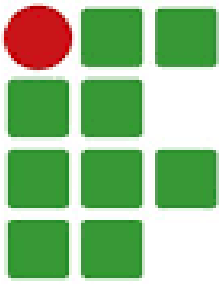


INSTITUTO FEDERAL
São Paulo
Câmpus Cubatão

PJS - TUTORIAL

DISCENTES:

Adriano Júnior de Souza Almeida

Allan Matheus Jesus da Silva

Andre Ferreira Araujo

Ihuá Fillype dos Santos Saraiva

Gabriela de Almeida Santos

Vitor Mesquita Alves

CUBATÃO

2023

Adriano Júnior de Souza Almeida

Allan Matheus Jesus da Silva

Andre Ferreira Araujo

Ihuá Fillype dos Santos Saraiva

Gabriela de Almeida Santos

Vitor Mesquita Alves

PJS - TUTORIAL

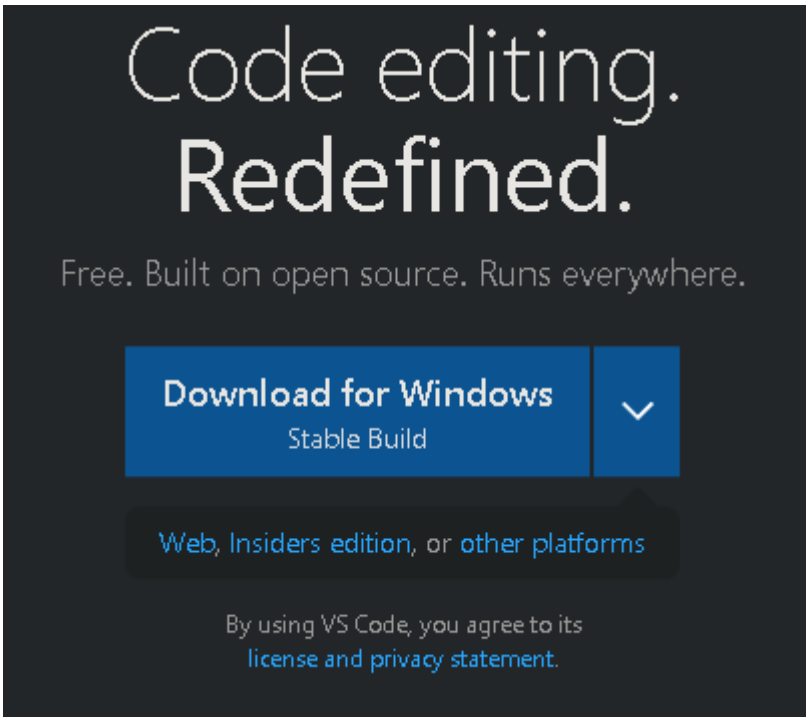
Trabalho de Projeto de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo.
Prof: Maurício Asenjo.

CUBATÃO

2023

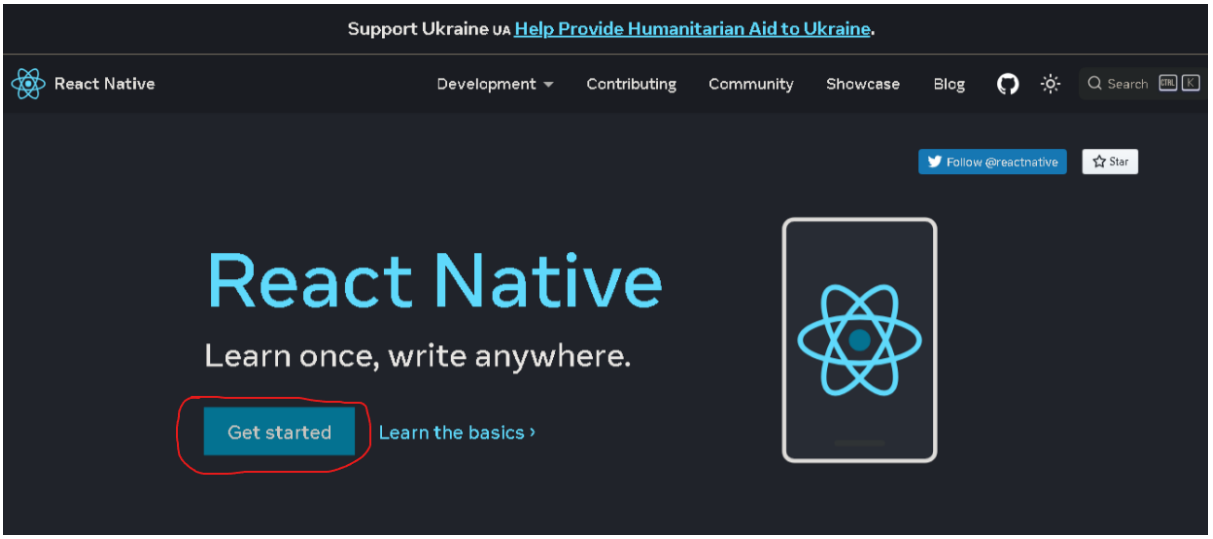
Preparando o Ambiente

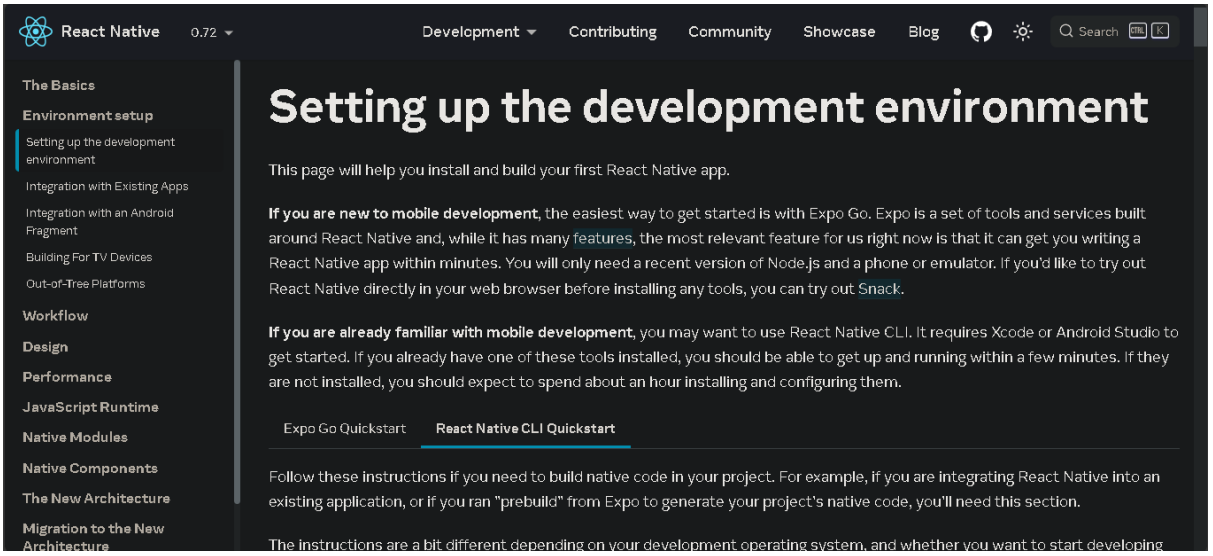
Nosso trabalho precisou usar React Native, segue abaixo o tutorial de como baixar:
Antes de seguirmos é preciso que baixe o Visual Studio Code clicando em Download for Windows:
[Visual Studio Code - Code Editing. Redefined]



1° Acesse o site oficial do React Native.
Link: [React Native · Learn once, write anywhere]

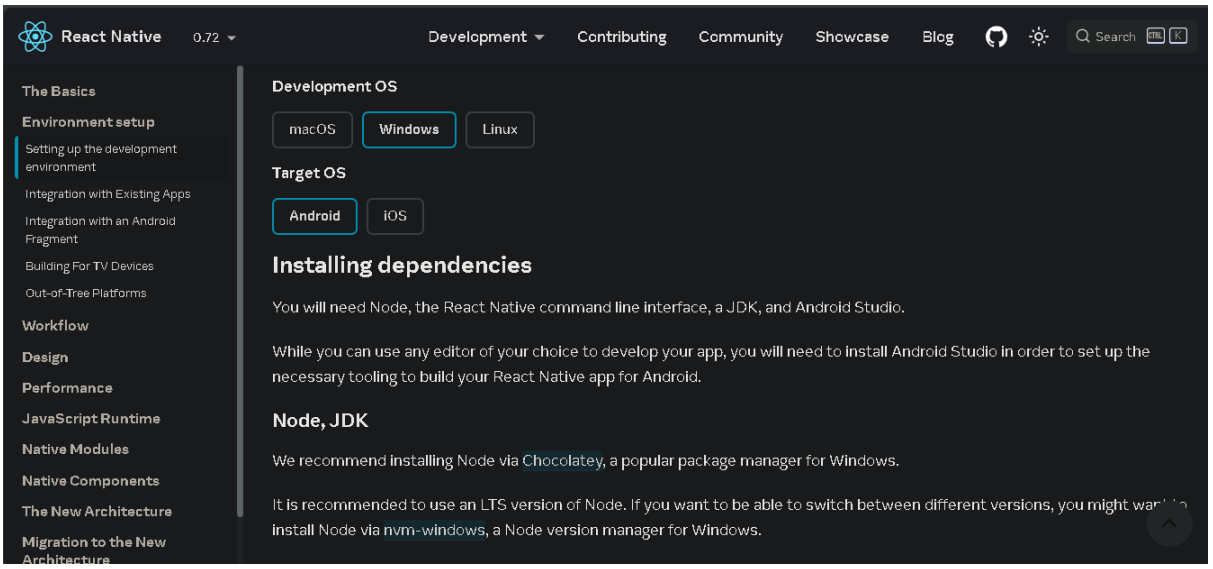
2° Clicando em “Get started”



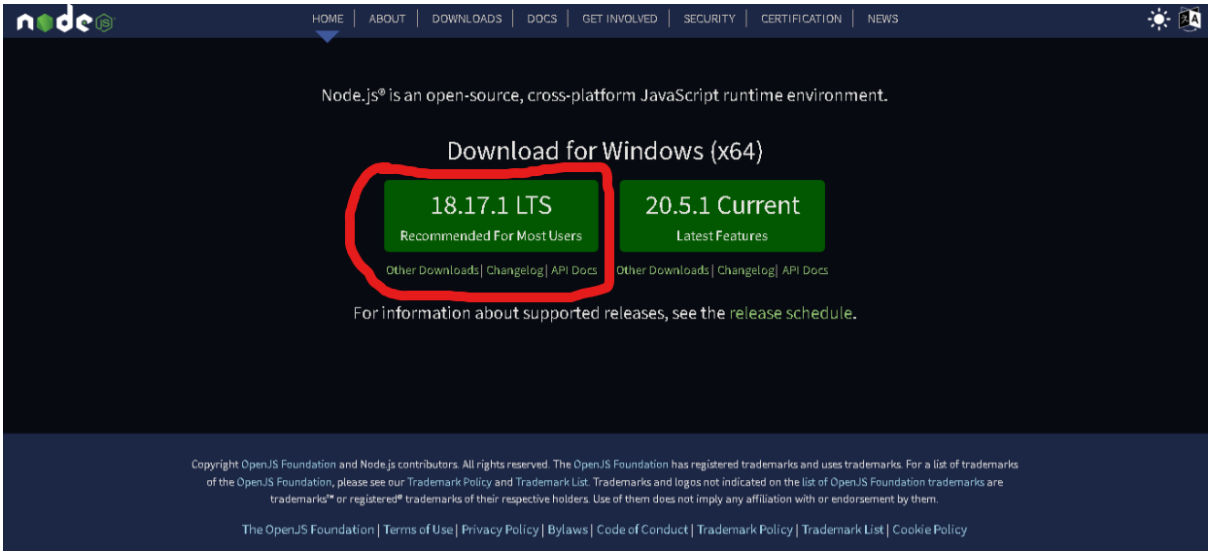


Você se depara com essa tela. NÃO SE ASSUSTE

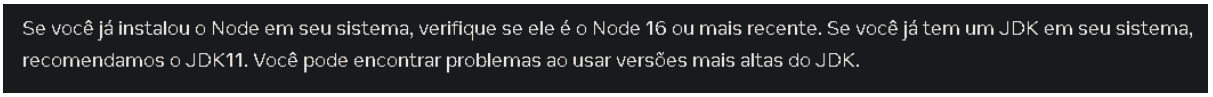
3° Descendo um pouco você vai se deparar com as dependências para ter o React Native, sendo elas: JDK, NodeJS e Android Studio.



4° BAIXANDO AS DEPENDÊNCIAS. Começando com Nodejs, após acessar o site oficial deles você deverá efetuar o download LTS.
link: [Node.js (nodejs.org)]



5° BAIXANDO AS DEPENDÊNCIAS. No site do React Native eles pedem que o JDK seja o 11:

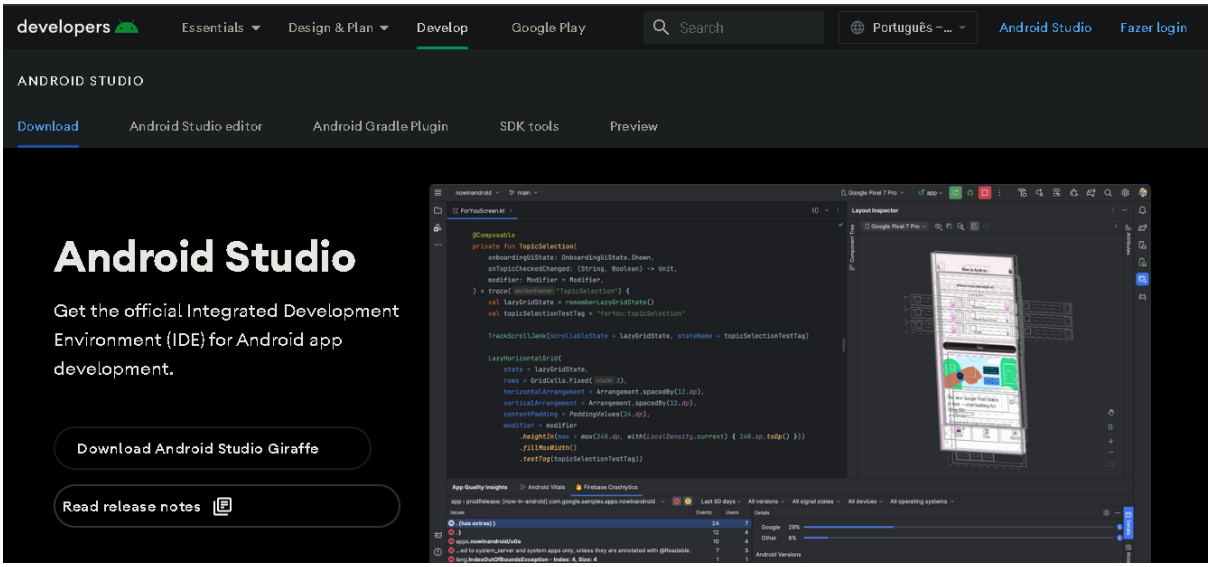


Para baixar o JDK vamos ter que usar o site da oracle: [Java Archive Downloads - Java SE 11 (oracle.com)]

Você vai procurar “Windows x64 Installer”

OBS: Você vai precisar criar uma conta na Oracle para conseguir baixar o JDK.

