

Usabilidade, Desenvolvimento Web, Mobile e Jogos

APP Câmera



Prof. Juliano Gaspar





Prof. Dr. Juliano Gaspar

Email: julianogaspar@gmail.com

Formação

- **Cientista da Computação** pela UNIVALI (SC)
- **Mestre em Informática Médica** pela UP (Portugal)
- **Doutor em Saúde Digital** pela UFMG
- **Pós-doutor em Tec. para Educação em Saúde** pela UFMG

Educação

Professor Convidado do Departamento de GOB da UFMG

- o Introdução à Pesquisa Científica II
- o Informática Médica
- o Informação em Decisão em Saúde

Professor Grupo Ânima: Una, Unibh e Faseh

- o Projetos e Processos em TI
- o Usabilidade, desenvolvimento Web, Mobile e Jogos
- o Pós-graduação em Saúde Digital e Telemedicina

Professor da Facisa BH

- o Linguagens de Programação (Linguagem C)
- o Programação orientada à objetos (Python)
- o Programação para dispositivos móveis
- o Programação para Web

Professor da IESLA

- o Gestão das Tecnologias da Informação

Inovação, Pesquisa, Desenvolvimento e Extensão

- o Vice-coordenador do núcleo de pesquisa em Informática Aplicada à Saúde da UFMG
- o Membro do CINTESIS - Universidade Porto - Portugal
- o Membro da SBIS - Sociedade Brasileira de Informática em Saúde
- o Presidente do Comitê Científico do CBIS 2022
- o Revisor de revistas científicas

Linhas de pesquisa e projetos

- o Detecção da prematuridade através da interação entre a luz e a pele neonatal
- o Sistema de Informação para Atenção Materno Infantil
- o Eliminando a morte materna: uma resposta a esse desafio no seu bolso.

Programas e projetos de extensão

- o Informática e Saúde
- o Prevenção da COVID-19 em APP
- o Meu Pré-natal (APP)
- o Projeto Educação Continuada em Informática



Usabilidade, Desenvolvimento Web, Mobile e Jogos

Prof. Juliano Gaspar

Contatos

Email

julianogaspar@gmail.com

Instagram

[@prof.julianogaspar](https://www.instagram.com/prof.julianogaspar)

Twitter

<https://twitter.com/JulianoGaspar21>

Currículo Lattes

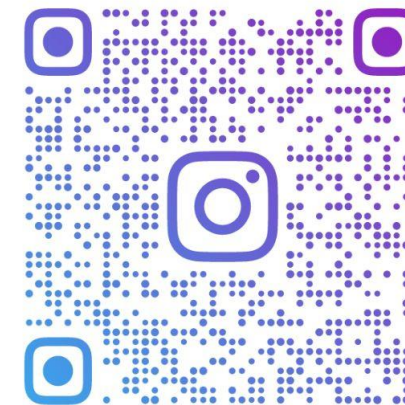
<http://lattes.cnpq.br/3926707936198077>

