```jsx
import React from "react";
import { SafeAreaView, Text } from "react-native";

/* PASSAGEM DE PARÂMETROS NOS COMPONENTES

    O react-native aceita que parâmetros sejam passados dentro dos componentes. Isso faz com que os nossos componentes ganhem vida por receber dados e valores que podem ser renderizados na tela, juntamente com a nossa aplicação, veja como funciona a passagem de parâmetros sequêncialmente...

    OBSERVAÇÃO!!!
    Por padrão usamos o nome "props" para chamar o objeto que contém as chaves com valores
 */

export function ParametroVazio(props) {
    /* 1º PARÂMETROS SÃO SEMPRE OBJETOS NO REACT-NATIVE

        O react-native trabalha sempre com objetos durante a passagem de parâmetros, veja que nesse componente recebemos um parâmetro vazio e retorna um objeto vazio.
     */

    console.warn(props)
    return (
        <SafeAreaView>
            <Text>
                Note que um parâmetro que não recebe nada é apenas um objeto vazio.
            </Text>
        </SafeAreaView>
    )
}

export function ParametroComValores(props) {
    /* 2º PARÂMETROS SÃO RECEBIBOS COMO ATRIBUTOS DE UM COMPONENTE:

        Sempre que um Componente recebe outro, na verdade ele está gerando uma instância do componente recebido.
```

```
        Como toda instância, ela pode receber atributos com valores, podemos atribuir valores a ela da seguinte forma: <Componente atributo1={valor} atributo2={valor}>

        Esses atributos são recebidos como um único objeto no componente onde podem ser referenciados pelo nome da chave. Como podemos ver no exemplo abaixo...

        OBS: Veja no componente App.js como os atributos foram passados

     */
    console.warn(props)
    return (
        <SafeAreaView>
            <Text>
                Recebi esse valor como mínimo: {props.min} e este como máximo: {props.max}
            </Text>
            <Text>
                A soma destes valores dá: {props.min + props.max}
            </Text>
        </SafeAreaView>
    )
}

export function ParametroValoresComDesctructuring({nome}){
    /* 3º USANDO DESTRUCTURING PARA FACILITAR AINDA MAIS A OBTENÇÃO DE VALORES:

        Podemos destruturizar as chaves que recebemos do objeto usando destructuring para receber os nomes das chaves diretamente.

        Só relembrando, o destructuring recebe a chave e já atribui sobre uma variável com o mesmo nome da chave, por isso podemos usar o nome da chave diretamente.

        OBS: Veja no componente App.js como os atributos foram passados

     */

    console.warn(nome)
    return (
```

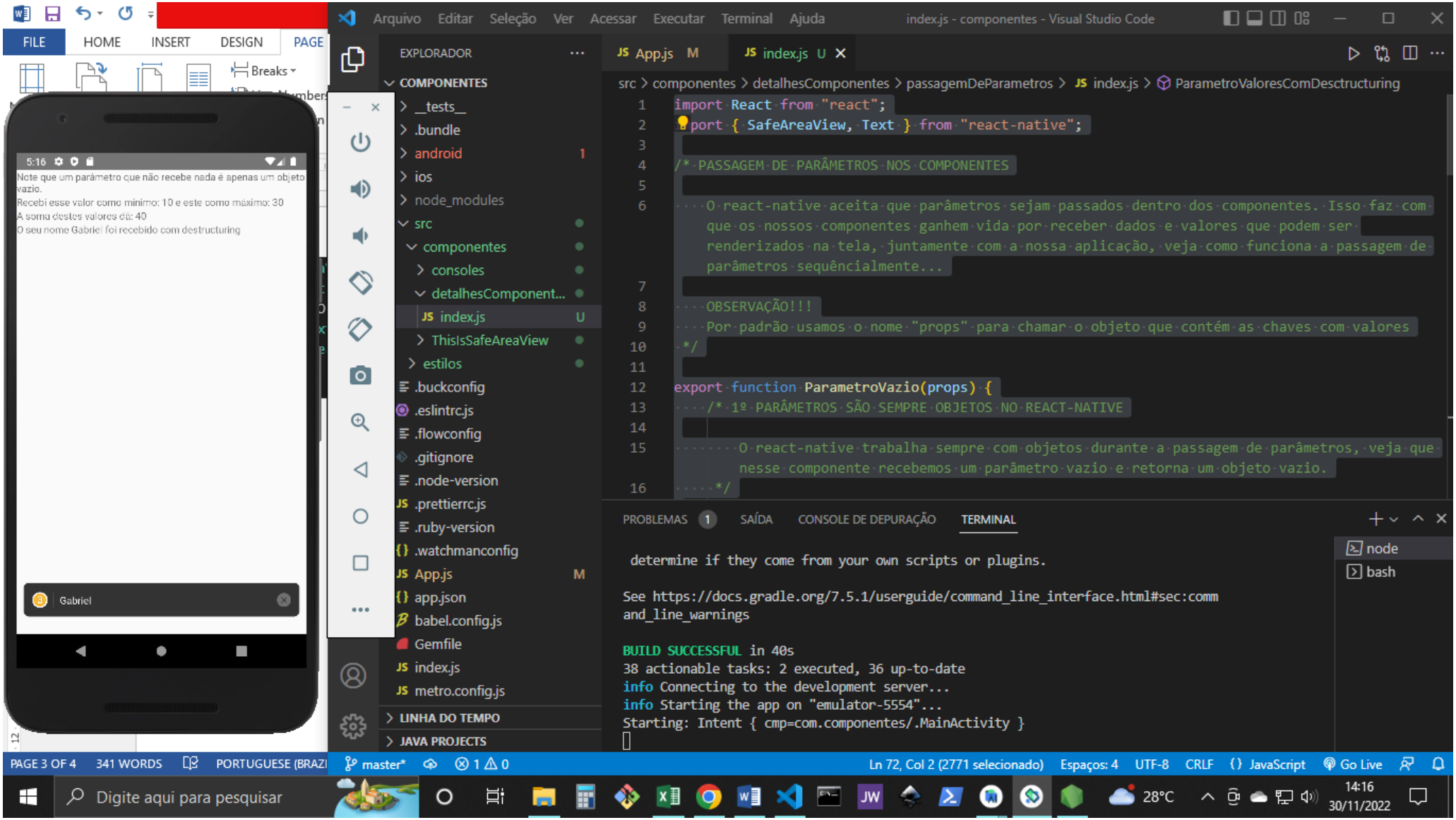
```
        <SafeAreaView>
            <Text>
                O seu nome {nome} foi recebido com destructuring
            </Text>
        </SafeAreaView>
    )
}
```