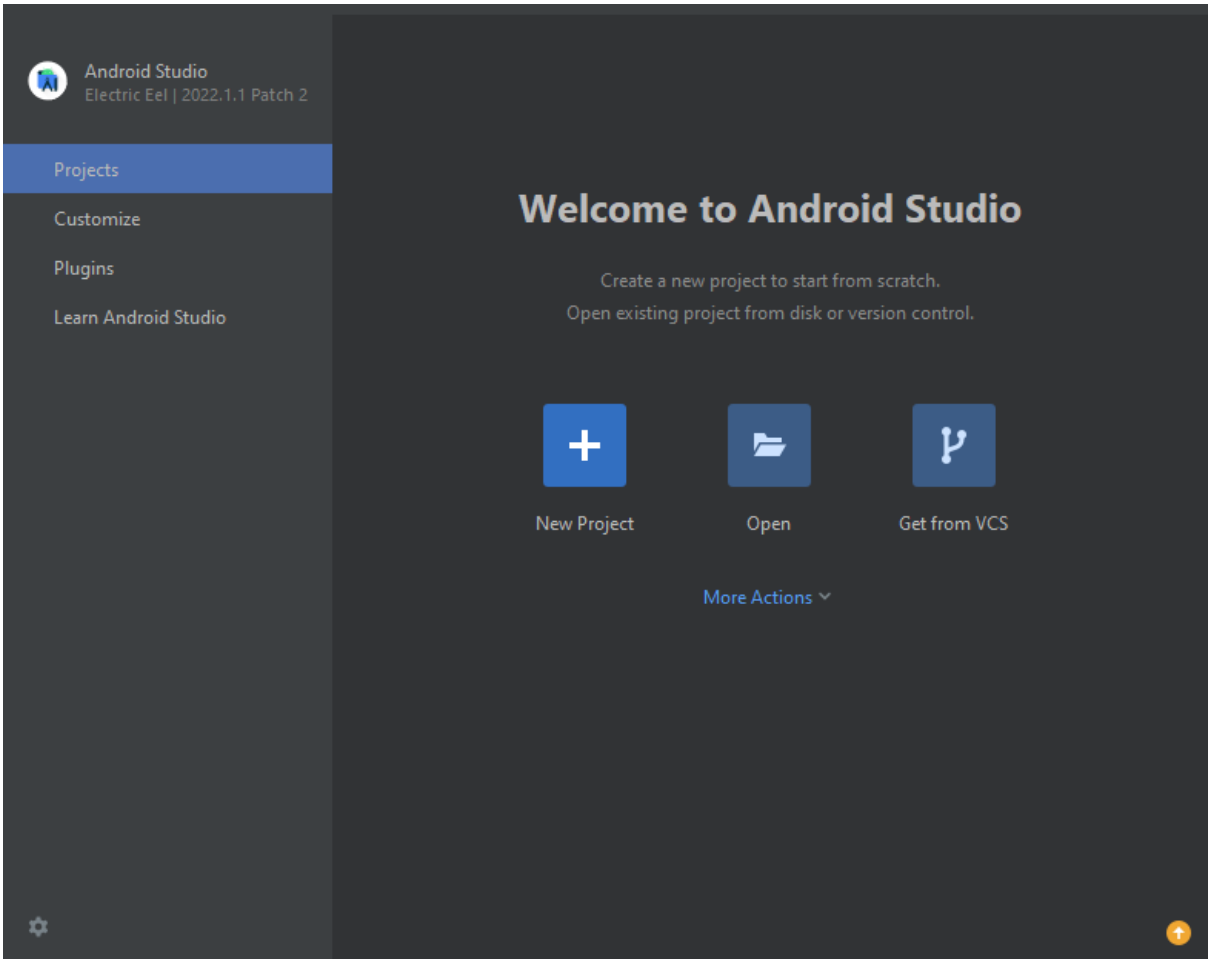
6° BAIXANDO AS DEPENDÊNCIAS(Android Studio). Site oficial do Android Studio: [Download Android Studio & App Tools - Android Developers]



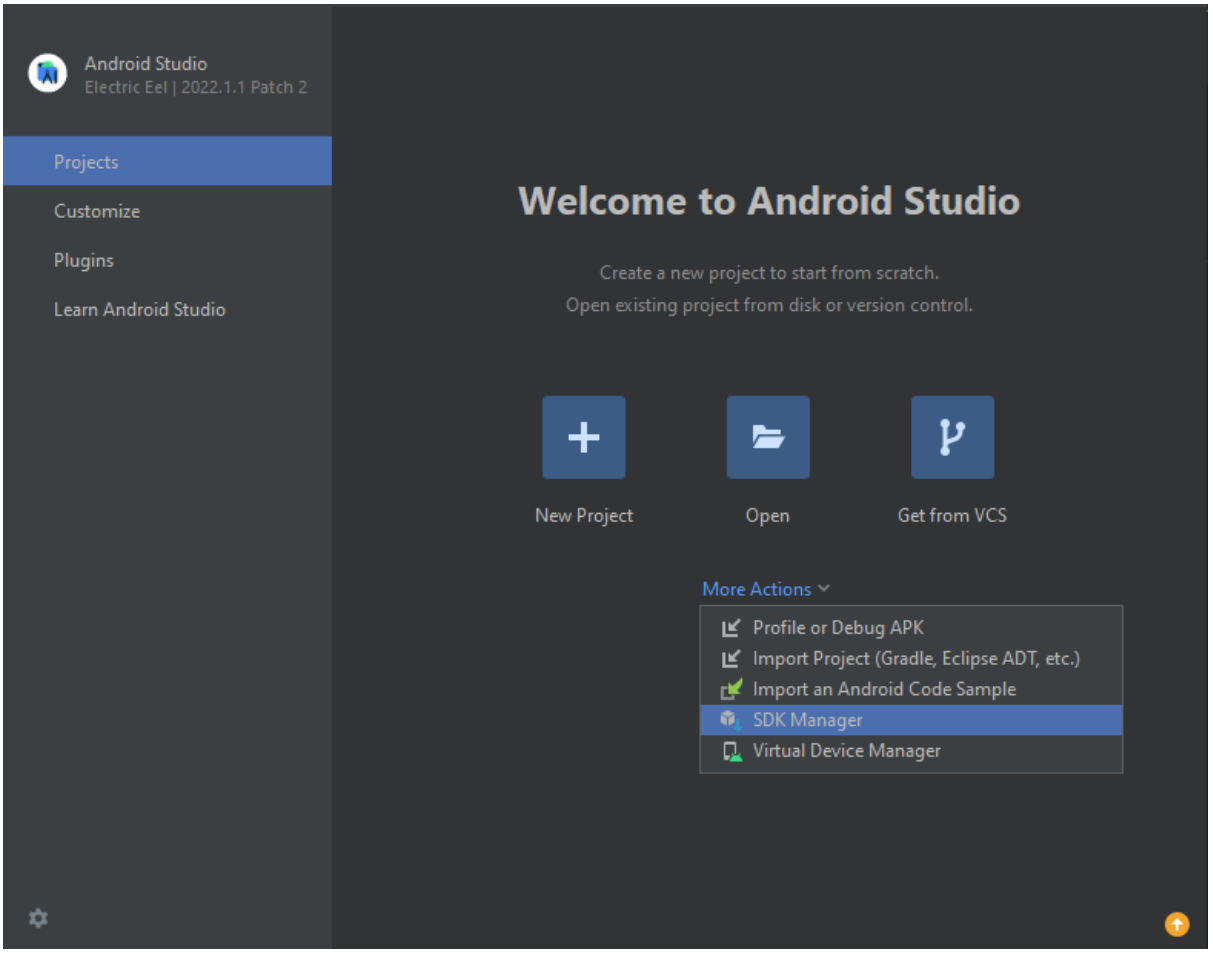
Clicando em “Download Android Studio Giraffe”

CONFIGURANDO:

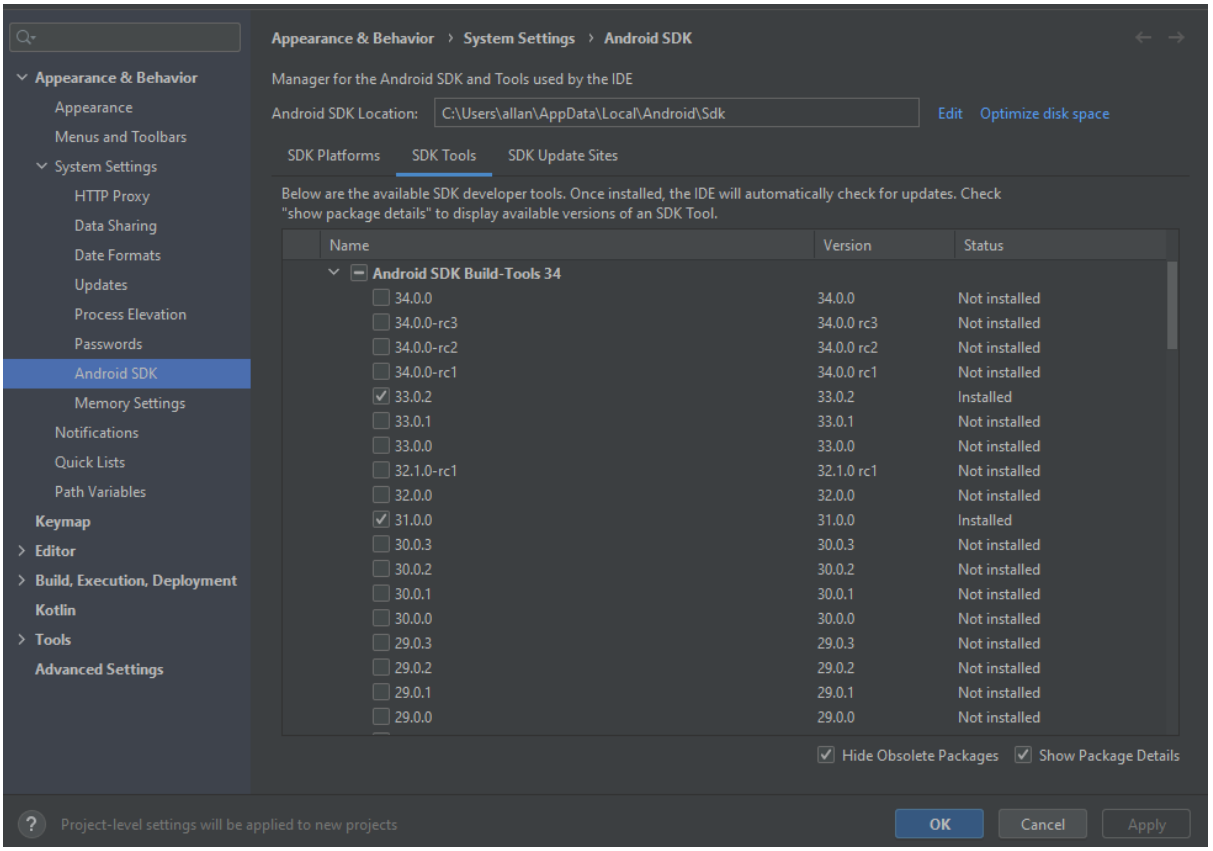
Android Studio: Ao baixar você vai se deparar com essa tela.



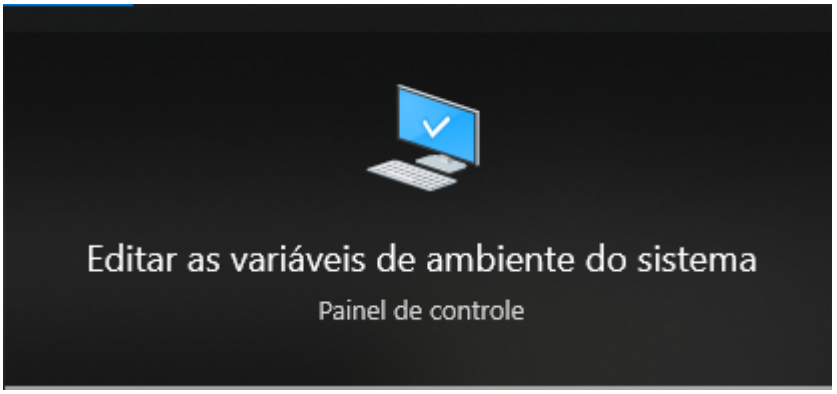
1º passo: clique em “More Actions” e em SDK manager:



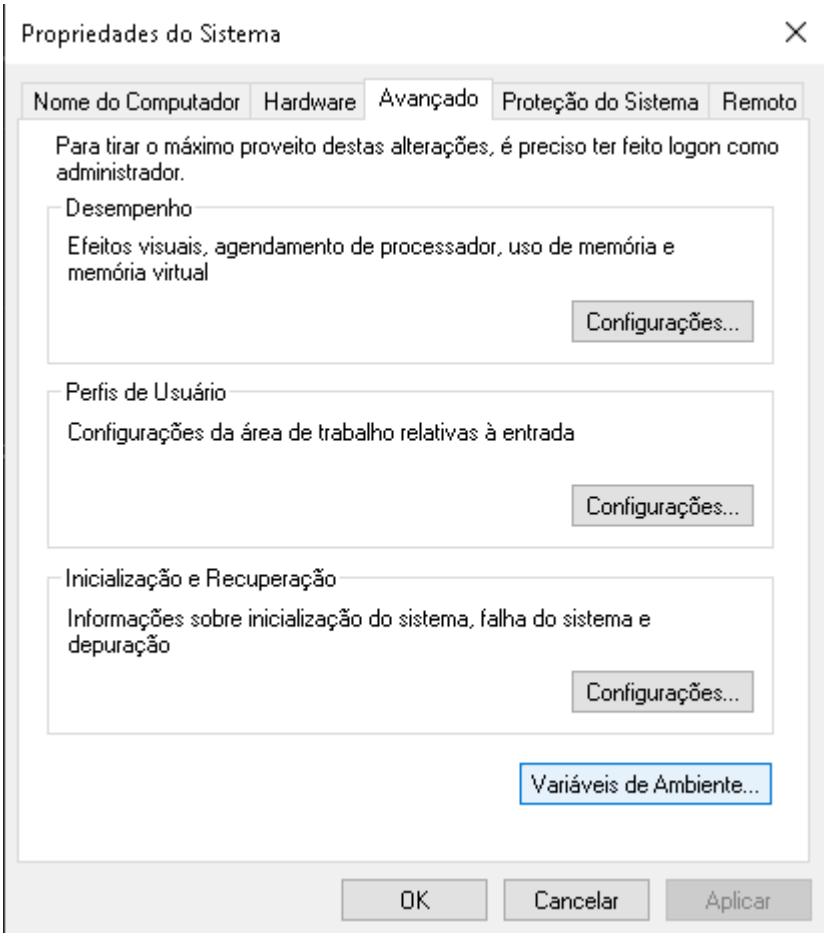
2º Passo: Vá em SDK tools e selecione a versão 31.0.0 e depois clique em apply.



