacionado a algum outro componente.

Flutter/Dart

- **VerticalDirection**

Os valores de enum do VerticalDirection são:

- **up**
  - Que define que os elementos fluem de baixo pra cima.
- **down**
  - Que define que os elementos fluem de cima pra baixo.

Flutter/Dart

- **VerticalDirection**

```
VerticalDirection.up;
VerticalDirection.down;
```

Flutter/Dart

- **Card**

O Card é uma classe/widget que implementa um card, ou seja, um painel com cantos levemente arredondados e uma sombra de elevação, seguindo o padrão do Material Design.

Flutter/Dart

- **Card**

As principais propriedades do Card são:

➔ **child**

- Que recebe um Widget? para definir o componente que será apresentado dentro do card.

➔ **color**

- Que recebe um Color? para definir a cor de fundo do card.

Flutter/Dart

- **Card**

➔ **elevation**

  - Que recebe um double? para definir a elevação do card.

➔ **margin**

  - Que recebe um EdgeInsetsGeometry? uma margem que circula o card.

Flutter/Dart

- **Card**

```
Card(
    child: Widget?,
    color: Color?,
    elevation: double?,
    margin: EdgeInsetsGeometry?
)  // Card
```

Flutter/Dart

- **Card**

➔ **shadowColor**

- Que recebe um Color? para definir a cor de sombra do card.

➔ **shape**

- Que recebe um ShapeBorder? para definir o formato personalizado do card.

Flutter/Dart

- **Card**

➔ **surfaceTintColor**

- Que recebe um Color? para definir a cor da elevação do card.

Flutter/Dart

- **Card**

```
Card(
    shadowColor: Color?,
    shape: ShapeBorder?,
    surfaceTintColor: Color?
)  // Card
```

Flutter/Dart

- **SizedBox**

O SizedBox é uma classe/widget do Flutter que implementa um elemento com um tamanho fixo.

Flutter/Dart

- **SizedBox**

As principais propriedades do SizedBox são:

➔ **child**

- Que recebe um Widget? para definir o componente que será apresentado dentro do SizedBox.

Flutter/Dart

- **SizedBox**

➔ **height**

  - Que recebe um double? para definir a altura do componente.

➔ **width**

  - Que recebe um double? para definir a largura do componente.

Flutter/Dart

- **SizedBox**

```
SizedBox(
    child: Widget?,
    height: double?,
    width: double?
)  // SizedBox
```

Flutter/Dart

- **Container**

O Container é uma classe/widget do Flutter que implementa um container para os elementos.

Similar a uma div do HTML.

Muito versátil por possibilitar definir tamanhos, posicionamentos e coloração aos componentes.

Flutter/Dart

- **Container**

As principais propriedades do Container são:

➔ **child**

- Que recebe um Widget? para definir o componente que será apresentado dentro do container.

➔ **alignment**

- Que recebe um AlignmentGeometry? para definir o alinhamento do componente filho.

Flutter/Dart

- **Container**

➔ **color**

- Que recebe um Color? para definir a cor de fundo.

➔ **decoration**

- Que recebe um Decoration? para definir uma estilização para o fundo.

Flutter/Dart

- **Container**

```
Container(
    child: Widget?,
    alignment: AlignmentGeometry?,
    color: Color?,
    decoration: Decoration?
)  // Container
```

Flutter/Dart

- **Container**

➔ **margin**

- Que recebe um EdgeInsetsGeometry? para definir uma margem ao redor do decoration.

➔ **padding**

- Que recebe um EdgeInsetsGeometry? para definir um espaçamento por dentro do decoration.

Flutter/Dart

- **Container**

➔ **transform**

- Que recebe um Matrix4? para definir uma transformação para o container depois que ele já foi colorido e decorado.

Flutter/Dart

- **Container**

```
Container(
    margin: EdgeInsetsGeometry?,
    padding: EdgeInsetsGeometry?,
    transform: Matrix4?
)  // Container
```

Exercício

Exercício

1. **Recriar uma tela com cards dispostos assim:**

github.com/hmaltaurodev/slides