

```
import React, { useState } from "react";
import { Button, SafeAreaView, StyleSheet, Text, View } from "react-native";
```

```
/* RENDERIZAÇÕES CONDICIONAIS:
```

No react-native é muito importante saber fazer renderizações condicionais, por se tratar de aplicativos, muitas vezes vamos ter que mudar toda a renderização de uma página com apenas um click.

Existem formas mais eficazes de fazer renderizações no react. Veja algumas:

```
*/
```

```
//OPERADOR TERNÁRIO:
```

//Abaixo temos um componente com 2 botões que aumentam e abaixam um determinado valor, sempre que o valor é impar ele pinta o quadrado de vermelho, quando é par, pinta o quadrado de verde.

```
export default () => {
  const [num, setNum] = useState(1)
  const [status, setStatus] = useState('impar')

  const comparaValor = (valor) => {
    if((valor % 2) === 0){
      setStatus('impar')
    } else {
      setStatus('par')
    }
  }

  const aumentaValor = () => {
    setNum(num + 1)
    comparaValor(num)
  }

  const abaixaValor = () => {
    setNum(num - 1)
    comparaValor(num)
  }
}
```

```

return (
  <SafeAreaView style={styles.background}>
    <Text style={styles.enunciado}>Se o valor for "par" o quadrado fica vermelho, se for "impar" fica verde...</Text>
    <Text style={styles.texto}>`${num} é ${status}`</Text>
    <View style={styles.wrapperBotoes}>
      <Button title='+' onPress={aumentaValor} />
      <Button title='- ' onPress={abaixaValor} />
    </View>
    {
      /* Aqui temos o uso do ternário, veja que quando o valor é par ele renderiza de um modo, quando é impar ele renderiza
de outro.

      Note que fizemos isso apenas com um trecho de código invés de gerar um if else com todo o componente. */
      num % 2 == 0
      ? <View style={[styles.quadrado, styles.green]} />
      : <View style={[styles.quadrado, styles.red]} />
    }
  </SafeAreaView>
)
}

const styles = StyleSheet.create({
  background: {
    alignItems: 'center',
    flex: 1,
    justifyContent: 'center',
  },
  enunciado: {
    fontSize: 24,
    margin: 10,
  },
  texto: {
    color: 'black',
    fontSize: 20,
    margin: 10,

```

```
    },
    wrapperBotoes: {
      height: 40,
      margin: 10,
      width: 80,
    },
    quadrado: {
      height: 100,
      margin: 40,
      width: 100,
    },
    red: {
      backgroundColor: 'red',
    },
    green: {
      backgroundColor: 'green',
    },
  })
```

ARQUIVO DO APP.JS...

```
import React from 'react'
import RenderizacaoCondicionalComTernario from './src/componentes/RenderizacaoCondicional/RenderizacaoCondicionalComTernario'

export default () => {
  return (
    /* Coloque seus componentes abaixo...*/
    <RenderizacaoCondicionalComTernario />
  )
}
```

## RESULTADO NO CELULAR COM VALOR DE IMPAR...

The image shows a development environment with a mobile emulator on the left and the Visual Studio Code editor on the right. The emulator displays a web page with the text "Se o valor for 'par' o quadrado fica vermelho, se for 'impar' fica verde..." and "1 é impar". Below the text are two squares: a blue one with a "+" sign and a red one with a "-" sign. The Visual Studio Code editor shows the file explorer on the left with a list of files and folders. The main editor area displays the code for "App.js" and "RenderizacaoCondicionalComTernario.js". The terminal at the bottom shows the command prompt and the output of the application.

**Visual Studio Code Interface:**

- File Explorer (EXPLORADOR):** Shows a list of files and folders, including "COMPONENTES", "estilos", "exercicios", "fundamentos", "ComunicacaoDireta", "ComunicacaoIndireta", "EstadosNoReact", "JS index.js", ".buckconfig", ".eslintrc.js", ".flowconfig", ".gitignore", ".node-version", ".prettierrc.js", ".ruby-version", ".watchmanconfig", "JS App.js", "app.json", "babel.config.js", "Gemfile", "index.js", "metro.config.js", "package.json", and "yarn.lock".
- Code Editor:** Displays the code for "App.js" and "RenderizacaoCondicionalComTernario.js".

```
JS App.js > ...
1 import React from 'react'
2 import RenderizacaoCondicionalComTernario from './src/componentes/RenderizacaoCondicional/RenderizacaoCondicionalComTernario'
3
4 export default () => {
5   return (
6     /* Coloque seus componentes abaixo...*/
7     <RenderizacaoCondicionalComTernario />
8   )
9 }
10
```
- Terminal:** Shows the command prompt and the output of the application.

```
<-----> 0% WAITING
> IDLE

info Connecting to the development server...
info Starting the app on "emulator-5554"...
Starting: Intent { cmp=com.componentes/.MainActivity }
```

## RESULTADO NO CELULAR COM VALOR DE PAR...

The image shows a mobile emulator on the left and the Visual Studio Code editor on the right. The emulator displays a text prompt in Portuguese: "Se o valor for 'par' o quadrado fica vermelho, se for 'impar' fica verde..." (If the value is 'even' the square turns red, if it's 'odd' it turns green...). Below the text, it says "2 é par" (2 is even) and shows a blue square with a "+" sign and a green square with a "-" sign. The Visual Studio Code editor shows the file explorer on the left with the project structure. The main editor displays the file `RenderizacaoCondicionalComTernario.js` with the following JavaScript code:

```
71 wrapperBotoes: {
72   height: 40,
73   margin: 10,
74   width: 80,
75 },
76 quadrado: {
77   height: 100,
78   margin: 40,
79   width: 100,
80 },
81 red: {
82   backgroundColor: 'red',
83 },
84 green: {
85   backgroundColor: 'green',
86 },
87 }
```

The bottom panel of Visual Studio Code shows the terminal with the following output:

```
<-----> 0% WAITING
> IDLE

info Connecting to the development server...
info Starting the app on "emulator-5554"...
Starting: Intent { cmp=com.componentes/.MainActivity }
```

The status bar at the bottom indicates the page is 5 of 5, with 353 words, in Portuguese (Brazil), and the date is 07/12/2022 at 08:31.