Orcid ID

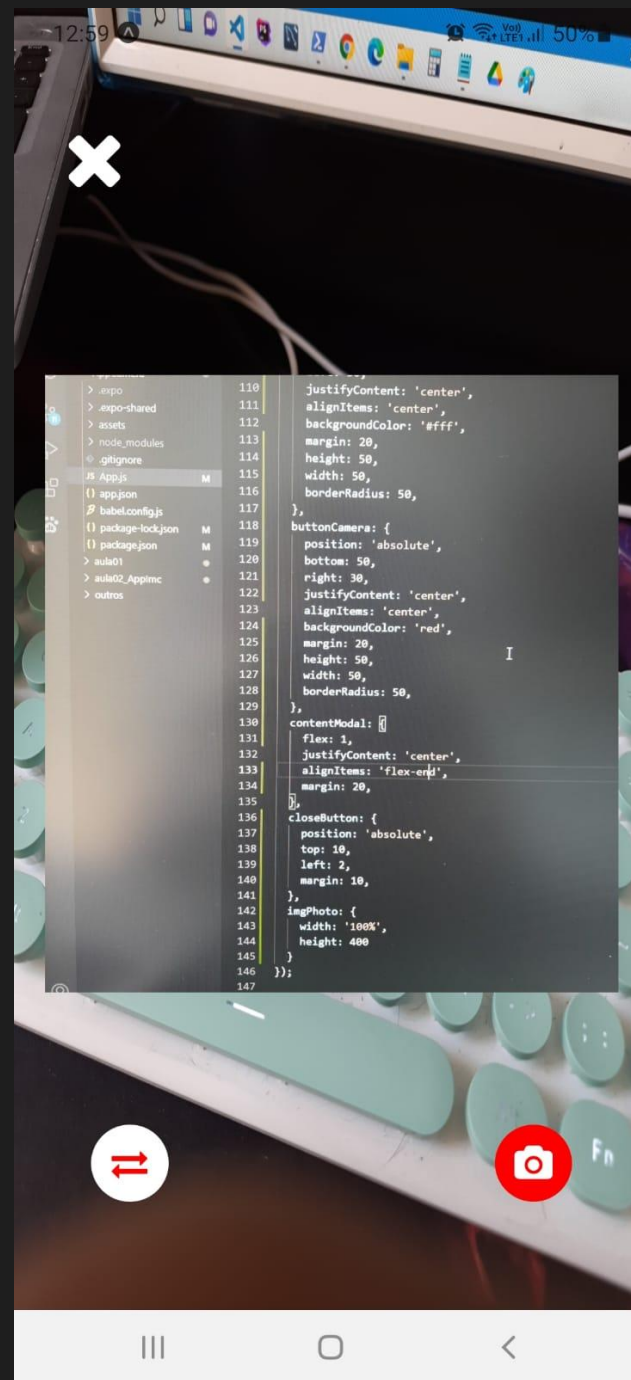
<https://orcid.org/0000-0003-0670-9021>

Research Gate

https://www.researchgate.net/profile/Juliano_Gaspar

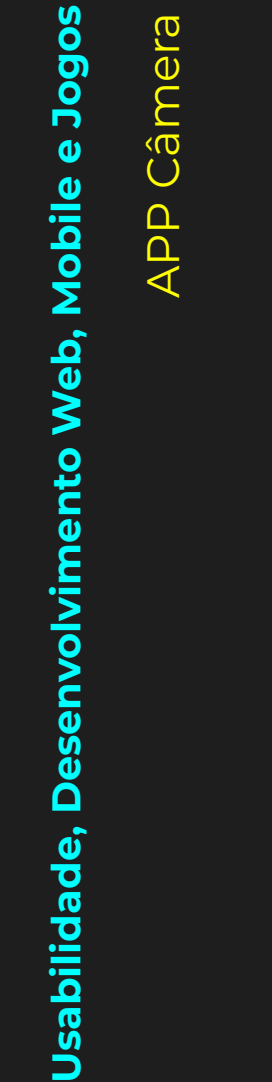


PROF.JULIANOGASPAR



Tópicos abordados:

- APIs
- Camera
- Contatos
- Sensores
- FontAwesome
- Modal
- useEffect
- useRef
- Permissões
- entre outros




```
- cd AppCamera
- npm start # you can open iOS, Android, or web from here, or run them directly
- npm run android
- npm run ios # requires an iOS device or macOS for access to an iOS simulator
- npm run web
PS C:\Users\Juliano\Documents\ProjetosReact\Testes> cd .\AppCamera\
```

```
PS C:\Users\Juliano\Documents\ProjetosReact\Testes\AppCamera> npm start
```

Prof. Juliano Gaspar



 Expo

Get Started

Guides

EAS

API Reference

Feature Preview

Q Search

Asset

AsyncStorage

Audio

AuthSession

AV

BackgroundFetch

BarcodeScanner

Barometer

Battery

BlurView

Branch

Brightness

Calendar

• Camera

captureRef

Cellular

Checkbox

Camera

`expo-camera` provides a React component that renders back camera. The camera's parameters like zoom, are adjustable. With the use of `Camera`, one can also then saved to the app's cache. Moreover, the component and bar codes appearing in the preview. Run the [example](#) features working together!

Installation

```
$ expo install expo-camera
```

Platform Compatibility

Android Device	Android Emulator	iOS Device	iOS Simulator	Web
✓	✗	✓	✗	✓

💡 Android devices can use one of two available Camera APIs: you can opt-in to using `Camera2` with the `useCamera2Api` prop.

<https://docs.expo.dev/versions/v44.0.0/sdk/camera/>



```
import React, { useState, useEffect } from 'react';
import { StyleSheet, Text, View, TouchableOpacity } from 'react-native';
import { Camera } from 'expo-camera';

export default function App() {
  const [hasPermission, setHasPermission] = useState(null);
  const [type, setType] = useState(Camera.Constants.Type.back);

  useEffect(() => {
    (async () => {
      const { status } = await Camera.requestCameraPermissionsAsync();
      setHasPermission(status === 'granted');
    })();
  }, []);

  if (hasPermission === null) {
    return <View />;
  }
  if (hasPermission === false) {
    return <Text>No access to camera</Text>;
  }
}
```



```
return (  
  <View style={styles.container}>  
    <Camera style={styles.camera} type={type}>  
      <View style={styles.buttonContainer}>  
        <TouchableOpacity  
          style={styles.button}  
          onPress={() => {  
            setType(  
              type === Camera.Constants.Type.back  
                ? Camera.Constants.Type.front  
                : Camera.Constants.Type.back  
            );  
          }}>  
          <Text style={styles.text}> Flip </Text>  
        </TouchableOpacity>  
      </View>  
    </Camera>  
  </View>  
}  
  
const styles = StyleSheet.create({ ... });
```




