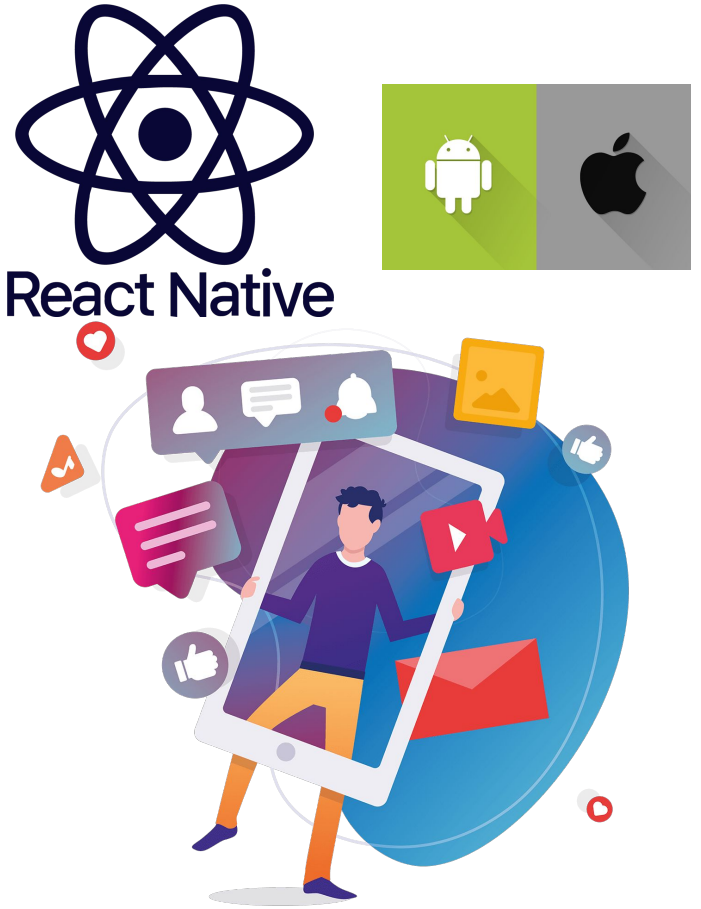
Aula 02 - React Native

Prof. MSc. Kelson Almeida

Flexbox

- **Flexbox** é um modelo de layout que permite alinhar e distribuir espaço entre itens em um container, mesmo quando seu tamanho é desconhecido.
- Flexbox torna fácil criar layouts responsivos e adaptáveis
- Funciona com base em um container (flex container) e seus filhos (flex items).



Configurando um Flex Container

- Todo <View> no React Native é um flex container potencial.
- Propriedades principais: *flexDirection*, *justifyContent*, *alignItems*.
  - flexDirection: controla a direção dos itens filhos (horizontal ou vertical).
  - justifyContent: alinha itens no eixo principal.
  - alignItems: alinha itens no eixo cruzado.

jsx

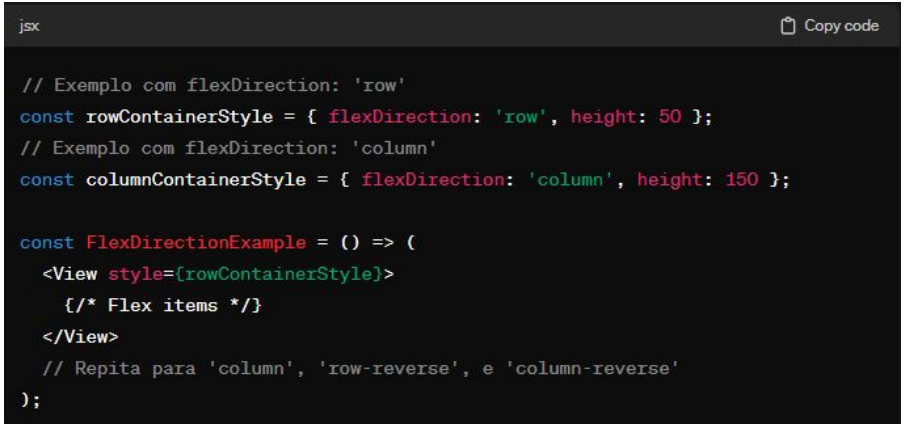
```jsx
import React from 'react';
import { View } from 'react-native';

const containerStyle = {
  flexDirection: 'column', // ou 'row'
  justifyContent: 'center', // Alinhamento no eixo principal
  alignItems: 'center', // Alinhamento no eixo cruzado
  height: 200, // Altura do container para visualização
};

const FlexContainer = () => (
  <View style={containerStyle}>
    {/* Flex items aqui */}
  </View>
);
```

Flex Direction

- Opções: row, column, row-reverse, column-reverse.



Justify Content

- Opções: flex-start, center, flex-end, space-between, space-around, space-evenly.

jsx Copy code

```jsx
const justifyContentStyle = {
  flexDirection: 'row',
  justifyContent: 'space-between', // Experimente com diferentes valores aqui
  height: 50,
};

const JustifyContentExample = () => (
  <View style={justifyContentStyle}>
    {/* Flex items */}
  </View>
);
```

Align Items

- Opções: flex-start, center, flex-end, stretch, baseline.

```jsx
const alignItemsStyle = {
  flexDirection: 'row',
  alignItems: 'center', // Experimente com diferentes valores aqui
  height: 100,
};

const AlignItemsExample = () => (
  <View style={alignItemsStyle}>
    {/* Flex items */}
  </View>
);
```

Criando um Layout Responsivo

- Praticar o uso de Flexbox criando um layout com cabeçalho, conteúdo e rodapé.

jsx Copy code

```jsx
import React from 'react';
import { View, Text } from 'react-native';

const AppLayout = () => (
  <View style={{ flex: 1 }}>
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Cabeçalho</Text>
    </View>
    <View style={{ flex: 2, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Conteúdo Principal</Text>
    </View>
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Rodapé</Text>
    </View>
  </View>
);
```

Vamos praticar? [1] [com o prof]

- **Objetivo:** Criar um layout vertical que contém três <View>s, cada uma com altura de 100 pixels e largura completa da tela. As <View>s devem ter cores de fundo diferentes para distingui-las visualmente.

Vamos praticar? [2] [com o prof]

- **Objetivo:** Dentro de um container <View>, posicionar um item filho <View> no centro vertical e horizontalmente. O item filho deve ter dimensões de 50x50 pixels e uma cor de fundo.

Vamos praticar? [3] [com o prof]

- **Objetivo:** Criar um layout de três <View>s em linha (row), onde a <View> do meio ocupa o espaço restante. As <View>s da esquerda e da direita devem ter 50 pixels de largura e a mesma altura. Todas devem ter cores de fundo diferentes.

Vamos praticar? [4] [com o prof]

- **Objetivo:** Criar um layout vertical onde três <View>s ocupam igualmente o espaço disponível na tela. Cada <View> deve ter uma cor de fundo diferente.

Vamos praticar? [5] [com o prof]

- **Objetivo:** Desenvolver um layout simples para um aplicativo de lista de tarefas que inclua um cabeçalho, uma área para adicionar novas tarefas, e uma lista de tarefas adicionadas.

- Estrutura do Projeto
  - Cabeçalho: Mostrará o título do aplicativo.
  - Área de Entrada de Tarefa: Um campo de texto onde o usuário pode digitar o nome de uma nova tarefa e um botão para adicionar a tarefa à lista.
  - Lista de Tarefas: Área onde as tarefas adicionadas serão listadas.