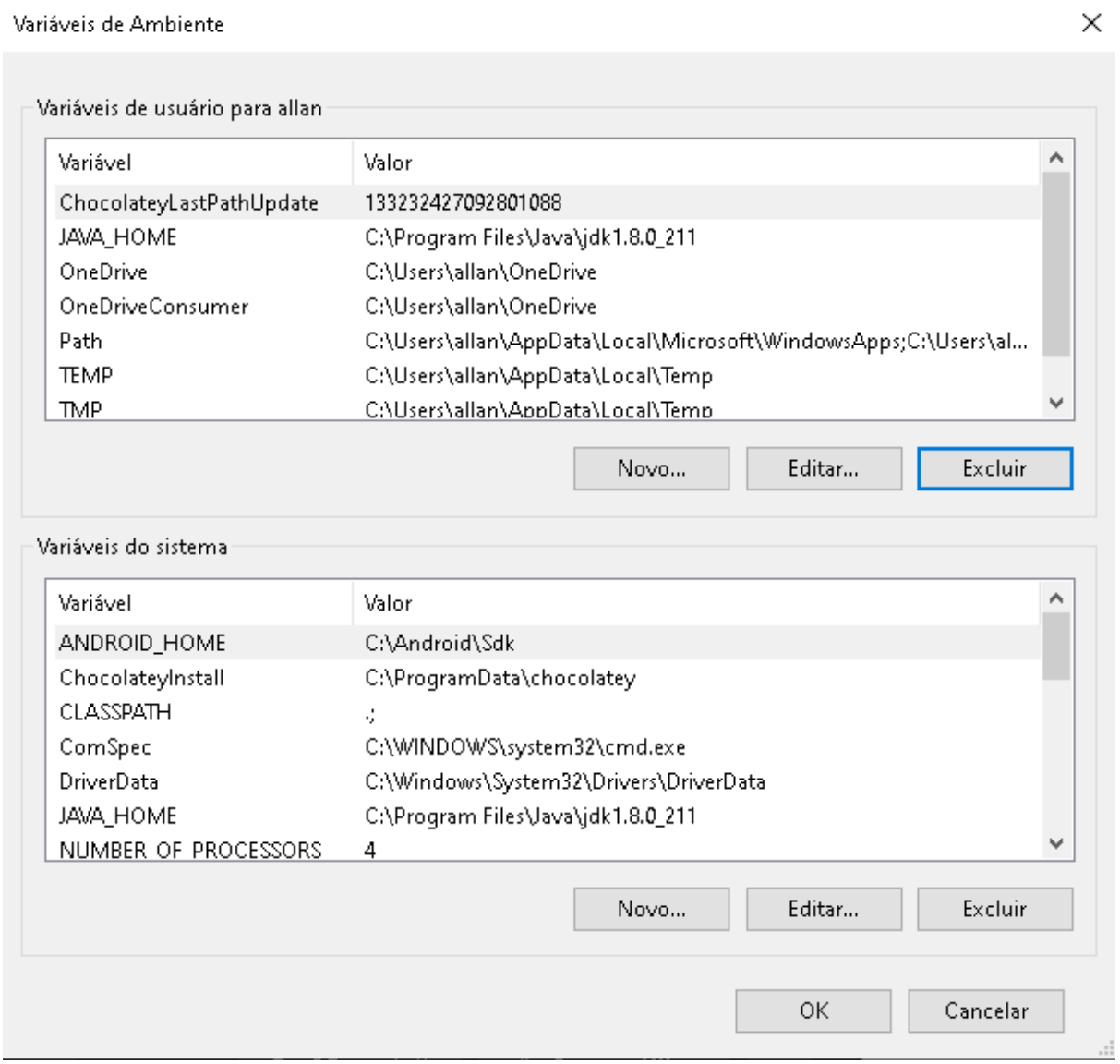
3º Passo: Clique no botão windows e pesquise por “Editar variáveis de ambiente do sistema”



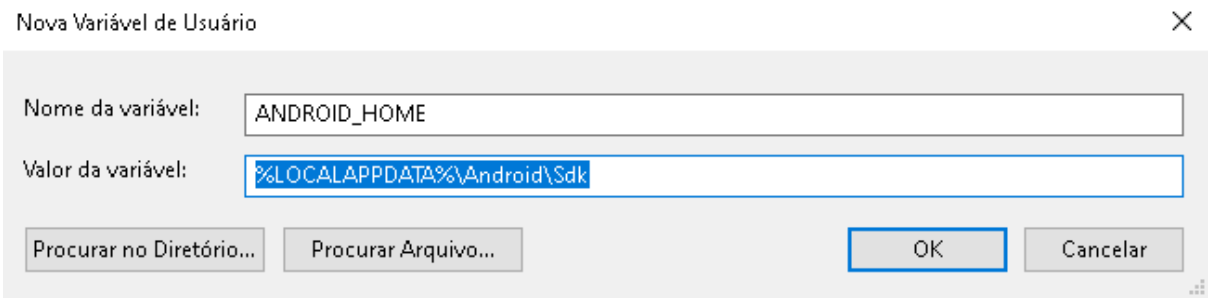
4º Passo: Você vai se deparar com essa tela, clique em “Variáveis de ambiente”



5º Passo: Nesta tela clique em Novo

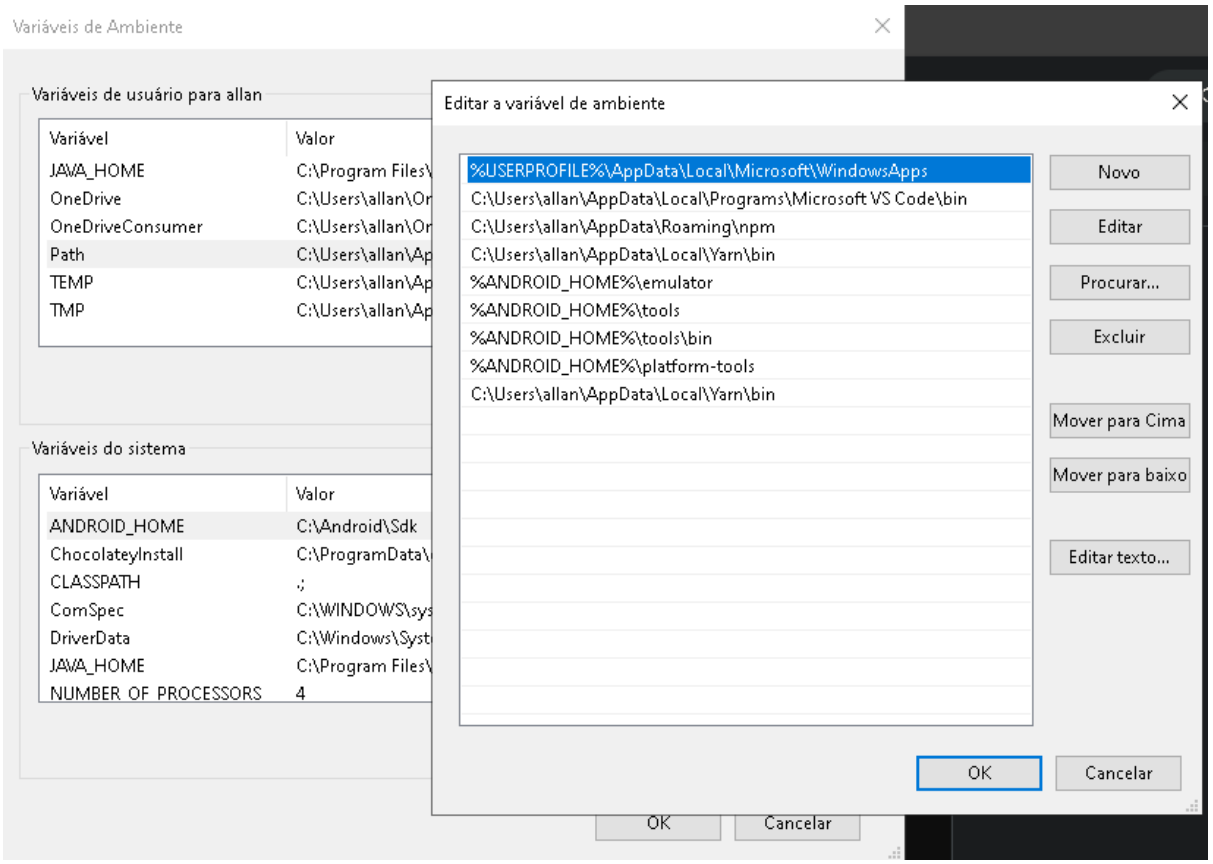


6º Passo: Escreva “ANDROID_HOME” em nome da variável e coloque %LOCALAPPDATA%\Android\Sdk no valor da variável.



e clique em OK

7ºPasso: Ainda na tela procure por “patch” e clique duas vezes .

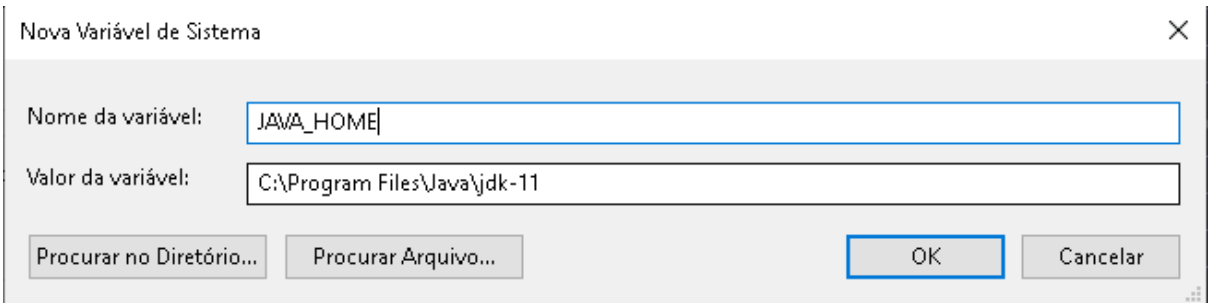


vai aparecer essa tela da direita.

8º Passo: Clique em novo e adicione esse local %LOCALAPPDATA%\Android\Sdk\platform-tools e clique em ok.

9º Passo para finalizar essa parte vamos precisar encontrar o endereço da pasta jdk, para isso é só abrir o explorador de arquivos, C:, meus programas, java e JDK11. Copie o endereço.

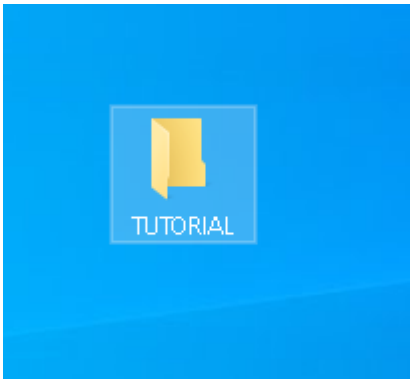
10º Passo: Vá em Variável do sistema e clique em novo e adicione em nome “ JAVA_HOME” e no valor da variável adicione o endereço da pasta JDK.



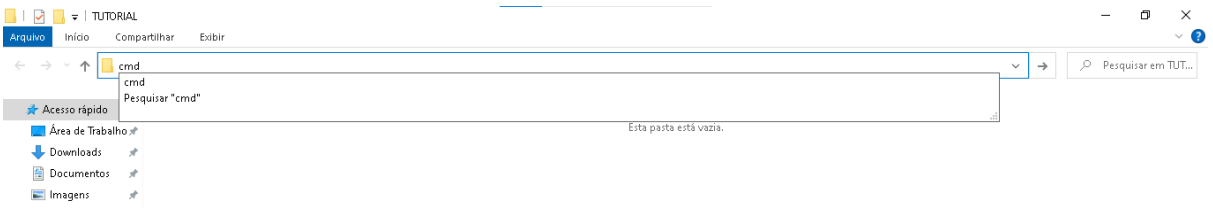
OBS: reinicie o computador para que o computador consiga processar as configurações.