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

export default function App() {
  return (
    <SafeAreaView style={styles.container}>

    </SafeAreaView>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
  },
});
```





```
import React, { useState, useEffect } from 'react';  
import { StyleSheet, Text, View, SafeAreaView } from 'react-native';  
import { Camera } from 'expo-camera';
```

```
export default function App() {  
  
  const [type, setType] = useState(Camera.Constants.Type.back)  
  
  return (  
    <SafeAreaView style={styles.container}>  
      <Camera  
        style={styles.camera}  
        type={type}  
      >  
    </Camera>  
    </SafeAreaView>  
  );  
}  
  
const styles = StyleSheet.create({  
  container: {  
    flex: 1,  
    justifyContent: 'center',  
  },  
  camera: {  
    width: '100%',  
    height: '100%',  
  },  
});
```



```
import React, { useState, useEffect } from 'react';
import { StyleSheet, Text, View, SafeAreaView } from 'react-native';
import { Camera } from 'expo-camera';
```

```
export default function App() {
```

```
  const [type, setType] = useState(Camera.Constants.Type.back)
  const [hasPermission, setHasPermission] = useState(null)
```

constante para armazenar a permissão ou não de acesso à câmera

```
  useEffect ( () => {
  }, [])
```

Estrutura básica para criar um useEffect

- É chamado sempre que o componente inicializa e executa a função que está dentro.

```
  return (
    <SafeAreaView style={styles.container}>
      <Camera
        style={styles.camera}
        type={type}
      >
    </Camera>
    </SafeAreaView>
  );
}
```




