Aluno: João Vitor da Silva

Aula dia 05/05

Desenvolvimento de Dispositivos Moveis

Atividade Extra

Boas Práticas:

Organização de código, Comentários, Uso de recursos, Responsividade, Testes, Segurança e Documentação.

View:

View é uma classe fundamental em Android que representa um elemento de interface do usuário. Ela é usada para exibir e interagir com elementos gráficos na tela do dispositivo.

a classe View é a base para muitas outras classes de interface do usuário em Android, como Button, TextView, ImageView e EditText. Cada uma dessas classes herda da classe View e adiciona suas próprias funcionalidades e propriedades.

a View pode ser configurada com várias propriedades, como tamanho, posição, cor e transparência. Além disso, a View também pode responder a eventos de entrada do usuário, como toques, cliques e gestos.

Componentes e Atributos:

- ScrollView: é uma classe que permite rolar um conteúdo maior do que a tela do dispositivo. se você tiver um layout com muitos elementos que não cabem na tela, você pode colocá-los dentro de um ScrollView e permitir que o usuário role para ver o conteúdo oculto.

- CardView: é uma classe que representa um cartão em Material Design ele é usado para exibir informações de forma organizada e clara em um layout. O CardView pode conter imagens, texto e outros elementos de interface do usuário.

- Orientation: é para deixar o app na horizontal ou vertical

- Componentes obrigatórios

  - Layout_width (Altura)

  - Layout_height (Largura)

Atributos comuns:

- Layout_width e layout_height: definem a largura e a altura do elemento na tela.

- Background: define a cor de fundo do elemento.

- Text: define o texto a ser exibido no elemento, como um TextView ou Button.

- Src: define a imagem a ser exibida em um ImageView.

- OnClick: define o método a ser chamado quando o elemento for clicado.

- Visibility: define se o elemento é visível ou não na tela.

- Padding: define o espaço em branco em torno do conteúdo do elemento.

- Gravity: define a posição do conteúdo dentro do elemento, como no centro ou à esquerda.

LAYOUT:

Estrutura visual que define como os elementos de interface do usuário, como botões, imagens, caixas de texto e outros componentes, são dispostos na tela.

- É gerado por um arquivo XML

- É organizado em forma de árvore

Contêineres:

- LinearLayout: organiza seus filhos em uma linha ou coluna única, é útil para layouts simples

- RelativeLayout: organiza seus filhos com relação às posições uns dos outros e aos limites do contêiner pai

- ConstraintLayout: é um layout flexível que permite definir as relações de posição entre elementos visuais de forma mais fácil e eficiente, é recomendado para layouts mais complexos

- FrameLayout: organiza os filhos em camadas, onde cada filho se sobrepõe ao anterior, é útil para adicionar widgets flutuantes ou para sobrepor elementos visuais

Outros:

Termos comuns:

Atividades: ‘Activity’

Constantes: ‘Constants’

Fragmentos: ‘Fragments’

Visualizações: ‘Views’

Intenções: ‘Intents’

Recursos': 'Resources'

Manifesto': 'Manifest'

Gradle: 'Gradle'

- OnCreate()

  Sempre chama um super na pasta pai

  Define para qual tela vai ir

- A Classe 'R' é gerada em todo build do app

- ViewGroup:

  Armazenar um ou mais views

  São Layouts

- Medição DP: é uma unidade de medida usada em Android para garantir que o layout de um aplicativo fique consistente em diferentes dispositivos com diferentes densidades de pixels.

- Match_Parent: é usado para definir que a largura ou altura de um componente deve corresponder à largura ou altura do componente pai, o componente ocupará todo o espaço disponível no componente pai na direção especificada.

- Wrap_Content: é usado para definir que a largura ou altura de um componente deve se ajustar ao tamanho do conteúdo dentro dele, o componente será dimensionado automaticamente para se ajustar ao conteúdo que ele contém.

- Para dimensões usar nomes genéricos

- Categorias: (Small- Normal - Large - XLarge)

- Para alterar as margens: Start - End

- TextView é usado para textos estáticos, é usado para exibir informações, mensagens, instruções e outros tipos de conteúdo textual em um aplicativo

- Exemplo de acesso ao componente da tela botão:

val btnTraduzir = findViewByID<Button>(R.id.main_btn_traduzir)

- Entidades começa sempre com Main_...