USANDO O VISUAL STUDIO CODE:

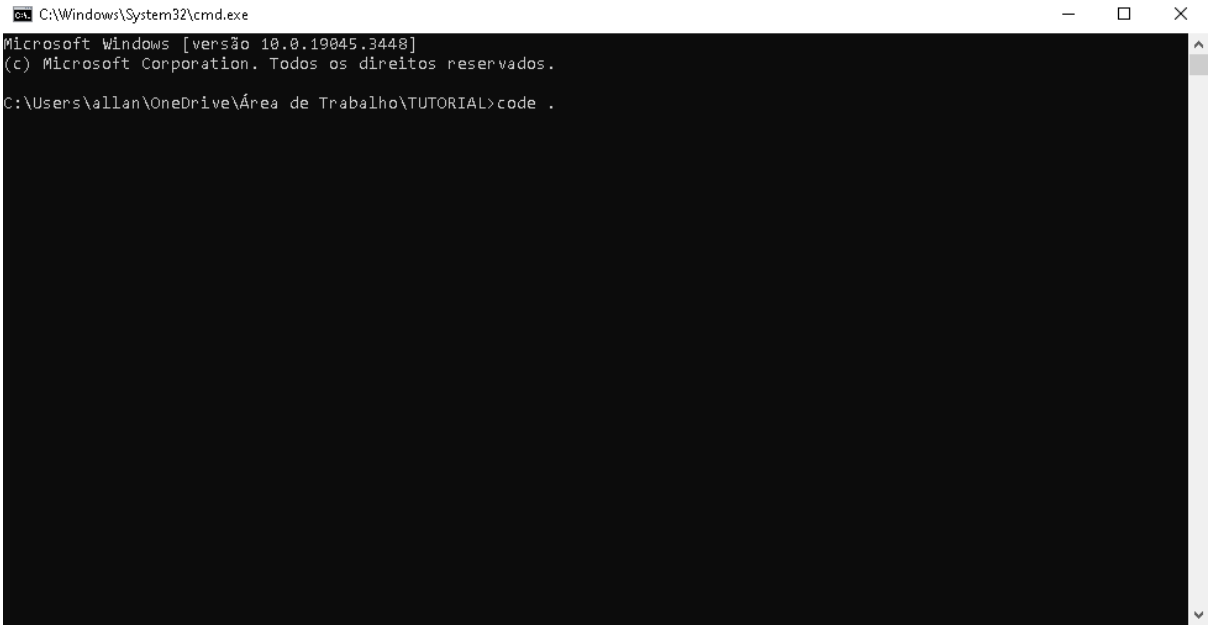
1º crie uma pasta em sua área de trabalho com o nome do projeto:



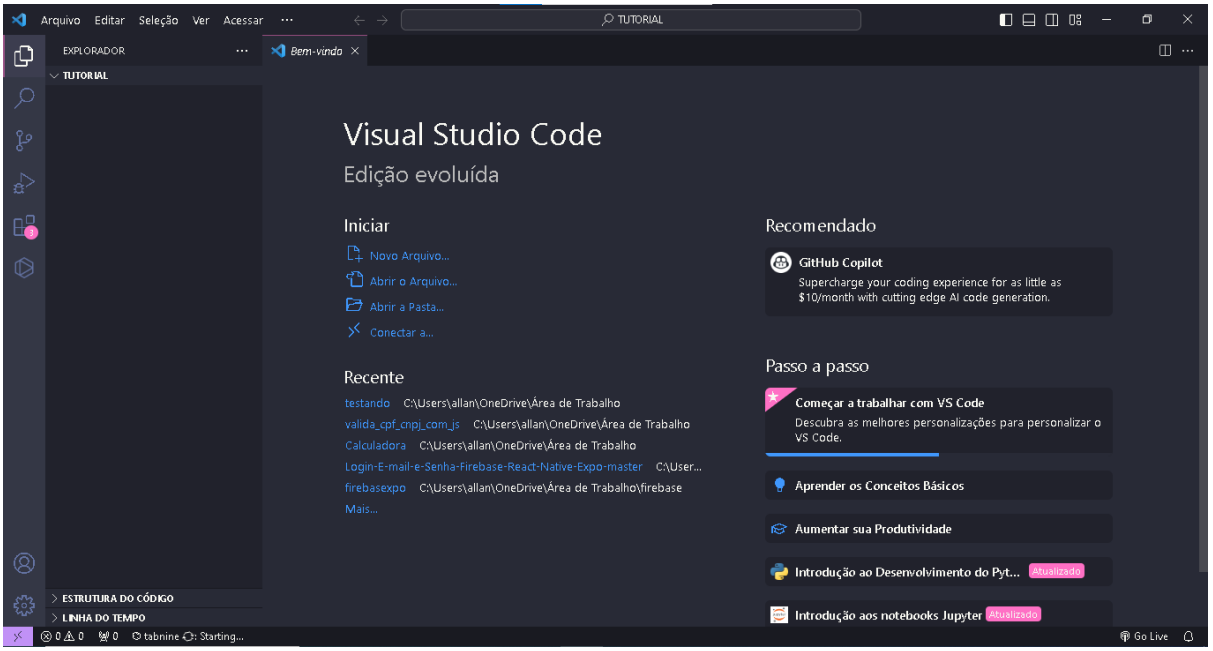
2º Abra e digite “cmd” no campo indicado:



3º Digite “ code . “

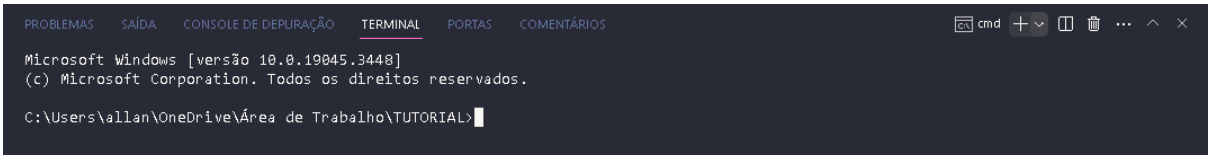


4º Com isso você vai abrir o VSC direto na pasta criada do seu projeto

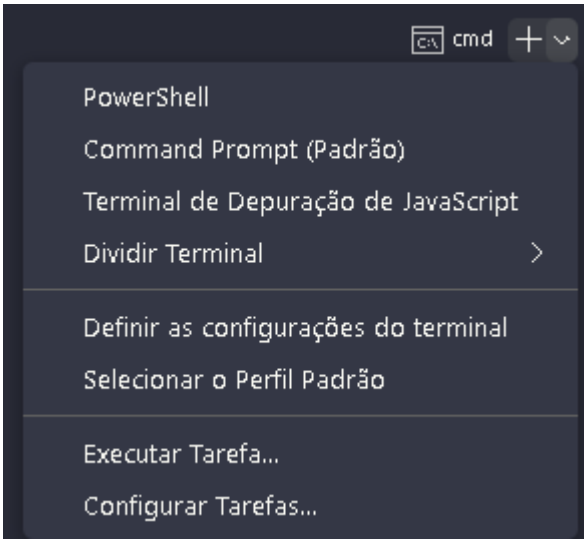


5º Configure o terminal para que ele utilize o CMD, clique em “Ver” e depois em terminal.

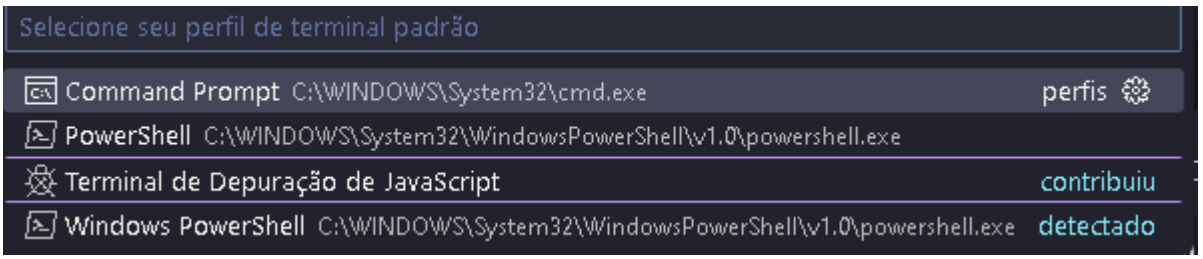
6º Clique no sinal de + indicado na imagem abaixo:



7º Selecione a opção “selecionar o perfil padrão “



8º E em seguida selecione o CMD:



9º Com o terminal configurado, vamos verificar se todas as nossas variáveis estão funcionando.

10º Com o terminal aberto, digite “ echo %JAVA_HOME% “:

```
C:\Tutorial-1>echo %JAVA_HOME%
C:\Program Files\Java\jdk-11
```

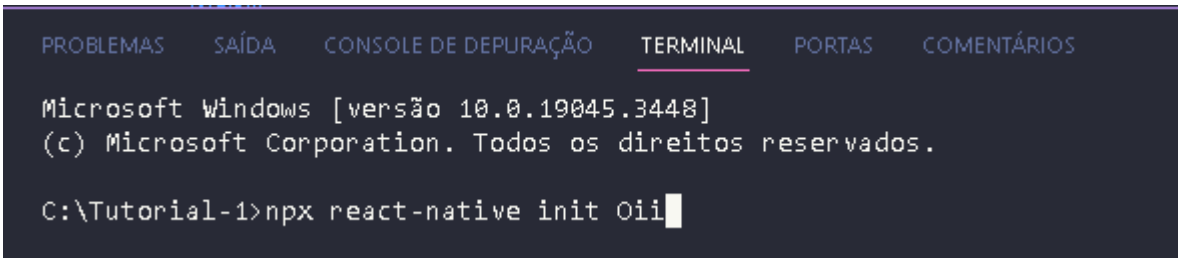
INFORMAÇÃO IMPORTANTE

```
C:\Users\allan\OneDrive\Área de Trabalho\TUTORIAL>
```

CASO o seu também apareceu “ OneDrive “ vai ser preciso criar uma nova pasta direto no C:

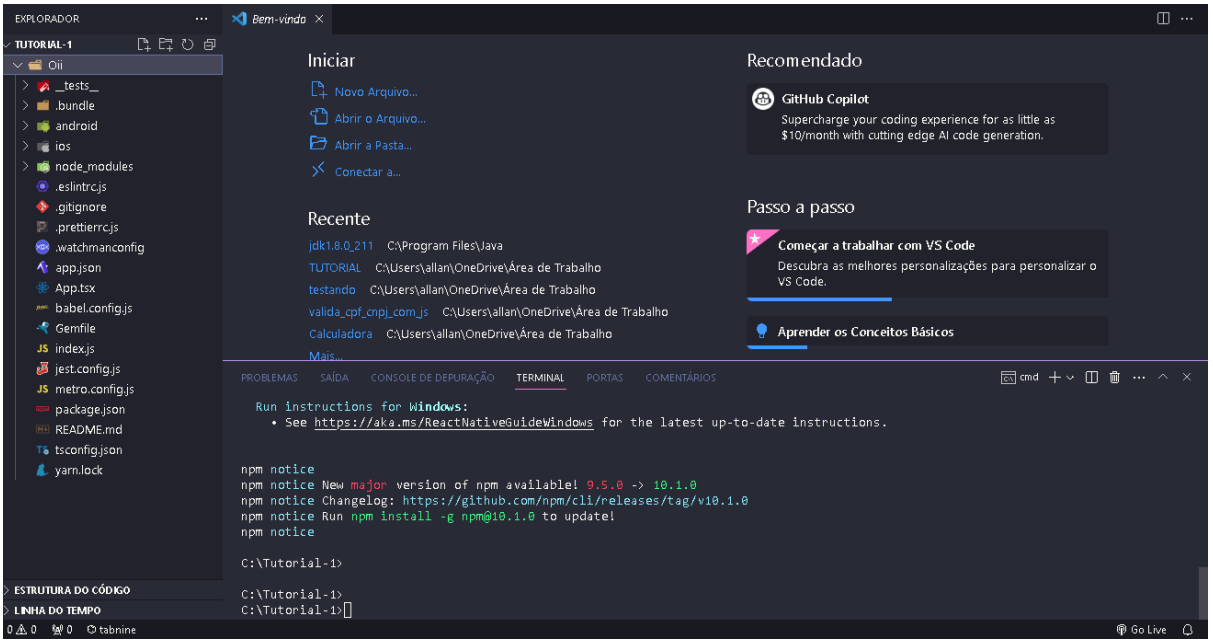
OBS: faça o mesmo esquema de abrir pelo CMD o VSC

11º Ainda com o terminal aberto, digite “ npx react-native init nome-do-projeto ”
OBS: Quando for colocar o nome do projeto, pense em um nome único e não composto.

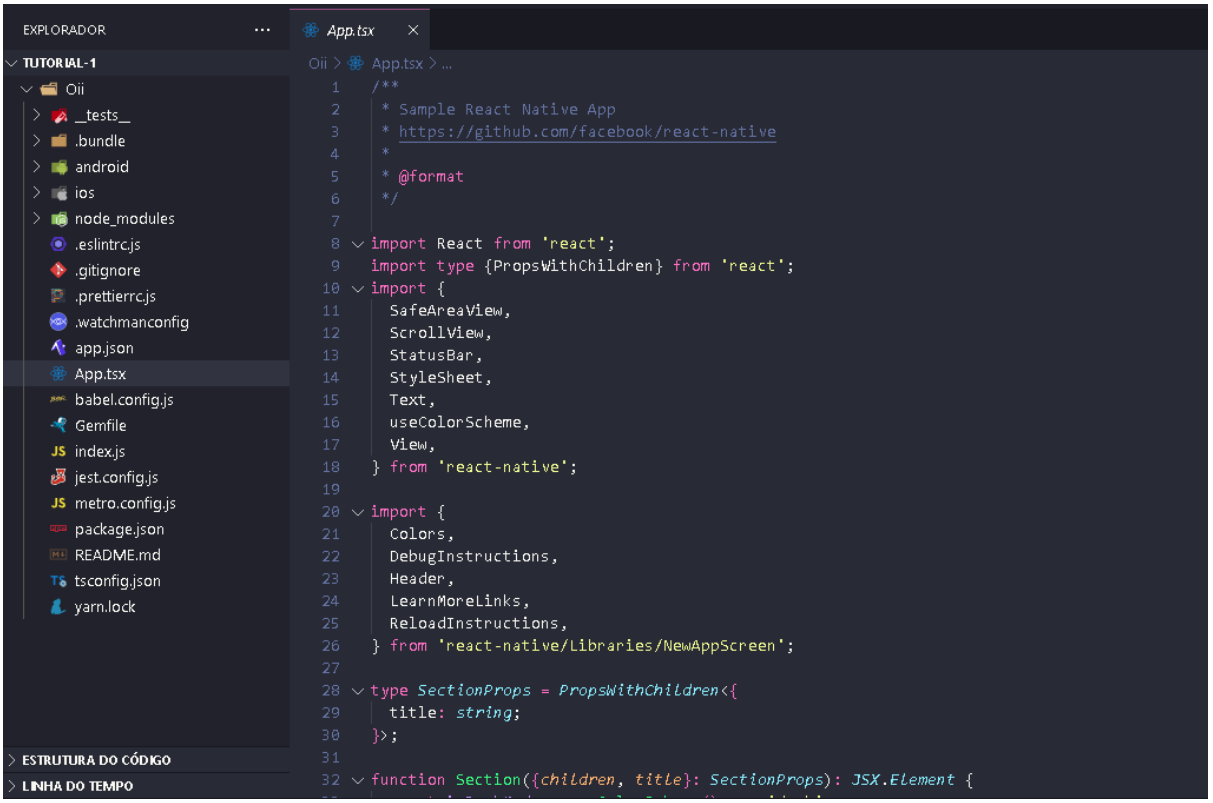


Aperte enter e espere

12º Com o download concluído você irá reparar que foram criadas pastas.