```
export default function App() {
```

```
  const [type, setType] = useState(Camera.Constants.Type.back)
```

```
  const [hasPermission, setHasPermission] = useState(null)
```

```
  useEffect(() => {
```

```
    (async () => {
```

```
      const { status } = await Camera.requestCameraPermissionsAsync()
```

```
      setHasPermission(status === 'granted');
```

```
    })();
```

```
  }, [])
```

```
  if (hasPermission === null) {
```

```
    return <View/>
```

```
  }
```

```
  if (hasPermission === false) {
```

```
    return <Text>Acesso negado!</Text>
```

```
  }
```

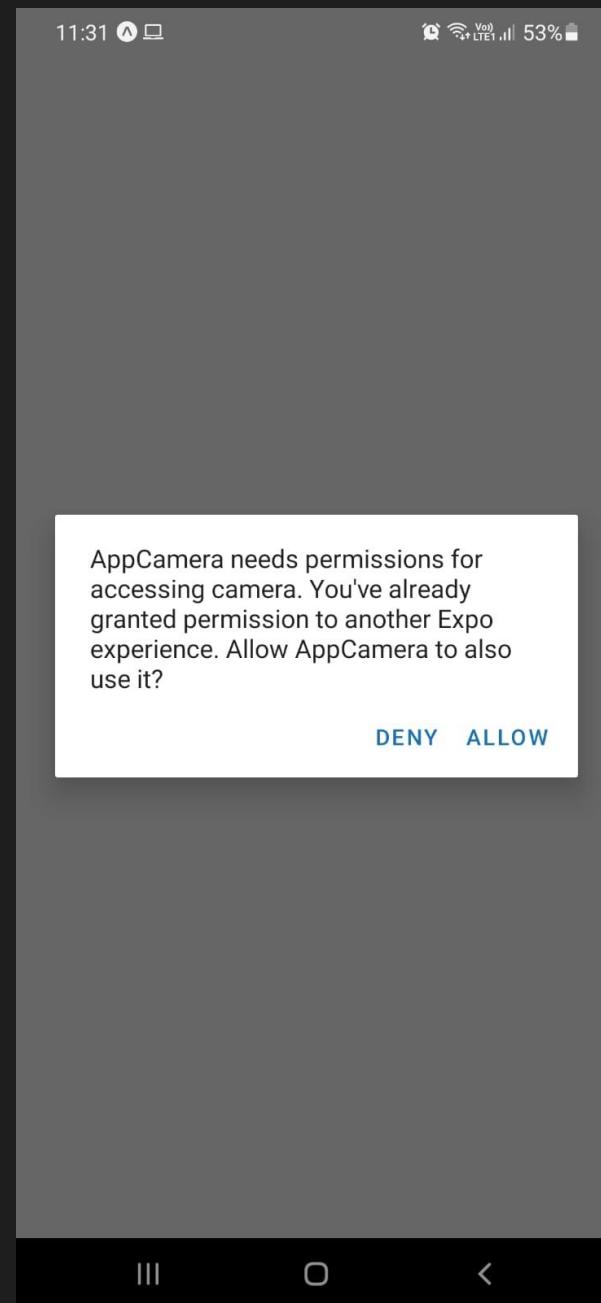
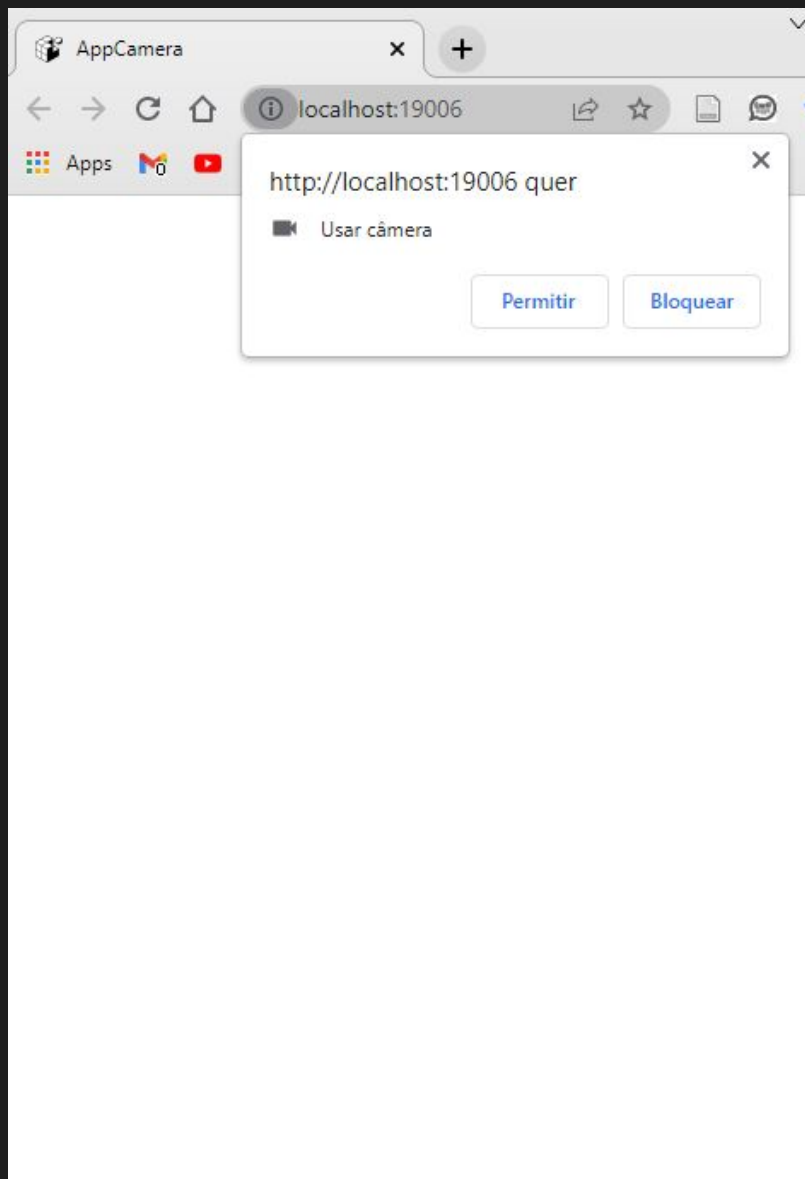


Função assíncrona que faz o pedido de permissão e retorna a resposta de usuário.



Valida se tem permissão, ou se ela foi negada.



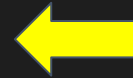




