Interface para Dispositivos Móveis

SESI SENAI < Desenvolvimento de Sistemas >

SESI SENAI

01 { .. Criação de Interface Parte 3

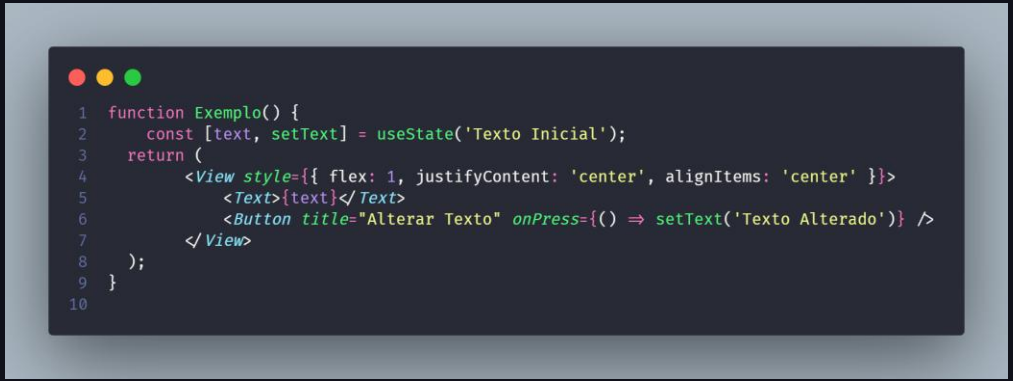
< Introdução a Dispositivos Móveis >

} . .

Controle dos Elementos de Tela

Mudanças de estados:

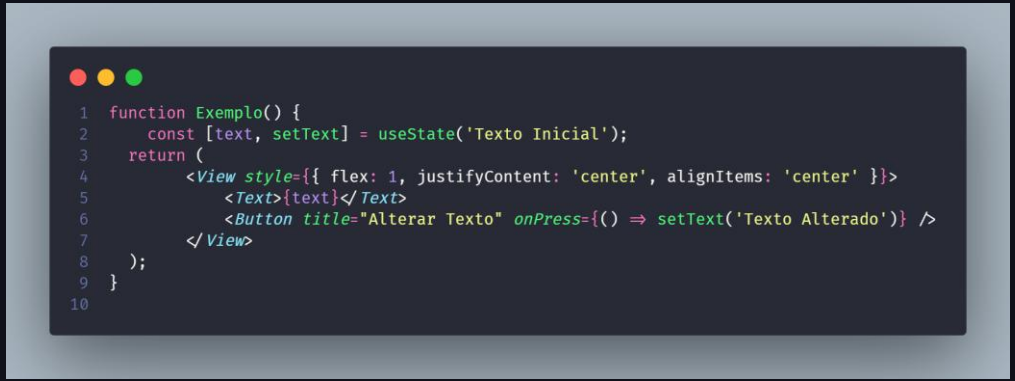
As mudanças de estado em React e React Native referem-se à atualização do estado de um componente, o que pode levar a uma re-renderização desse componente ou de seus filhos. O estado de um componente é um objeto que armazena informações que podem mudar ao longo do tempo, como dados de formulário, status de carregamento, valores selecionados, etc.



```
function Exemplo() {
    const [text, setText] = useState('Texto Inicial');
  return (
        <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
            <Text>{text}</Text>
            <Button title="Alterar Texto" onPress={() => setText('Texto Alterado')} />
        </View>
  );
}
```

Controle dos Elementos de Tela

Os elementos de tela podem ser controlados usando o estado e manipuladores de eventos, precisamos sempre ‘setar’ como foi criado e como ficará alterado após a mudança de estado.
O componente é o { useState } do React.



```
function Exemplo() {
    const [text, setText] = useState('Texto Inicial');
  return (
        <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
            <Text>{text}</Text>
            <Button title="Alterar Texto" onPress={() => setText('Texto Alterado')} />
        </View>
  );
}
```