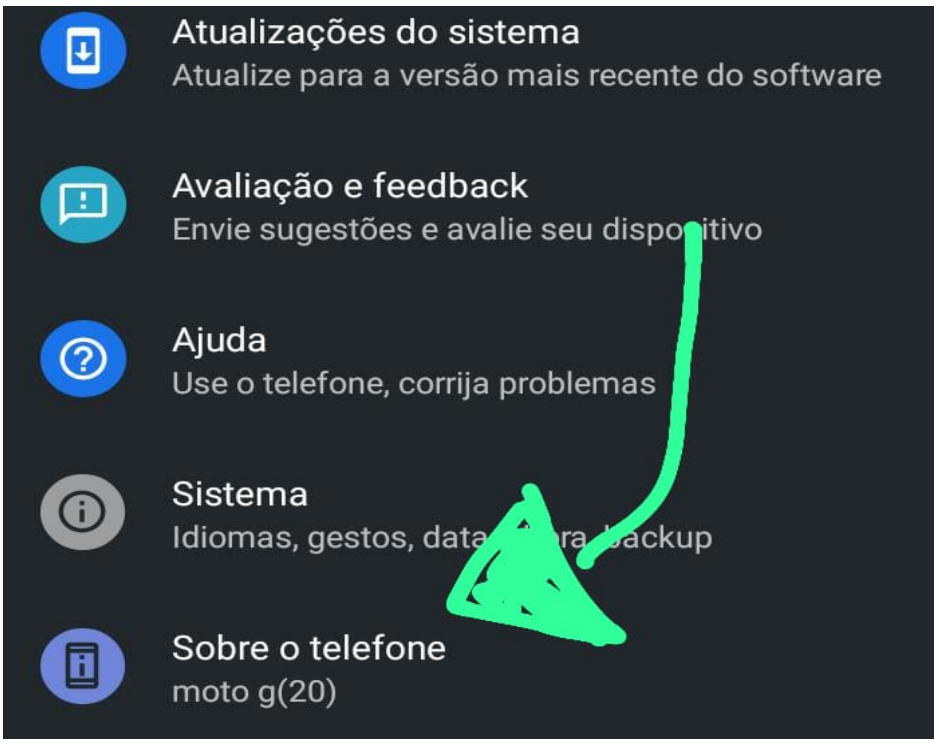
13º O “ App.tsx “ vai ser a aba principal para programar é nela onde vai sair as telas do app.

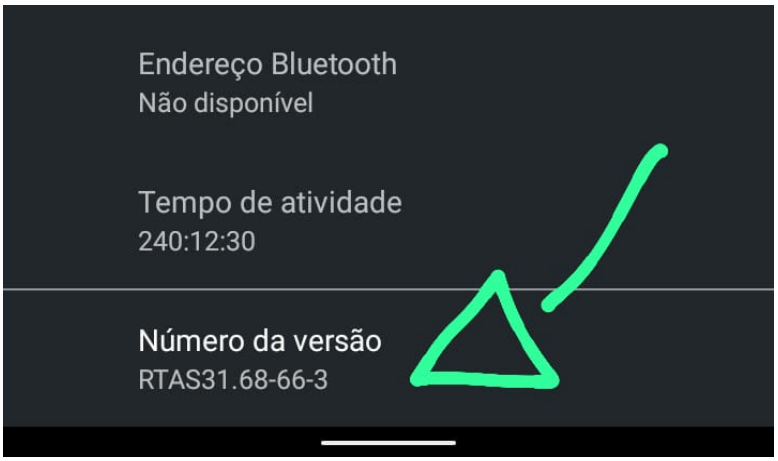


COMO RODAR O PROGRAMA

Existem algumas maneiras de executar o programa e no nosso projeto utilizamos um celular conectado via USB para transmitir o programa no celular. Segue abaixo como fazer isso:

1º Vai ser necessário configurar uma parte do seu celular, vá em configurações do seu aparelho e procure por “ Sobre o telefone “ em seguida clique em “ Número da versão “ até aparecer que você é um desenvolvedor.





2º Procure por " opções do desenvolvedor " e marque " Depuração USB "

4º Agora no VSC abra o terminal e digite o comando “ adb devices”

OBS: conecte o seu celular no pc por um cabo USB.

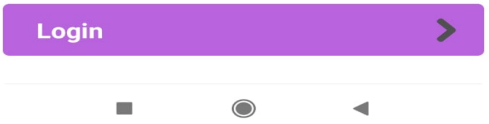
```
C:\Tutorial-1>adb devices
List of devices attached
ZF523NWMQL      device
```

5º Acesse o seu projeto pelo terminal usando o “ CD” e logo em seguida digite o comando “ npx react react-native run-android “e aguarde.

6º Seguindo todos os passos no seu celular irá abrir o app tudo certinho. Agora é só programar.

Como Fizemos o APP

1º TELA: A Splash Screen



Para a criação dessa tela o código foi o seguinte:

```
import React from 'react';
import {Image, StyleSheet, TouchableOpacity, Text, View} from 'react-native';
import {useNavigation} from '@react-navigation/native

export function Splash(){
  return (
    <View style={[StyleSheet.absoluteFillObject, styles.container]}>
      <Image source={require('./../../assets/brain.png')} style={styles.brainPic} />
      <Text style={styles.textLogo}>RM</Text>
```

```
    </View>
  );

}
export const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    backgroundColor: '#BA63DF',
    justifyContent:'center',
    flexDirection:'row',
  },
  brainPic:{
    width:220,
    height:180,
    position: 'absolute'
  },
  textLogo:{
    color:'#BA63DF',
    fontWeight:'bold',
    fontSize:30,
  }
})
```

Vamos entender o que fizemos:

1º Passo: As Importações

```
import React from 'react';
import {Image, StyleSheet, TouchableOpacity, Text, View} from
'react-native';
import {useNavigation} from '@react-navigation/native
```

Essa primeira linha de código são as importações de bibliotecas do react native, foi necessário baixar somente o React-navigation, para isso foi necessário ir no site deles e seguir os procedimentos da documentação(assim como fizemos com o react native, porém um pouco diferente nesse caso).
Link do site: [Getting started | React Navigation]

2º Passo: A Tela

```
export function Splash(){
  return (
    <View style={[StyleSheet.absoluteFillObject, styles.container]}>
      <Image source={require('./../../assets/brain.png')}
style={styles.brainPic} />
      <Text style={styles.textLogo}>RM</Text>
    </View>
  );
 }
```

Essa segunda linha do código é o que vai aparecer na tela do seu celular é o corpo do programa, sempre vai ser necessário usar a tag “<View>” para construir sua tela, pois ela é o componente mais básico do React-Native e é usada para agrupar outros componentes.

Já a tag “<Image source={require(...)} style = (..)/>” é usada para adicionar uma imagem no programa, para isso é necessário colocar a imagem no arquivo do programa para que ela seja compilada junto dele. Agora o comando “style = {styles.brainPic}” é usado para definir o estilo (style) que deseja usar no componente. Nesse caso, estamos utilizando o style brainPic que está dentro da variável “styles” a qual foi criada por nós em um outro arquivo.

O <Text> é para introduzir um texto que você queira na tela do app, nesse caso o texto é “ RM “. OBS: a tag <text> também leva um style.

3º Passo: Os Style

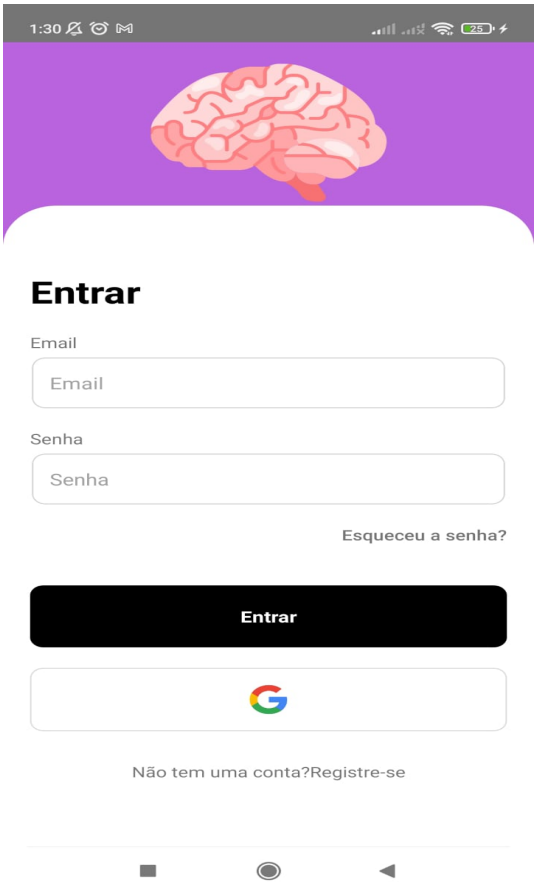
```
export const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    backgroundColor: '#BA63DF',
    justifyContent:'center',
    flexDirection:'row',
  },
  brainPic:{
    width:220,
    height:180,
    position: 'absolute'
  },
  textLogo:{
    color:'#BA63DF',
    fontWeight:'bold',
    fontSize:30,
  }
})
```

Bom como falado anteriormente os styles servem justamente para podermos estilizar os componentes e a própria página, usando o css. O css basicamente vai da uma forma, um estilo, deixar mais bonito, mais detalhado o app:

Flex: Quando aplicamos flex em um componente, ele se transforma em um *flex container*. Todos os componentes que são descendentes diretos do *flex container* podem ser remanejados com outras propriedades de **Flex**, por exemplo, alinhamento vertical, alinhamento horizontal, espaçamento, entre outros.

Color: Muda a cor.
fontWeight: define a espessura ou finura dos caracteres em um texto
fontSize: define o tamanho dos caracteres em um texto
Width: define a largura de um componente na tela
height: define a altura de um componente na tela
position: define como um componente é posicionado na tela
alignItems: define como os componentes filhos devem ser alinhados ao longo do eixo secundário do seu contêiner
backgroundcolor: define a cor de fundo de um componente no React Native.
justifyContent: define como os componentes filhos devem ser alinhados ao longo do eixo principal do seu contêiner.
flexDirection: controla a direção em que os componentes filhos de um contêiner são organizados. Isso também é chamado de eixo principal

2º TELA: O Login



Para a criação dessa tela foi utilizado o seguinte código:

```
import React from 'react';
import {Image, StyleSheet, TouchableOpacity, Text, View,TextInput} from 'react-native';
import {useNavigation} from '@react-navigation/native'
import { Input } from '../../components/input';
import {useForm, Controller} from 'react-hook-form';
import auth from '@react-native-firebase/auth';
import { Alert } from 'react-native/Libraries/Alert/Alert';
import {useState} from 'react'
import * as yup from 'yup';
import {yupResolver} from '@hookform/resolvers/yup';
import { AppLoader } from '../Loader';
import { Splash } from '../splash';

const [IsLoading, setIsLoading] = useState(false);

type FormDataProps ={
    email:string;
    password:string;
}
const signUpSchema = yup.object({
    email: yup.string().required('Informe o email').email('E-mail inválido'),
    password: yup.string().required('Informe a senha')
})
```

```
const navigation = useNavigation();

  function handleGoToRegister(){
    navigation.navigate('register')
  }
  function handleGoToWelcome(){
    navigation.navigate('welcome')
  }

export  function Login() {
    const [loginpending, setLoginPending] = useState(false);

    function handleSignIn(data: FormDataProps){
        logar(data.email, data.password)

        setLoginPending(true);
    }

    function logar(ema, pass){

    setIsLoading(true);

    let error;
    let email = ema;
    let password = pass;
    setLoginPending(true);

    auth()
    .signInWithEmailAndPassword(email, password)
    .then(() => handleGoToWelcome())
    .catch((error)=> (error))

    }

    const{control, handleSubmit, formState:{errors}} = useForm<FormDataProps>( {resolver:
yupResolver(signUpSchema)});

  return (
    <>
    <View style={styles.container}>
        <Image source={require('./../../assets/brain.png')} style={styles.picBrain}/>
      <View style={styles.containerBox} >
        <Text style={styles.title}>Entrar</Text>

        <Text style={styles.textLabel}>Email</Text>
        <Controller
            control={control}
            name="email"
            rules={{
                required:'Informe o Email.',
                pattern:{

value:/^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/,
                 message: 'Email inválido'
                }
            }}
```

```
            render={({field:{onChange}})=>(<Input
                errorMessage={errors.email?.message}
                onChangeText={onChange}
                placeholder='Email'
                />)}
        />


        <Text style={styles.textLabel}>Senha</Text>

        <Controller
            control={control}
            name="password"
            render={({field:{onChange}})=>(<Input
                errorMessage={errors.password?.message}
                onChangeText={onChange}
                secureTextEntry
                placeholder='Senha'
                />)}
        />

        <Text style={styles.textAlert}>Esqueceu a senha?</Text>
        <TouchableOpacity style={styles.btn}onPress={handleSubmit(handleSignIn)}>
            <Text style={styles.textBtn} >Entrar</Text>
        </TouchableOpacity>

        <TouchableOpacity style={styles.btnGoogle}>
            <Image source={require('./../../assets/google.png')} style={styles.picGoogle}
/>
        </TouchableOpacity>

        <TouchableOpacity style={styles.btnToRegister}
        onPress={handleGoToRegister}
        >
            <Text>Não tem uma conta?Registre-se</Text>
        </TouchableOpacity>

      </View>

    </View>
    {loginpending ? <Splash/>: null}
    </>
  );

}
export const styles = StyleSheet.create({
container:{
    flex: 1,
    alignItems: 'center',
    backgroundColor:'#BA63DF',
},
containerBox: {
    padding:20,
    backgroundColor:'#fff',
    justifyContent: 'center',
    flex:1,
    width:'100%',
```

```
    borderTopLeftRadius:40,
    borderTopRightRadius:40
},
title:{
    color:'black',
    fontWeight:'bold',
    fontSize: 30,
    marginBottom:20
},
btn:{
    backgroundColor:'black',
    alignItems:'center',
    justifyContent:'center',
    padding:20,
    borderRadius:10,
    marginBottom:20
},
btnGoogle:{
    backgroundColor:'white',
    alignItems:'center',
    justifyContent:'center',
    padding:15,
    borderRadius:10,
    borderWidth:1,
    borderColor:'#D8DADC',
    marginBottom:30
},
textBtn:{
    color:'white',
    fontWeight:'bold',
    fontSize:15
},
btnToRegister:{
    justifyContent:'center',
    alignItems:'center'
},
picGoogle:{
    width:30,
    height:30
},
picBrain:{
    width:180,
    height:160
},
inputEntrada:{
    backgroundColor:"#FFFFFF",
    width:350,
    height:50,
    borderRadius:10,
    marginBottom:5,
    padding:10,
    borderWidth:1,
    borderColor:'#D8DADC'
  },
  textLabel:{
    marginBottom:5
  },
```

```
  textAlert:{
    fontWeight:"500",
    alignSelf:"flex-end",
    marginBottom:40
  }
})
```

Vamos entender o que fizemos.

1° **Passo:** As Importações

```
import React from 'react';
import {Image, StyleSheet, TouchableOpacity, Text, View,TextInput} from
'react-native';
import {useNavigation} from '@react-navigation/native'
import { Input } from '../../components/input';
import {useForm, Controller} from 'react-hook-form';
import auth from '@react-native-firebase/auth';
import { Alert } from 'react-native/Libraries/Alert/Alert';
import {useState} from 'react';
import * as yup from 'yup';
import {yupResolver} from '@hookform/resolvers/yup';
import { AppLoader } from '../Loader';
import { Splash } from '../splash';
```

Nessa primeira parte do código foi-se importado todos os componentes, telas, bibliotecas e outras coisas que foram utilizadas para se criar a tela e suas funcionalidades.
Aqui, além das importações padrões do react-native foram importadas, as telas AppLoader e Splash de seus respectivos diretórios, bem como o componente Input que foi criado por nós para ser utilizado em outras partes do app. De resto, temos importações de bibliotecas e outros componentes que utilizamos para fazer a conexão com o banco de dados, o formulário de login e a navegação entre as telas.