```
import React, { useState, useEffect } from 'react';
import { StyleSheet, Text, View, SafeAreaView, TouchableOpacity } from 'react-native';
import { Camera } from 'expo-camera';
import { FontAwesome } from '@expo/vector-icons';
```



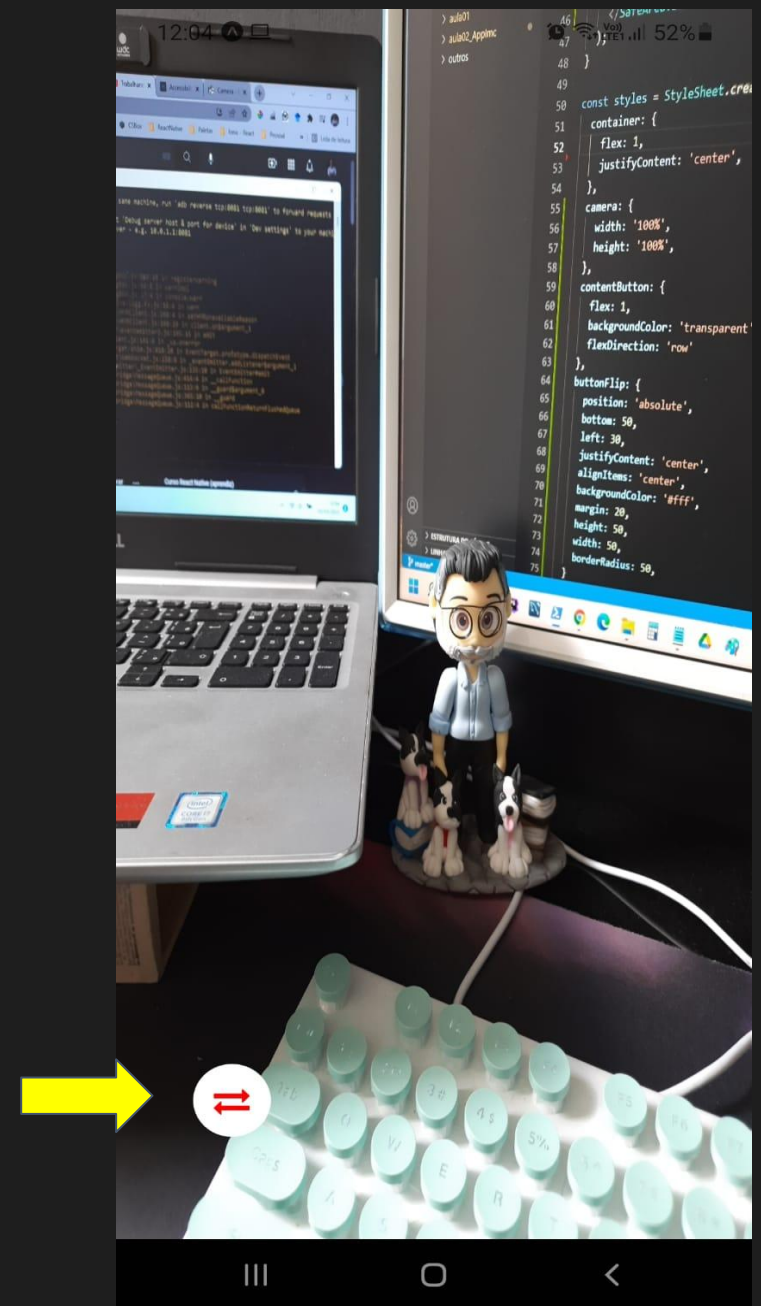
```
...
return (
  <SafeAreaView style={styles.container}>
    <Camera
      style={styles.camera}
      type={type}
    >
      <View style={styles.contentButton}>
        <TouchableOpacity
          style={styles.buttonFlip}
          onPress={() => {
            setType(type === Camera.Constants.Type.back ?
              Camera.Constants.Type.front
              : Camera.Constants.Type.back)
          }}
        >
          <FontAwesome name='exchange' size={23} color='red'></FontAwesome>
        </TouchableOpacity>
      </View>
    </Camera>
  </SafeAreaView>
);
}
```



Criando um botão para trocar a camera trazeira e frontal.



```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
  },
  camera: {
    width: '100%',
    height: '100%',
  },
  contentButton: {
    flex: 1,
    backgroundColor: 'transparent',
    flexDirection: 'row'
  },
  buttonFlip: {
    position: 'absolute',
    bottom: 50,
    left: 30,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#fff',
    margin: 20,
    height: 50,
    width: 50,
    borderRadius: 50,
  }
});
```





```
import React, { useState, useEffect, useRef } from 'react';
import { StyleSheet, Text, View, SafeAreaView, TouchableOpacity } from 'react-native';
import { Camera } from 'expo-camera';
import { FontAwesome } from '@expo/vector-icons';
```