Tratamento de eventos e exceções

O tratamento de eventos é essencial para interagir com o usuário.
Exceções podem ser tratadas usando blocos try-catch e exibindo mensagens de erro apropriadas.

```
export default function App() {
  const handlePress = () => {
    try {
      // Código que pode gerar uma exceção
    } catch (error) {
      Alert.alert('Erro', error.message);
    }
  };

  return (
    <View>
      <Button title="Pressione" onPress={handlePress} />
    </View>
  );
}
```

Entrada de Dados e Validação

- Entrada de dados pode ser feita através de componentes como TextInput.
- A validação de dados é importante para garantir a integridade e a precisão dos dados inseridos pelo usuário.

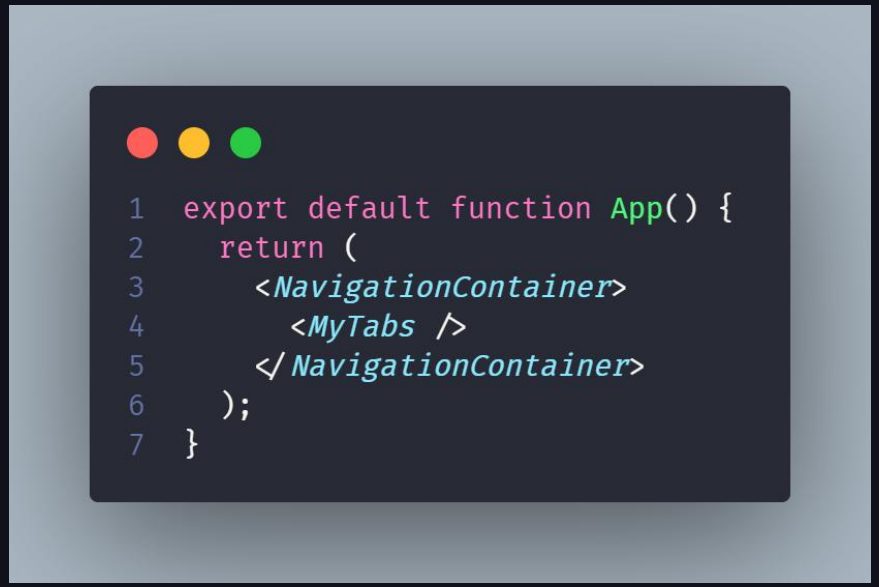
```jsx
export default function App() {
  const [email, setEmail] = useState('');

  const validateEmail = () => {
    if (email.includes('@')) {
      // Processamento de dados
    } else {
      Alert.alert('Erro', 'Por favor, insira um email válido.');
    }
  };

  return (
    <View>
      <TextInput
        placeholder="Digite seu email"
        value={email}
        onChangeText={setEmail}
      />
      <Button title="Enviar" onPress={validateEmail} />
    </View>
  );
}
```

Criando os Navigations

Esse é o modelo de navegação para sua tela com o App.js, você irá colocar as tags de Navigation e Tabs, pois esse modelo é todos as páginas dentro de uma única tela, o correto será criar páginas distintas.



Criando os Navigations

Esse é o modelo de navegação para sua tela, você irá escolher qual a tela inicial, não esqueça de fechar a tag e as chaves.

```
const Tab = createBottomTabNavigator();

function MyTabs() {
  return (
    <Tab.Navigator
      initialRouteName="Home"
      screenOptions={{
        tabBarActiveTintColor: "#e91e63",
      }}
    >
      <Tab.Screen
        name="Home"
        component={Home}
        options={{
          tabBarLabel: "Home",
          headerShown: false,
          tabBarIcon:({focused, size, color}) => {
            if(focused){
                return <Ionicons size={size} color={color} name='home'/>
            }

            return <Ionicons size={size} color={color} name='home-outline'/>
        },
        }}
      />
```

Desafio

Agora com a tela de login criada na última aula, faça validação de login com valores estáticos para entrar na tela Home.