53 45 52 45 49 20 46 49 45 4c 20
41 4f 53 20 50 52 45 43 45 49 54
4f 53 20 44 41 20 48 4f 4e 52 41
20 45 20 44 41 20 43 49 c3 8a 4e
43 49 41 2c 20 50 52 4f 4d 4f 56
45 4e 44 4f 20 4f 20 55 53 4f 20
45 20 4f 20 44 45 53 45 4e 56 4f
4c 56 49 4d 45 4e 54 4f 20 44 41
20 49 4e 46 4f 52 4d c3 81 54 49
43 41 20 45 4d 20 42 45 4e 45 46
c3 8d 43 49 4f 20 44 4f 20 43 49
44 41 44 c3 83 4f 20 45 20 44 41
20 53 4f 43 49 45 44 41 44 45 2e

RESIDÊNCIA DE SOFTWARE

CAPACITAR
TREINAR
EMPREGAR
TRANSFORMAR

Firjan SENAI

serratec
Parque Tecnológico Região Serrana

FrontEnd Mobile
02 - Ambiente de desenvolvimento

Objetivos da aula

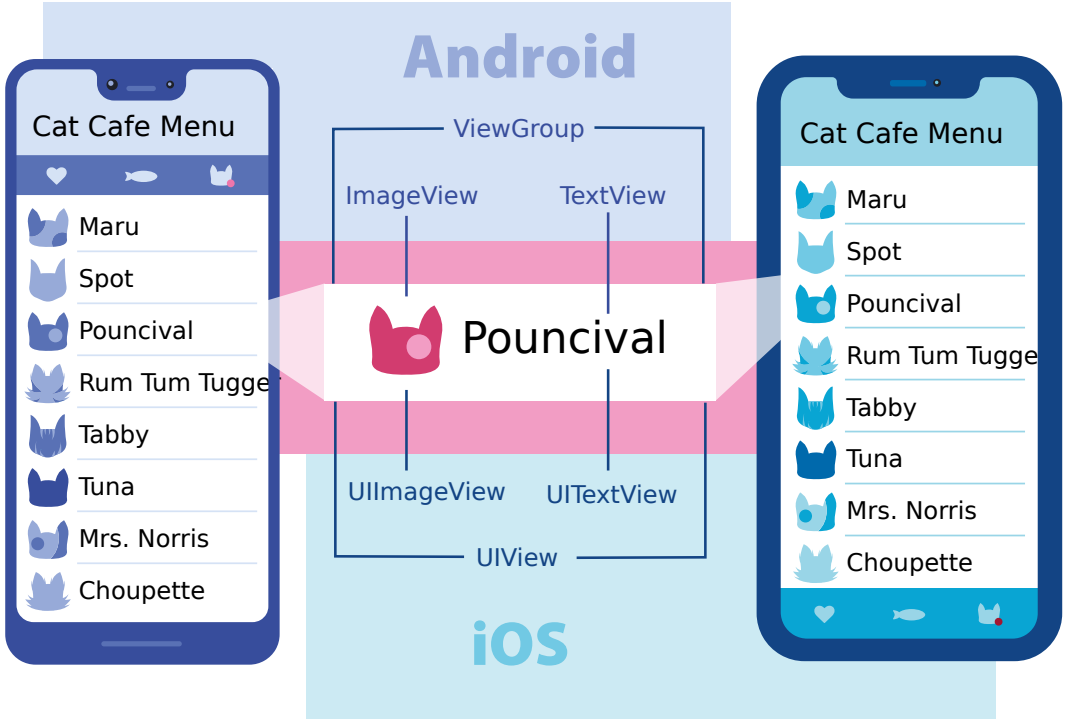
01 – Entendendo Views no Desenvolvimento Mobile

02 – Aprendendo Sobre Core Components

03 – Criando Componentes com TypeScript

Views

Em desenvolvimento para Android e iOS, uma V**iew** é o bloco de construção básico da interface do usuário: um pequeno elemento retangular na tela que pode ser usado para exibir texto, imagens ou responder à interação do usuário. Mesmo os menores elementos visuais de um aplicativo, como uma linha de texto ou um botão, são tipos de views. Alguns tipos de views podem conter outras views. É views até o infinito!