- **import {useForm, Controller} from 'react-hook-form';**
  É uma biblioteca que auxilia na criação e validação dos formulários.
  Documentação para se utilizar: https://react-hook-form.com/get-started

- **import auth from '@react-native-firebase/auth';**
  Oferece suporte à autenticação usando senhas, números de telefone, provedores de identidade federados conhecidos, como Google, Facebook e Twitter, entre outros.
  Documentação para se utilizar: https://rnfirebase.io/auth/usage

- **import { Alert } from 'react-native/Libraries/Alert/Alert';**
  É uma API que funciona tanto no Android quanto no iOS e pode exibir alertas estáticos.
  Por ser parte de uma biblioteca que já é nativa do react-native, não é necessário baixar mais nada.

- **import {useState} from 'react';**
  Permite a criação de estado no componente através de função e faz o gerenciamento do estado local do componente retorna um array como resultado.
  Por ser parte de uma biblioteca que já é nativa do react-native, não é necessário baixar mais nada.

- **import * as yup from 'yup';**
  É um construtor de schema JavaScript para análise e validação de valor.
  Você pode encontrar um tutorial de como baixá-lo aqui: https://blog.betrybe.com/desenvolvimento-web/yup/

- **import {yupResolver} from '@hookform/resolvers/yup';**
  Função que irá gerar as validações com base no schema criado com o yup.
  Por ser parte de uma biblioteca do hook-from, não é necessário baixar mais nada.

2° **Passo:** O Código

```
const [IsLoading, setIsLoading] = useState(false);

type FormDataProps ={
    email:string;
    password:string;
}
const signUpSchema = yup.object({
    email: yup.string().required('Informe o email').email('E-mail
inválido'),
    password: yup.string().required('Informe a senha')
})

const navigation = useNavigation();

  function handleGoToRegister(){
    navigation.navigate('register')
  }
  function handleGoToWelcome(){
    navigation.navigate('welcome')
  }
```

```
const [IsLoading, setIsLoading] = useState(false);
```

- Isso cria uma variável chamada `IsLoading` e uma função chamada `setIsLoading`. O `useState` é uma função especial do React Native que é usada para gerenciar estados em um aplicativo. Neste caso, começamos com `IsLoading` definido como `false`, o que significa que inicialmente não há carregamento em andamento. Essa variável será usada para controlar se alguma ação está ocorrendo, como o carregamento de dados.

```
type FormDataProps = { email: string; password: string; }
```

- Isso define um tipo chamado `FormDataProps`, que descreve a estrutura de dados esperada para um objeto. Nesse caso, ele define que um objeto `FormDataProps` deve ter duas propriedades: `email` (que deve ser uma string) e `password` (que também deve ser uma string). Isso é útil para garantir que os dados estejam no formato correto.

```
const signUpSchema = yup.object({ email:
yup.string().required('Informe o email').email('E-mail
inválido'), password: yup.string().required('Informe a senha') })
```

- Aqui, estamos usando uma biblioteca chamada "yup" para criar um esquema de validação para os dados do usuário. Esse esquema

especifica que o `email` deve ser uma string obrigatória e deve ser um endereço de e-mail válido. Além disso, o `password` também é obrigatório. Isso ajuda a garantir que os dados inseridos pelo usuário atendam aos critérios necessários.

```
const navigation = useNavigation();
```

- Aqui, estamos obtendo uma referência à funcionalidade de navegação do aplicativo. A variável `navigation` será usada para mover o usuário entre as diferentes telas do aplicativo.

```
function handleGoToRegister() { navigation.navigate('register') }
```

- Esta é uma função chamada `handleGoToRegister`. Quando chamada, ela usa a variável `navigation` para navegar o usuário para a tela chamada 'register'. Isso é útil para permitir que o usuário vá para a tela de registro quando ocorrer um evento específico, como um clique em um botão.

```
function handleGoToWelcome() { navigation.navigate('welcome') }
```

- Similar à função anterior, esta é chamada `handleGoToWelcome` e navega o usuário para a tela chamada 'welcome' quando chamada. Isso é útil para permitir que o usuário vá para a tela de boas-vindas quando necessário.

3° **Passo:** A Tela

```
export  function Login() {
    const [loginpending, setLoginPending] = useState(false);

    function handleSignIn(data: FormDataProps){
        logar(data.email, data.password)

        setLoginPending(true);
    }

    function logar(ema, pass){

    setIsLoading(true);

    let error;
    let email = ema;
    let password = pass;
    setLoginPending(true);
```

```
    auth()
    .signInWithEmailAndPassword(email, password)
    .then(() => handleGoToWelcome())
    .catch((error)=> (error))

    }

    const{control, handleSubmit, formState:{errors}} = useForm<FormDataProps>(
{resolver: yupResolver(signUpSchema)});

  return (
    <>
    <View style={styles.container}>
        <Image source={require('./../../assets/brain.png')}
style={styles.picBrain}/>
      <View style={styles.containerBox} >
        <Text style={styles.title}>Entrar</Text>

        <Text style={styles.textLabel}>Email</Text>
        <Controller
            control={control}
            name="email"
            rules={{
                required:'Informe o Email.',
                pattern:{
value:/^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/,
                 message: 'Email inválido'
                }
            }}
            render={({field:{onChange}})=>(<Input
                errorMessage={errors.email?.message}
                onChangeText={onChange}
                placeholder='Email'
                />)}
        />


        <Text style={styles.textLabel}>Senha</Text>

        <Controller
            control={control}
            name="password"
            render={({field:{onChange}})=>(<Input
                errorMessage={errors.password?.message}
                onChangeText={onChange}
                secureTextEntry
                placeholder='Senha'
                />)}
        />
```

```
        <Text style={styles.textAlert}>Esqueceu a senha?</Text>
        <TouchableOpacity style={styles.btn}onPress={handleSubmit(handleSignIn)}>
            <Text style={styles.textBtn} >Entrar</Text>
        </TouchableOpacity>

        <TouchableOpacity style={styles.btnGoogle}>
            <Image source={require('./../../assets/google.png')}
style={styles.picGoogle} />
        </TouchableOpacity>

        <TouchableOpacity style={styles.btnToRegister}
        onPress={handleGoToRegister}
        >
            <Text>Não tem uma conta?Registre-se</Text>
        </TouchableOpacity>

      </View>

    </View>
    {loginpending ? <Splash/>: null}
    </>
  );
}
```

```
function handleSignIn(data: FormDataProps) { ... }
```

- Esta é uma função chamada `handleSignIn` que é chamada quando o usuário tenta fazer login. Ela recebe um objeto `data` que deve conter informações de login, como email e senha. A função chama-se outra função chamada `logar` para realizar o processo de login.

```
function logar(ema, pass) { ... }
```

- Esta é outra função chamada `logar` que recebe o email (`ema`) e a senha (`pass`) do usuário como parâmetros. Ela configura o estado `isLoading` como `true`, o que indica que o aplicativo está em processo de carregamento. Em seguida, utiliza a biblioteca de autenticação (possivelmente Firebase, dado o uso de `auth()`) para tentar realizar o login com as credenciais fornecidas. Se o login for bem-sucedido, ele redireciona o usuário para a tela de boas-vindas (`handleGoToWelcome`). Se houver algum erro durante o processo de login, o erro é capturado, mas não está claro como ele é tratado no código.

```
const { control, handleSubmit, formState: { errors } } =
useForm<FormDataProps>({ resolver: yupResolver(signUpSchema) });
```

- Aqui, estão sendo usados hooks de controle de formulário para gerenciar o formulário de login. Ele configura as regras de validação

definidas anteriormente no `signUpSchema` para validar os campos de email e senha.

O código a partir de `<View style={styles.container}>` até `<TouchableOpacity style={styles.btnToRegister}>` define a interface de usuário (UI) para a tela de login. Ele inclui campos para o email e senha, um botão "Entrar", um botão para autenticação via Google e um link para a tela de registro.

`{loginpending ? <Splash /> : null}`

- Isso exibe o componente chamado `<Splash />` se `loginpending` for `true`. Ou seja, é uma tela de carregamento que é mostrada enquanto o processo de login está ocorrendo.

4° **Passo:** Os Styles

```
export const styles = StyleSheet.create({
container:{
    flex: 1,
    alignItems: 'center',
    backgroundColor:'#BA63DF',
},
containerBox: {
    padding:20,
    backgroundColor:'#fff',
    justifyContent: 'center',
    flex:1,
    width:'100%',
    borderTopLeftRadius:40,
    borderTopRightRadius:40
},
title:{
    color:'black',
    fontWeight:'bold',
    fontSize: 30,
    marginBottom:20
},
```

```
btn:{
    backgroundColor:'black',
    alignItems:'center',
    justifyContent:'center',
    padding:20,
    borderRadius:10,
    marginBottom:20
},
btnGoogle:{
    backgroundColor:'white',
    alignItems:'center',
    justifyContent:'center',
    padding:15,
    borderRadius:10,
    borderWidth:1,
    borderColor:'#D8DADC',
    marginBottom:30
},
textBtn:{
    color:'white',
    fontWeight:'bold',
    fontSize:15
},
btnToRegister:{
    justifyContent:'center',
    alignItems:'center'
},
picGoogle:{
    width:30,
    height:30
},
picBrain:{
    width:180,
    height:160
},
inputEntrada:{
    backgroundColor:"#FFFFFF",
    width:350,
    height:50,
    borderRadius:10,
    marginBottom:5,
    padding:10,
    borderWidth:1,
    borderColor:'#D8DADC'
  },
```

```
  textLabel:{
    marginBottom:5
  },
  textAlert:{
    fontWeight:"500",
    alignSelf:"flex-end",
    marginBottom:40
  }
})
```

container: Este estilo é aplicado a um componente de nível superior (talvez a tela inteira) e define a aparência do contêiner. Ele especifica que o contêiner deve ocupar todo o espaço disponível, alinhar os itens no centro e ter um fundo de cor roxa (#BA63DF).

containerBox: Este estilo é aplicado a um contêiner dentro do contêiner principal. Ele tem um fundo branco e preenche com espaço interno (padding). Além disso, ele tem bordas arredondadas nas partes superiores esquerda e direita, o que pode dar a ele uma aparência de cartão.

title: Este estilo é aplicado a um elemento de texto (provavelmente um título) dentro do contêiner. Ele define o tamanho da fonte, a cor e o peso da fonte.

btn: Este estilo é aplicado a um botão. Ele tem um fundo preto, texto branco, e bordas arredondadas.

btnGoogle: Este estilo é aplicado a outro botão, provavelmente usado para fazer login com o Google. Ele tem um fundo branco, bordas arredondadas e uma borda cinza ao redor.

textBtn: Este estilo é aplicado ao texto dentro dos botões. Ele define a cor e o peso da fonte.

btnToRegister: Este estilo é aplicado a um contêiner (provavelmente um TouchableOpacity) que redireciona para a tela de registro. Ele é usado para alinhar o texto e o link "Não tem uma conta? Registre-se".

picGoogle e picBrain: Estes estilos são aplicados a imagens (talvez ícones). Eles definem a largura e a altura das imagens.

inputEntrada: Este estilo é aplicado a campos de entrada de texto. Ele define a aparência de um campo de entrada, incluindo um fundo branco, bordas arredondadas, e uma borda cinza ao redor.

textLabel: Este estilo é aplicado a elementos de texto que podem ser rótulos para campos de entrada. Ele define uma pequena margem inferior.

textAlert: Este estilo é aplicado a um texto que pode ser usado para alertas ou mensagens informativas. Ele tem uma margem inferior maior e um peso de fonte mais forte.

4° Tela: O Cadastro

Para a criação dessa tela foi utilizado o seguinte código:

```
import React from 'react';
import {Image, StyleSheet, TouchableOpacity, Text, View} from 'react-native';
import {useNavigation} from '@react-navigation/native'
import { Input } from '../../components/input';
import { Button } from '../../components/input/btn';
import { useForm, Controller } from 'react-hook-form';
import * as yup from 'yup';
import {yupResolver} from '@hookform/resolvers/yup';
import auth from '@react-native-firebase/auth';
```

```
import {useState} from 'react'
import { Alert } from 'react-native';
import { Splash } from '../splash';

const navigation = useNavigation();

function handleGoToLogin(){
    navigation.navigate('login');
}

type FormDataProps ={
    name:string;
    email:string;
    password:string;
    confpassw:string;
}
const signUpSchema = yup.object({
    name: yup.string().required('Informe o nome'),
    email: yup.string().required('Informe o email').email('E-mail inválido'),
    password: yup.string().required('Informe a senha').min(6, 'a senha deve ter no minimo 6
digitos.'),
    confpassw: yup.string().required('Confirme a senha').oneOf([yup.ref('password')], 'A
confirmação da senha não confere.')
})

const [IsLoading, setIsLoading] = useState(false);

export function Register() {

    function handleSignUp(data: FormDataProps){

        handleNewUser(data.email, data.password)
        setLoginPending(true)
    }

    function handleNewUser(em, pass){
        let email = em;
        let password = pass;
        setIsLoading(true);
        setLoginPending(false)
        auth()
        .createUserWithEmailAndPassword(email, password)
        .then(() => handleGoToLogin())
        .catch((error) => console.log(error))
    }

    const [loginpending, setLoginPending] = useState(false);

    const{control, handleSubmit, formState:{errors}} = useForm<FormDataProps>( {resolver:
yupResolver(signUpSchema)});

  return (
    <>
    <View style={styles.container}>
        <Image source={require('./../../assets/brain.png')} style={styles.picBrain} />
      <View style={styles.containerBox} >
        <Text style={styles.title}>Criar conta</Text>
```

```
        <Controller
            control={control}
            name="name"
            render={({field:{onChange}})=> (<Input
                onChangeText={onChange}
                errorMessage={errors.name?.message}
                placeholder='Nome'
                />)}
        />
        <Controller
            control={control}
            name="email"
            rules={{
                required:'Informe o Email.',
                pattern:{

value:/^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/,
                 message: 'Email inválido'
                }
            }}
            render={({field:{onChange}})=>(<Input
                errorMessage={errors.email?.message}
                onChangeText={onChange}
                placeholder='Email'
                />)}
        />
        <Controller
            control={control}
            name="password"
            render={({field:{onChange}})=>(<Input
                errorMessage={errors.password?.message}
                onChangeText={onChange}
                secureTextEntry
                placeholder='Senha'
                />)}
        />
        <Controller
            control={control}
            name="confpassw"
            render={({field:{onChange}})=>(<Input
                errorMessage={errors.confpassw?.message}
                onChangeText={onChange}
                placeholder='Confirme a senha.'
                />)}
        />

        <TouchableOpacity style={styles.btn}
        onPress={handleSubmit(handleSignUp)}
        >
            <Text style={styles.textBtn}>Registrar</Text>
        </TouchableOpacity>

        <TouchableOpacity style={styles.btnToRegister}
        onPress={handleGoToLogin}>
            <Text>Já tem uma conta?Entrar</Text>
        </TouchableOpacity>
```

```
      </View>
    </View>
    {loginpending ? <Splash/>: null}
    </>
  );
}
export const styles = StyleSheet.create({
container:{
    flex: 1,
    alignItems: 'center',
    backgroundColor:'#BA63DF',

},
containerBox: {
    padding:20,
    backgroundColor:'#fff',
    justifyContent: 'center',
    flex:1,
    width:'100%',
    borderTopLeftRadius:40,
    borderTopRightRadius:40
},
title:{
    color:'black',
    fontWeight:'bold',
    fontSize: 30,
    marginBottom:30
},
btn:{
    backgroundColor:'black',
    alignItems:'center',
    justifyContent:'center',
    padding:20,
    borderRadius:10,
    marginBottom:30
},
btnGoogle:{
    backgroundColor:'white',
    alignItems:'center',
    justifyContent:'center',
    padding:15,
    borderRadius:10,
    borderWidth:1,
    borderColor:'#D8DADC'
},
textBtn:{
    color:'white',
    fontWeight:'bold',
    fontSize:15
},
btnToRegister:{
    justifyContent:'center',
    alignItems:'center'
},
picBrain:{
    width:180,
```

```
    height:160
}
})
```