ARQUIVO APP.JS

```
import React from 'react'
import { ParametroComValores, ParametroValoresComDesctructuring, ParametroVazio } from
'./src/componentes/detalhesComponentes/passagemDeParametros'

export default () => {
  return (
    /* Coloque seus componentes abaixo...*/
    <>
      <ParametroVazio />
      <ParametroComValores min={10} max={30}/>
      <ParametroValoresComDesctructuring nome={'Gabriel'}/>
    </>
  )
}
```

RESULTADO...


FILE
HOME
INSERT
DESIGN
PAGE
Breaks
5:16
Note que um parâmetro que não recebe nada é apenas um objeto vazio.
Recebi esse valor como mínimo: 10 e este como máximo: 30
A soma destes valores dá: 40
O seu nome Gabriel foi recebido com destructuring
Gabriel
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda
index.js - componentes - Visual Studio Code
EXPLORADOR
COMPONENTES
__tests__
.bundle
android
ios
node_modules
src
componentes
consoles
detalhesComponent...
index.js
ThisIsSafeAreaView
estilos
.buckconfig
.eslintrc.js
.flowconfig
.gitignore
.node-version
.prettierrc.js
.ruby-version
.watchmanconfig
App.js
app.json
babel.config.js
Gemfile
index.js
metro.config.js
LINHA DO TEMPO
JAVA PROJECTS
App.js M
index.js U
src > componentes > detalhesComponentes > passagemDeParametros > index.js > ParametroValoresComDesctructuring
import React from "react";
import { SafeAreaView, Text } from "react-native";
/* PASSAGEM DE PARÂMETROS NOS COMPONENTES
O react-native aceita que parâmetros sejam passados dentro dos componentes. Isso faz com que os nossos componentes ganhem vida por receber dados e valores que podem ser renderizados na tela, juntamente com a nossa aplicação, veja como funciona a passagem de parâmetros sequêncialmente...
OBSERVAÇÃO!!!
Por padrão usamos o nome "props" para chamar o objeto que contém as chaves com valores
*/
export function ParametroVazio(props) {
/* 1º PARÂMETROS SÃO SEMPRE OBJETOS NO REACT-NATIVE
O react-native trabalha sempre com objetos durante a passagem de parâmetros, veja que nesse componente recebemos um parâmetro vazio e retorna um objeto vazio.
*/
PROBLEMAS 1
SAÍDA
CONSOLE DE DEPURAÇÃO
TERMINAL
node
bash
determine if they come from your own scripts or plugins.
See https://docs.gradle.org/7.5.1/userguide/command_line_interface.html#sec:command_line_warnings
BUILD SUCCESSFUL in 40s
38 actionable tasks: 2 executed, 36 up-to-date
info Connecting to the development server...
info Starting the app on "emulator-5554"...
Starting: Intent { cmp=com.componentes/.MainActivity }
PAGE 3 OF 4
341 WORDS
PORTUGUESE (BRAZI
master*
Ln 72, Col 2 (2771 selecionado)
Espaços: 4
UTF-8
CRLF
JavaScript
Go Live
Digite aqui para pesquisar
28°C
14:16
30/11/2022