Core Components

Componente React Native	View no Android	View no iOS	Analogia na Web	Descrição
`<View>`	`<ViewGroup>`	`<UIView>`	Um `<div>` sem rolagem	Um contêiner que suporta layout com flexbox, estilo, manipulação de toque e controles de acessibilidade.
`<Text>`	`<TextView>`	`<UITextView>`	`<p>`	Exibe, estiliza e aninha strings de texto, além de lidar com eventos de toque.
`<Image>`	`<ImageView>`	`<UIImageView>`	`<img>`	Exibe diferentes tipos de imagens.
`<ScrollView>`	`<ScrollView>`	`<UIScrollView>`	`<div>` com rolagem	Um contêiner genérico de rolagem que pode conter múltiplos componentes e views.
`<TextInput>`	`<EditText>`	`<UITextField>`	`<input type="text">`	Permite ao usuário inserir texto.

Criando um Componente

Um **componente** é uma função que inicia com a letra maiúscula e retorna (elementos dentro do “return”) um **JSX** (um elemento visual).

Um componente é uma unidade reutilizável de código que define uma parte da interface do usuário. É um “pedacinho da tela”, um “bloco de construção”.

Para que ele possa ser utilizado pela aplicação, devemos sempre **exportá-lo**.

Todos os compentes utilizados no JSX devem ser **importados** no início.

```
1   import { Text, View } from "react-native";
2
3   function MeuComponente() {
4     return (
5       <View>
6         <Text>Meu Componente</Text>
7       </View>
8     );
9   }
10
11  export default MeuComponente;
```

Usando um Componente

Para utilizar um componente, devemos primeiro **importá-lo**

O componente deve ser utilizado dentro do JSX como qualquer outro componente nativo, entre “tags” **<MeuComponente />**

Podemos reutilizar o componente várias vezes.

Cada chamada do componente cria uma cópia nova dele, cada um tendo suas próprias informações.

```
import { StyleSheet } from "react-native";
import MeuComponente from "./src/components/MeuComponente";

export default function App() {
  return <MeuComponente />;
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
    alignItems: "center",
    justifyContent: "center",
  },
});
```

Componente com Props

O Componente pode receber propriedades, como se fossem argumentos de uma função.

Com typescript, devemos **explicitar** qual o tipo da informação que o componente espera receber.

Utilizamos “**type**” para definir um tipo novo, também poderíamos ter usado “interface”.

Podemos listar dentro do “type” quantas variáveis quisermos, inclusive funções.

Alguns tipos são: **string, number, boolen**

```tsx
import { Text, View } from "react-native";

type MeuComponenteProps = {
  name: String;
};

function MeuComponente({ name }: MeuComponenteProps) {
  return (
    <View>
      <Text>Meu Componente</Text>
      <Text>{name}</Text>
    </View>
  );
}

export default MeuComponente;
```

Componente com Props

Para utilizar um componente com props, basta passar a props dentro das “tags” do componente, exatamente com o mesmo nome especificado dentro do “type”

Se chamarmos 3 vezes o mesmo componente e passarmos props diferentes, cada componente gerenciará suas props de forma independente.

Cada “cópia” do componente cuida das suas próprias informações.

Obs: O return só retorna um componente, é preciso englobar todos os demais com <></> ou com uma <View></View>

```
1  import { StyleSheet } from "react-native";
2  import MeuComponente from "./src/components/MeuComponente";
3
4  export default function App() {
5    return (
6      <>
7        <MeuComponente name="João Felipe" />
8        <MeuComponente name="Luiza" />
9        <MeuComponente name="Carlos" />
10     </>
11   );
12 }
```