Vamos entender o que fizemos:

1° Passo: As Importações

```
import React from 'react';
import {Image, StyleSheet, TouchableOpacity, Text, View} from 'react-native';
import {useNavigation} from '@react-navigation/native'
import { Input } from '../../components/input';
import { Button } from '../../components/input/btn';
import { useForm, Controller } from 'react-hook-form';
import * as yup from 'yup';
import {yupResolver} from '@hookform/resolvers/yup';
import auth from '@react-native-firebase/auth';
import {useState} from 'react'
import { Alert } from 'react-native';
import { Splash } from '../splash';
```

As importações são as mesmas da tela anterior.

2° Passo: O Código

```
const navigation = useNavigation();

function handleGoToLogin(){
    navigation.navigate('login');
}

type FormDataProps ={
    name:string;
    email:string;
    password:string;
    confpassw:string;
}
const signUpSchema = yup.object({
    name: yup.string().required('Informe o nome'),
    email: yup.string().required('Informe o email').email('E-mail inválido'),
    password: yup.string().required('Informe a senha').min(6, 'a senha deve ter no
minimo 6 digitos.'),
    confpassw: yup.string().required('Confirme a
senha').oneOf([yup.ref('password')], 'A confirmação da senha não confere.')
})

const [IsLoading, setIsLoading] = useState(false);
```

const navigation = useNavigation();

Aqui, você está obtendo a funcionalidade de navegação do React Navigation por meio do hook useNavigation(). Isso permite que você navegue entre diferentes telas do aplicativo quando necessário.

```
function handleGoToLogin() { ... }
```

Esta é uma função chamada handleGoToLogin que é chamada quando o usuário deseja voltar para a tela de login. Ela utiliza a variável navigation para navegar o usuário de volta para a tela de login.

type FormDataProps = { ... }

Isso define um tipo chamado FormDataProps que descreve a estrutura de dados esperada para um objeto de formulário de registro. Ele inclui propriedades como name, email, password e confpassw, cada uma com seu tipo de dado.

const signUpSchema = yup.object({ ... })

Aqui, você está usando a biblioteca "yup" para criar um esquema de validação para os dados do formulário de registro. O esquema especifica as regras de validação para cada campo, como a obrigatoriedade do nome, a validação de formato do email, a obrigatoriedade da senha (com no mínimo 6 caracteres) e a correspondência da confirmação da senha com a senha original.

const [IsLoading, setIsLoading] = useState(false);

Isso cria uma variável de estado chamada IsLoading e uma função setIsLoading para controlar o estado de carregamento. Inicialmente, o estado IsLoading é definido como false, o que significa que não há carregamento em andamento. Essa variável pode ser usada para exibir uma tela de carregamento quando algum processo estiver em execução.

3° Passo: A Tela

```
export function Register() {

    function handleSignUp(data: FormDataProps){

        handleNewUser(data.email, data.password)
        setLoginPending(true)
    }

    function handleNewUser(em, pass){
        let email = em;
        let password = pass;
        setIsLoading(true);
        setLoginPending(false)
        auth()
        .createUserWithEmailAndPassword(email, password)
        .then(() => handleGoToLogin())
        .catch((error) => console.log(error))
    }

    const [loginpending, setLoginPending] = useState(false);

    const{control, handleSubmit, formState:{errors}} = useForm<FormDataProps>(
{resolver: yupResolver(signUpSchema)});

  return (
    <>
    <View style={styles.container}>
        <Image source={require('./../../assets/brain.png')} style={styles.picBrain}
/>
      <View style={styles.containerBox} >
        <Text style={styles.title}>Criar conta</Text>

        <Controller
            control={control}
            name="name"
            render={({field:{onChange}})=> (<Input
                onChangeText={onChange}
```

```
                errorMessage={errors.name?.message}
                placeholder='Nome'
                />)}
        />
        <Controller
            control={control}
            name="email"
            rules={{
                required:'Informe o Email.',
                pattern:{

value:/^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/,
                 message: 'Email inválido'
                }
            }}
            render={({field:{onChange}})=>(<Input
                errorMessage={errors.email?.message}
                onChangeText={onChange}
                placeholder='Email'
                />)}
        />
        <Controller
            control={control}
            name="password"
            render={({field:{onChange}})=>(<Input
                errorMessage={errors.password?.message}
                onChangeText={onChange}
                secureTextEntry
                placeholder='Senha'
                />)}
        />
        <Controller
            control={control}
            name="confpassw"
            render={({field:{onChange}})=>(<Input
                errorMessage={errors.confpassw?.message}
                onChangeText={onChange}
                placeholder='Confirme a senha.'
                />)}
        />

        <TouchableOpacity style={styles.btn}
        onPress={handleSubmit(handleSignUp)}
        >
            <Text style={styles.textBtn}>Registrar</Text>
        </TouchableOpacity>

        <TouchableOpacity style={styles.btnToRegister}
        onPress={handleGoToLogin}>
            <Text>Já tem uma conta?Entrar</Text>
```

```
        </TouchableOpacity>

      </View>
    </View>
    {loginpending ? <Splash/>: null}
    </>
  );
}
```

function handleSignUp(data: FormDataProps) { ... }

Esta é uma função chamada handleSignUp que é chamada quando o usuário tenta se registrar. Ela recebe um objeto data contendo informações como nome, email, senha e confirmação de senha. A função chama outra função chamada handleNewUser para realizar o processo de registro.

function handleNewUser(em, pass) { ... }

A função handleNewUser é responsável por realizar o processo de registro do novo usuário. Ela recebe o email (em) e a senha (pass) como parâmetros, configura o estado isLoading como true (indicando que um processo está em andamento), e usa a função createUserWithEmailAndPassword da biblioteca de autenticação (possivelmente Firebase) para criar uma nova conta de usuário com o email e a senha fornecidos. Se o registro for bem-sucedido, o usuário é redirecionado para a tela de login (handleGoToLogin). Se ocorrer algum erro durante o registro, o erro é registrado no console.

const [loginpending, setLoginPending] = useState(false);

Aqui, você está criando uma variável de estado chamada loginpending e uma função setLoginPending para controlar se o processo de login está pendente. Inicialmente, loginpending é definido como false.

A partir de <View style={styles.container}> até <TouchableOpacity style={styles.btnToRegister}>, você está definindo a interface de usuário (UI) da

tela de registro. Isso inclui campos para nome, email, senha, confirmação de senha e botões para registrar o usuário e voltar para a tela de login.

{loginpending ? <Splash /> : null}

Isso exibe um componente chamado <Splash /> se loginpending for true. Parece ser uma tela de carregamento que é mostrada enquanto o processo de registro está em andamento.

4°Passo: Os Styles

```
export const styles = StyleSheet.create({
container:{
    flex: 1,
    alignItems: 'center',
    backgroundColor:'#BA63DF',

},
containerBox: {
    padding:20,
    backgroundColor:'#fff',
    justifyContent: 'center',
    flex:1,
    width:'100%',
    borderTopLeftRadius:40,
    borderTopRightRadius:40
},
title:{
    color:'black',
    fontWeight:'bold',
    fontSize: 30,
    marginBottom:30
},
btn:{
    backgroundColor:'black',
    alignItems:'center',
    justifyContent:'center',
    padding:20,
    borderRadius:10,
    marginBottom:30
},
btnGoogle:{
```

```
    backgroundColor:'white',
    alignItems:'center',
    justifyContent:'center',
    padding:15,
    borderRadius:10,
    borderWidth:1,
    borderColor:'#D8DADC'
},
textBtn:{
    color:'white',
    fontWeight:'bold',
    fontSize:15
},
btnToRegister:{
    justifyContent:'center',
    alignItems:'center'
},
picBrain:{
    width:180,
    height:160
}
})
```

container: Este estilo é aplicado ao componente de nível superior (provavelmente a tela inteira). Ele configura o fundo para uma cor roxa (#BA63DF), alinha os itens no centro e define a altura para ocupar todo o espaço disponível (flex: 1).

containerBox: Este estilo é aplicado a um contêiner dentro do contêiner principal. Ele tem um fundo branco (#fff), preenchimento interno (padding), e configura as bordas superiores esquerda e direita para serem arredondadas, o que pode dar à interface uma aparência de cartão.

title: Este estilo é aplicado a um elemento de texto (provavelmente um título). Ele define a cor do texto como preto, o tamanho da fonte como 30 pixels, peso da fonte como negrito (fontWeight: 'bold') e margem inferior de 30 pixels.

btn: Este estilo é aplicado a um botão. Ele define um fundo preto, cor do texto como branco, alinha o conteúdo no centro e configura um preenchimento (padding) interno. Também define bordas arredondadas e uma margem inferior de 30 pixels.

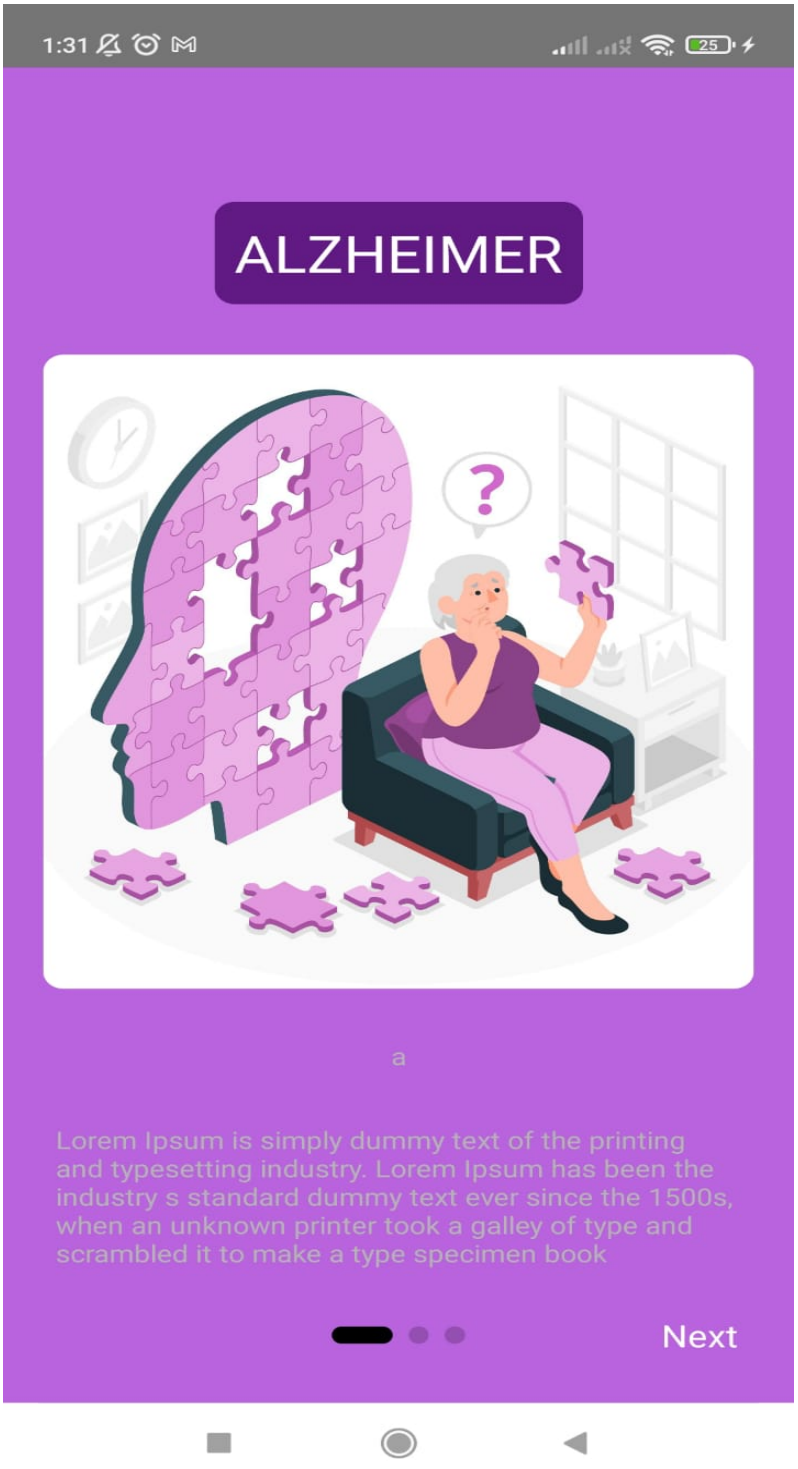
btnGoogle: Este estilo é aplicado a um segundo botão, provavelmente usado para fazer login com o Google. Ele tem um fundo branco, uma borda cinza (#D8DADC) ao redor, e configurações semelhantes de alinhamento e margem como o estilo anterior.

textBtn: Este estilo é aplicado ao texto dentro dos botões. Ele define a cor do texto como branco, o tamanho da fonte como 15 pixels e o peso da fonte como negrito.

btnToRegister: Este estilo é aplicado a um contêiner (provavelmente um TouchableOpacity) usado para redirecionar o usuário para a tela de login. Ele alinha o conteúdo no centro.

picBrain: Este estilo é aplicado a uma imagem (provavelmente um logotipo). Ele define a largura e a altura da imagem.