```
export default function App() {

  const [type, setType] = useState(Camera.Constants.Type.back)
  const [hasPermission, setHasPermission] = useState(null)

  const camRef = useRef(null)
  const [capturePhoto, setCapturePhoto] = useState(null)

  async function takePicture() {
    if(camRef){
      const data = await camRef.current.takePictureAsync();
      setCapturePhoto(data.uri)
    }
  }
}
```



- useRef usado para guardar referencias / endereços.
- camRef vai conter o endereço da foto tirada.
- setCapturePhoto vai armazenar no estado o endereço de camRef.
- takePicture vai tirar a foto e salvar a uri (endereço da foto)

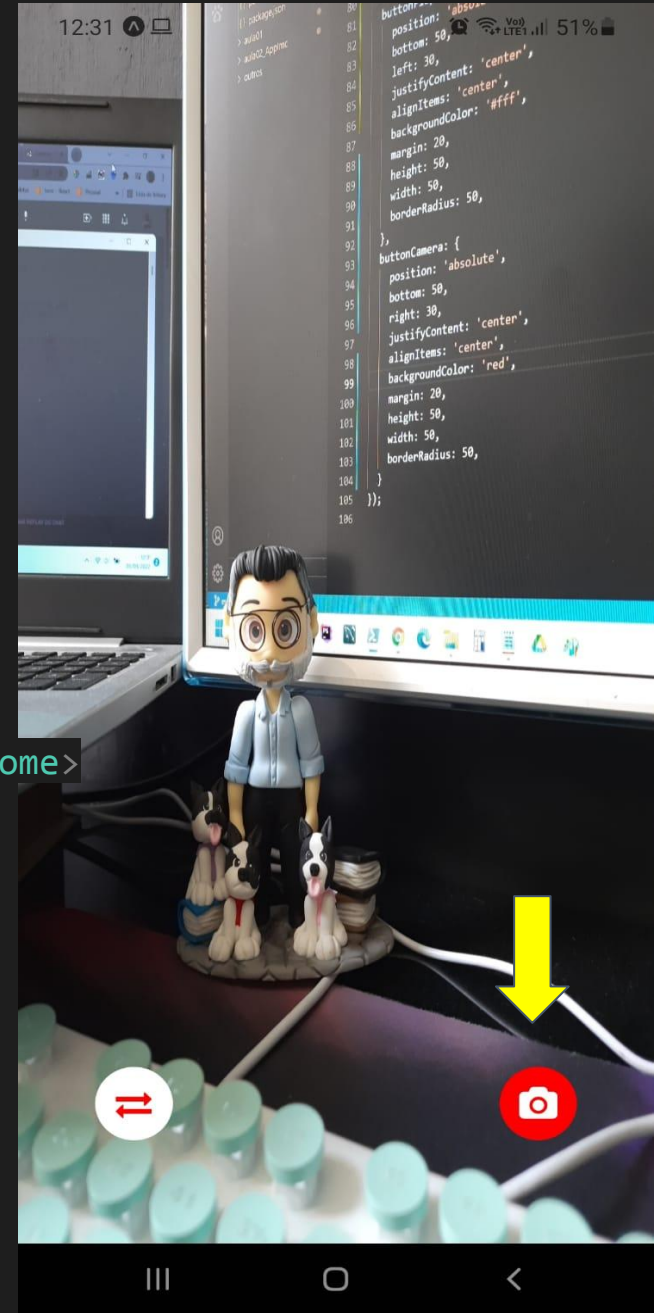
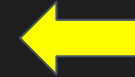


```
return (
  <SafeAreaView style={styles.container}>
    <Camera
      style={styles.camera}
      type={type}
    >
      <View style={styles.contentButton}>
        <TouchableOpacity
          ...
        ></FontAwesome>
        </TouchableOpacity>

        <TouchableOpacity
          style={styles.buttonCamera}
          onPress={takePicture}
        >
          <FontAwesome name='camera' size={23} color='#fff'></FontAwesome>
        </TouchableOpacity>
      </View>
    </Camera>
  </SafeAreaView>
);
```

← Criando um segundo botão para tirar a foto.

```
buttonCamera: {
  position: 'absolute',
  bottom: 50,
  right: 30,
  justifyContent: 'center',
  alignItems: 'center',
  backgroundColor: 'red',
  margin: 20,
  height: 50,
  width: 50,
  borderRadius: 50,
}
```



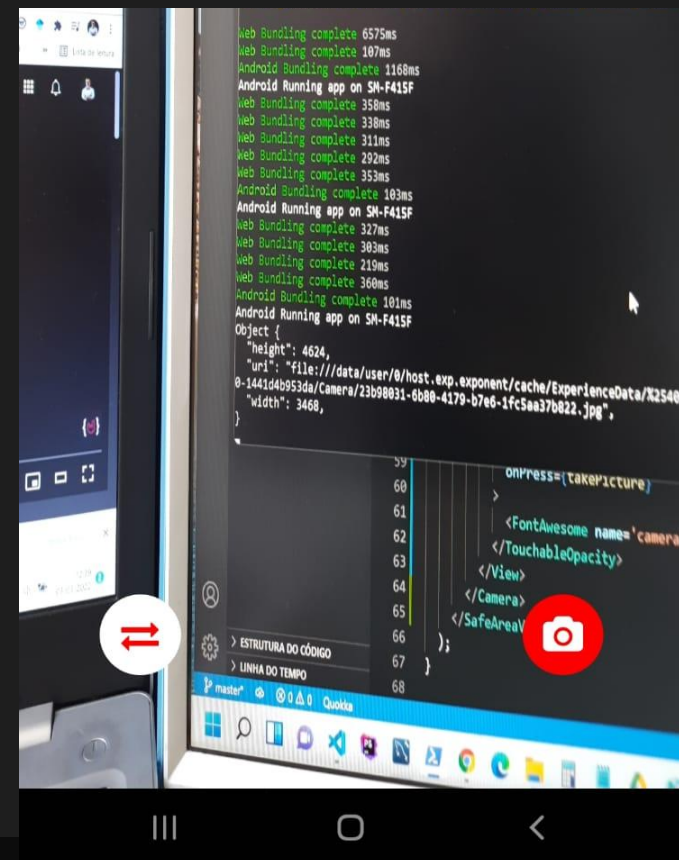


```
async function takePicture() {
  if (camRef) {
    const data = await camRef.current.takePictureAsync();
    setCapturePhoto(data.uri)
    console.log(data)
  }
}
```

```
return (
  <SafeAreaView style={styles.container}>
    <Camera
      style={styles.camera}
      type={type}
      ref={camRef}
    >
```

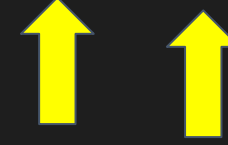
- console.log
- em camera referencia o ref com o camRef criado.

```
Web Bundling complete 219ms
Web Bundling complete 360ms
Android Bundling complete 101ms
Android Running app on SM-F415F
Object {
  "height": 4624,
  "uri": "file:///data/user/0/host.exp.exponent/cache/ExperienceData/%2540anonymous%252FAAppCamera-f066cf94-ae80-4b1e-87c0-1441d4b953da/Camera/23b98031-6b80-4179-b7e6-1fc5aa37b822.jpg",
  "width": 3468,
}
```





```
import React, { useState, useEffect, useRef } from 'react';
import { StyleSheet, Text, View, SafeAreaView, TouchableOpacity, Modal, Image } from 'react-native';
import { Camera } from 'expo-camera';
import { FontAwesome } from '@expo/vector-icons';
```



```
export default function App() {

  const [type, setType] = useState(Camera.Constants.Type.back)
  const [hasPermission, setHasPermission] = useState(null)
  const camRef = useRef(null)
  const [capturePhoto, setCapturePhoto] = useState(null)
  const [open, setOpen] = useState(false)

  async function takePicture() {
    if (camRef) {
      const data = await camRef.current.takePictureAsync();
      setCapturePhoto(data.uri)
      // console.log(data)
      setOpen(true)
    }
  }
}
```

- Vamos criar um modal para aparecer a foto tirada com um botão de fechar.
- Vai aparecer por cima da view da camera.



```
</Camera>

{capturePhoto && (
  <Modal
    animationType='slide'
    transparent={true}
    visible={open}
  >
    <View style={styles.contentModal}>
      <TouchableOpacity
        style={styles.closeButton}
        onPress={() => { setOpen(false) }}
      >
        <FontAwesome name='close' size={50} color='#fff'></FontAwesome>
      </TouchableOpacity>

      <Image style={styles.imgPhoto} source={{ uri: capturePhoto }}></Image>
    </View>
  </Modal>
)}
</SafeAreaView>
);
}
```



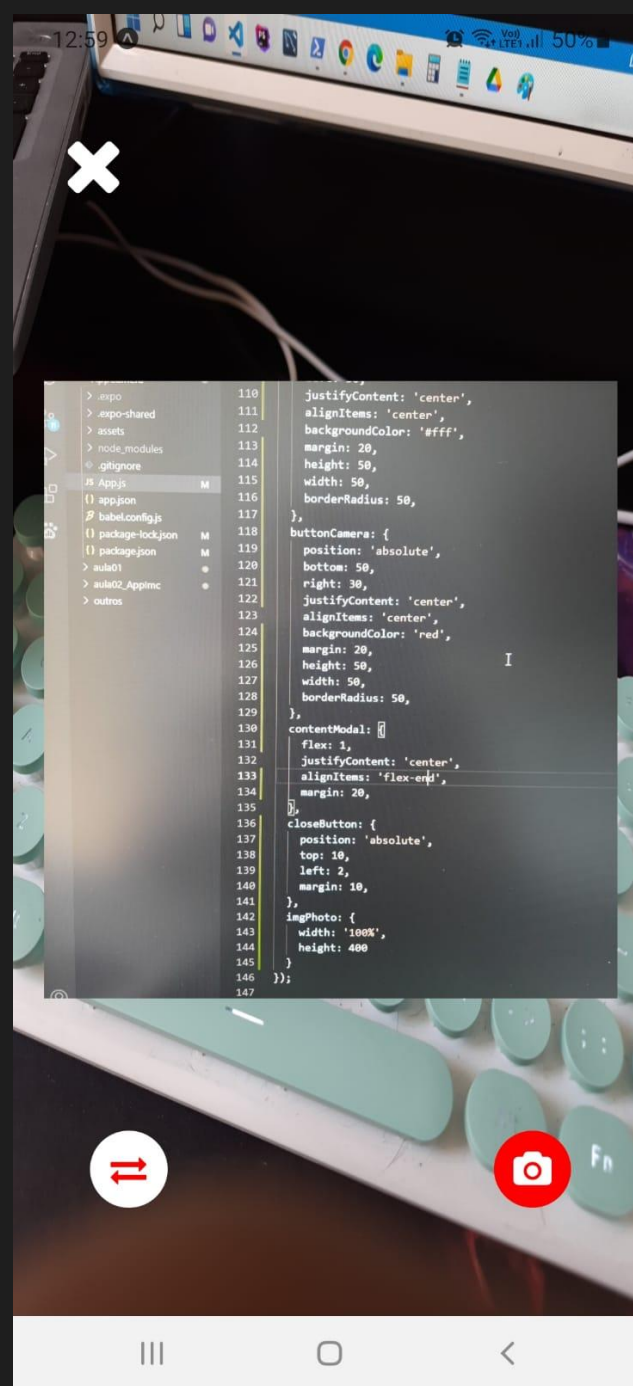
- Só mostra se tiver uma foto (uri) salva.



```

contentModal: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'flex-end',
  margin: 20,
},
closeButton: {
  position: 'absolute',
  top: 10,
  left: 2,
  margin: 10,
},
imgPhoto: {
  width: '100%',
  height: 400
}
});

```



**Usabilidade,
Desenvolvimento Web, Mobile e Jogos**

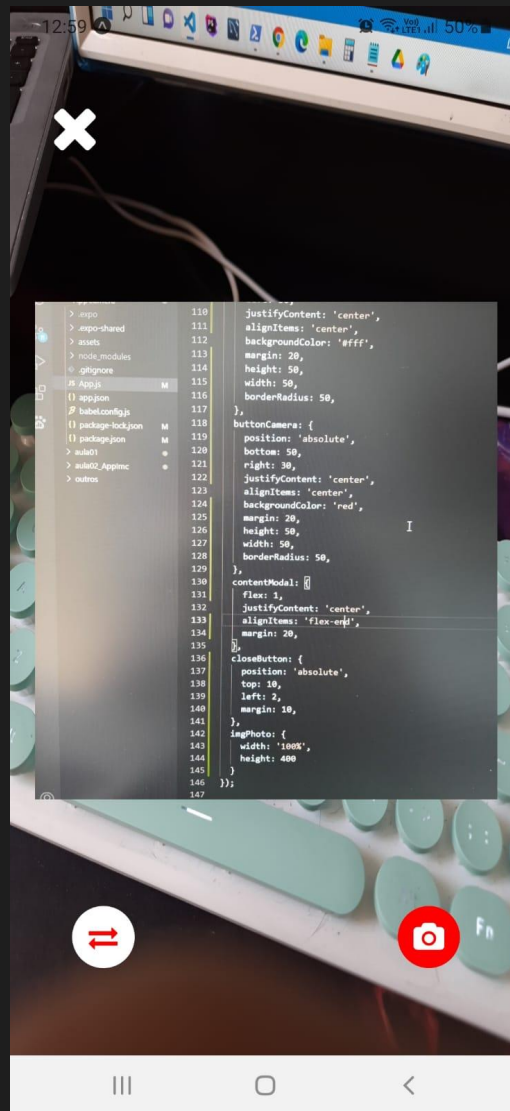
Obrigado!



Prof. Dr. Juliano Gaspar
julianogaspar@gmail.com

<http://lattes.cnpq.br/3926707936198077>

Prof. Juliano Gaspar



Desenvolvimento Mobile

APP Câmera