5° Tela: Bem Vindo ao App



Para a criação dessa tela foi utilizado o seguinte código:

```
import React from 'react';
import {Image, StyleSheet, TouchableOpacity, Text, View} from 'react-native';
import {useNavigation} from '@react-navigation/native'
import AppIntroSlider from 'react-native-app-intro-slider';
import { useState } from 'react';

const slides=[
    {
        key:'1',
        title:'a',
        text:'Lorem Ipsum is simply dummy text of the printing and typesetting industry.
Lorem Ipsum has been the industry s standard dummy text ever since the 1500s, when an
unknown printer took a galley of type and scrambled it to make a type specimen book',
        image:require('./../../assets/1.jpg')
    },
    {
        key:'2',
        title:'b',
        text:'Lorem Ipsum is simply dummy text of the printing and typesetting industry.
Lorem Ipsum has been the industrys standard dummy text ever since the 1500s, when an
unknown printer took a galley of type and scrambled it to make a type specimen book.',
        image:require('./../../assets/2.jpg')
    },
    {
        key:'3',
        title:'c',
        text:'Lorem Ipsum is simply dummy text of the printing and typesetting industry.
Lorem Ipsum has been the industrys standard dummy text ever since the 1500s, when an
unknown printer took a galley of type and scrambled it to make a type specimen book.',
        image:require('./../../assets/3.jpg')
    }
]

const navigation = useNavigation();

function handleGoTo(){
    navigation.navigate('inicio')
}

export function Welcome(){
    const [ showHome,setShowHome]= useState(false);

     function renderSlides({ item }){
        return(
        <View style = {styles.container}
```

```
        >
            <Text style={styles.title}>ALZHEIMER</Text>
            <Image source={item.image} style={styles.imgAlz}/>
            <Text style={styles.textSlider}>{item.title}</Text>
            <Text style={styles.textSlider}>{item.text}</Text>

        </View>)
     }
    if(showHome){
        return <Text>entrou na home</Text>
    }
    else{
  return (
    <AppIntroSlider
    renderItem={renderSlides}
    data={slides}
    activeDotStyle={
        {
            backgroundColor:'black',
            width:30
        }
    }
    onDone={()=> handleGoTo()}
    />
  );
    }
}
export const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    backgroundColor: '#BA63DF',
    justifyContent:'center',
    padding:20,
    gap:30
  },
  imgAlz:{
    width:'100%',
    height:'50%',
    resizeMode:'cover',
    borderRadius:10
  },
  textSlider:{
    color:'#b5b5b5',
    justifyContent:'center',
  },
  title:{
    fontSize:30,
    color:'#fff',
    backgroundColor:'#611A83',
    padding:10,
    borderRadius:10
  }
})
```

Vamos entender o que fizemos:

1° Passo: As Importações

```
import React from 'react';
import {Image, StyleSheet, TouchableOpacity, Text, View} from 'react-native';
import {useNavigation} from '@react-navigation/native'
import AppIntroSlider from 'react-native-app-intro-slider';
import { useState } from 'react';
```

2° Passo: O Código

```
const slides=[
    {
        key:'1',
        title:'a',
        text:'Lorem Ipsum is simply dummy text of the printing and typesetting
industry. Lorem Ipsum has been the industry s standard dummy text ever since the
1500s, when an unknown printer took a galley of type and scrambled it to make a
type specimen book',
        image:require('./../../assets/1.jpg')
    },
    {
        key:'2',
        title:'b',
        text:'Lorem Ipsum is simply dummy text of the printing and typesetting
industry. Lorem Ipsum has been the industrys standard dummy text ever since the
1500s, when an unknown printer took a galley of type and scrambled it to make a
type specimen book.',
        image:require('./../../assets/2.jpg')
    },
    {
        key:'3',
        title:'c',
        text:'Lorem Ipsum is simply dummy text of the printing and typesetting
industry. Lorem Ipsum has been the industrys standard dummy text ever since the
1500s, when an unknown printer took a galley of type and scrambled it to make a
type specimen book.',
        image:require('./../../assets/3.jpg')
    }
]
```

```
const navigation = useNavigation();

function handleGoTo(){
    navigation.navigate('inicio')
}
```

slides: Este é um array que contém objetos, cada um representando um slide na apresentação. Cada objeto possui as seguintes propriedades:

key: Uma chave única para identificar o slide.

title: O título do slide, que é uma string.

text: O texto de descrição do slide, que é uma string longa com informações fictícias de exemplo.

image: A imagem associada ao slide, que é carregada a partir de um arquivo de imagem local usando o require.

Esses objetos representam os diferentes slides que serão exibidos na tela de introdução do aplicativo, onde cada slide terá seu próprio título, texto descritivo e imagem.

const navigation = useNavigation();: Aqui, você está usando o hook useNavigation para obter a funcionalidade de navegação do React Navigation. Isso permite que você navegue o usuário para outra tela quando necessário.

function handleGoTo() { ... }: Esta é uma função chamada handleGoTo, que provavelmente é chamada quando o usuário conclui a apresentação de slides e deseja ir para a próxima tela (provavelmente a tela inicial do aplicativo). A função utiliza a variável navigation para navegar o usuário para a tela chamada 'inicio'.

3° Passo: A Tela

```
export function Welcome(){
```

```
    const [ showHome,setShowHome]= useState(false);

     function renderSlides({ item }){
        return(
        <View style = {styles.container}
        >
            <Text style={styles.title}>ALZHEIMER</Text>
            <Image source={item.image} style={styles.imgAlz}/>
            <Text style={styles.textSlider}>{item.title}</Text>
            <Text style={styles.textSlider}>{item.text}</Text>

        </View>)
     }
    if(showHome){
        return <Text>entrou na home</Text>
    }
    else{
  return (
    <AppIntroSlider
    renderItem={renderSlides}
    data={slides}
    activeDotStyle={
        {
            backgroundColor:'black',
            width:30
        }
    }
    onDone={()=> handleGoTo()}
    />
  );
    }
}
```

const [showHome, setShowHome] = useState(false);

Aqui, você está definindo um estado showHome com valor inicial false. Esse estado provavelmente é usado para controlar se a tela inicial do aplicativo deve ser mostrada após a conclusão da introdução de slides.

function renderSlides({ item }) { ... }

Esta é uma função chamada renderSlides que renderiza cada slide da apresentação. Ela recebe um objeto item como argumento, que representa um slide da apresentação. Dentro da função, você está renderizando o título, uma imagem, o título do slide (item.title) e o texto de descrição (item.text) em um contêiner.

if (showHome) { ... } else { ... }

Este é um condicional que verifica se showHome é true. Se for verdadeiro, ele retorna um componente de texto com a mensagem "entrou na home". Caso contrário, ele renderiza a apresentação de slides usando a biblioteca react-native-app-intro-slider.

<AppIntroSlider ... />

Aqui, você está renderizando a componente AppIntroSlider da biblioteca react-native-app-intro-slider. Este componente cria uma série de slides interativos e permite ao usuário passar pelos slides deslizando o dedo. Você está configurando esta componente com os seguintes props:

renderItem: Passando a função renderSlides para renderizar cada slide.

data: Passando o array slides que contém as informações sobre cada slide.

activeDotStyle: Configurando o estilo dos pontos de navegação ativos (indicando em qual slide o usuário está).

onDone: Definindo a função handleGoTo() para ser chamada quando o usuário concluir a apresentação de slides.

onDone={()=> handleGoTo()}

Quando o usuário conclui a apresentação de slides, a função handleGoTo é chamada. Presumivelmente, esta função navegará o usuário para a tela inicial do aplicativo.

4°Passo: Os Styles

```
export const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    backgroundColor: '#BA63DF',
    justifyContent:'center',
    padding:20,
    gap:30
  },
  imgAlz:{
    width:'100%',
    height:'50%',
```

```
    resizeMode:'cover',
    borderRadius:10
  },
  textSlider:{
    color:'#b5b5b5',
    justifyContent:'center',
  },
  title:{
    fontSize:30,
    color:'#fff',
    backgroundColor:'#611A83',
    padding:10,
    borderRadius:10
  }
})
```

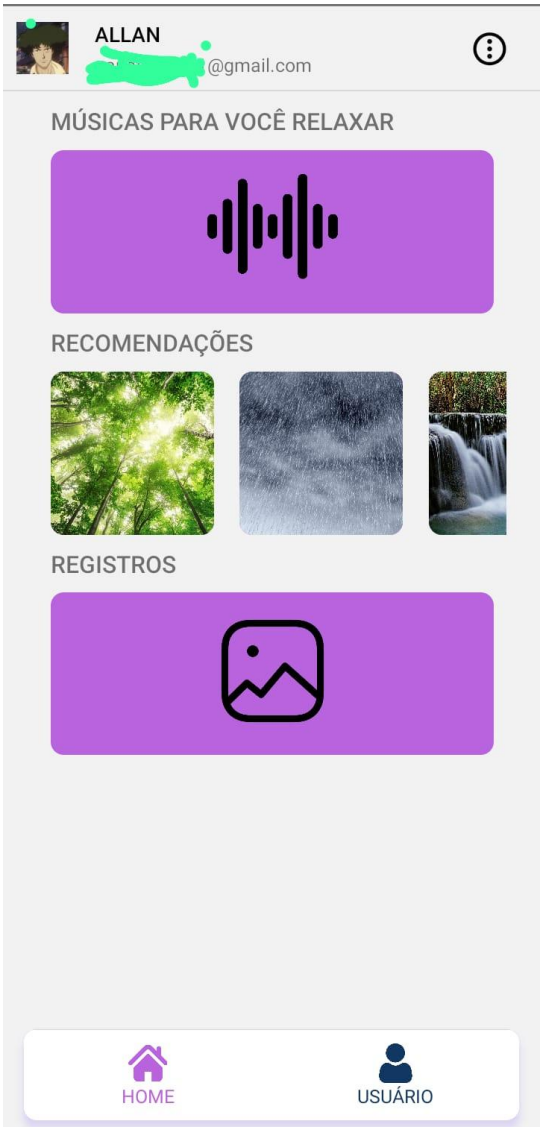
container: Este estilo é aplicado ao componente de nível superior (provavelmente a tela inteira). Ele configura o fundo para uma cor roxa (#BA63DF), alinha os itens no centro verticalmente e horizontalmente, define um espaçamento interno de 20 unidades e um espaçamento vertical entre elementos de 30 unidades. Isso cria um layout centralizado e espaçado.

imgAlz: Este estilo é aplicado a uma imagem. Ele define a largura e altura da imagem como 100% da largura e 50% da altura do contêiner, o que a torna cobrir metade da altura do contêiner. A propriedade resizeMode está definida como 'cover', o que permite que a imagem preencha todo o espaço disponível, mantendo sua proporção. Além disso, é aplicado um arredondamento das bordas da imagem com borderRadius: 10.

textSlider: Este estilo é aplicado a textos. Define a cor do texto como um tom de cinza (#b5b5b5) e centraliza o texto horizontalmente (por meio de justifyContent: 'center', que normalmente se aplica a contêineres, mas não afeta textos).

title: Este estilo é aplicado a um elemento de texto (provavelmente um título). Ele define o tamanho da fonte como 30 unidades, a cor do texto como branco (#fff), um fundo roxo escuro (#611A83) com arredondamento das bordas, um espaçamento interno de 10 unidades e, portanto, uma aparência de botão.

6° Tela: Home



Para a criação dessa tela foi utilizado o seguinte código:

```
import * as React from 'react';
import { Button, Text, View, Image, StyleSheet, Alert } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import { useNavigation } from '@react-navigation/native';
import { ScrollView, SectionList } from 'native-base';
import BottomSheet from './../../components/input/BottomSheet'
import { GoogleSignin } from '@react-native-google-signin/google-signin';
import auth from '@react-native-firebase/auth';
import { TouchableOpacity } from 'react-native';
import { Header } from '../../components/header';
import { Profile } from '../Profile';
import { ToggleButton } from 'react-native-paper';
import { Music } from '../MusicScreen';
```

```
const navigation = useNavigation();

const DATA = [
  {
    title: 'Main dishes',
    data: ['Pizza', 'Burger', 'Risotto'],
  },
  {
    title: 'Sides',
    data: ['French Fries', 'Onion Rings', 'Fried Shrimps'],
  },
  {
    title: 'Drinks',
    data: ['Water', 'Coke', 'Beer'],
  },
  {
    title: 'Desserts',
    data: ['Cheese Cake', 'Ice Cream'],
  },
];
function goToMusicScreen(){
  navigation.navigate('music')
}
function goToRainScreen(){
  navigation.navigate('rainS')
}
function goToForestScreen(){
  navigation.navigate('forestS')
}
function goToOceanScreen(){
  navigation.navigate('oceanS')
}
function goToCachoeiraScreen(){
  navigation.navigate('cachoeiraS')
}
function HomeScreen() {

  return (
    <ScrollView style={styles.containerHome}>
      <ScrollView style={styles.container}>
      <Text style={styles.textSection}>Músicas para você relaxar</Text>
        <TouchableOpacity style={styles.button} onPress={goToMusicScreen}>
            <Image
            style={styles.Pic}
            source={require('../../assets/music.png')}
          />
          </TouchableOpacity>
      <Text style={styles.textSection}>Recomendações</Text>
      <ScrollView horizontal>
      <TouchableOpacity style={styles.buttonR} onPress={goToForestScreen}>
          <Image
          style={styles.PicR}
          source={require('../../assets/natureza.jpg')}
          />
```

```
        </TouchableOpacity>
        <TouchableOpacity style={styles.buttonR} onPress={goToRainScreen}>
          <Image
          style={styles.PicR}
          source={require('../../assets/chuva.jpg')}
          />

        </TouchableOpacity>

        <TouchableOpacity style={styles.buttonR} onPress={ goToCachoeiraScreen}>
          <Image
          style={styles.PicR}
          source={require('../../assets/cachoeira.jpg')}
          />
        </TouchableOpacity>
          <TouchableOpacity style={styles.buttonR} onPress={ goToOceanScreen}>
          <Image
          style={styles.PicR}
          source={require('../../assets/fundoDoMar.jpg')}
          />
          </TouchableOpacity>
      </ScrollView>

            <Text style={styles.textSection}>Registros</Text>
              <TouchableOpacity style={styles.button} onPress={() => Alert.alert('Em
desenvolvimento')}>
                <Image
                style={styles.Pic}
                source={require('../../assets/gallery.png')}
                />
              </TouchableOpacity>

    </ScrollView>
    </ScrollView>
  );
}
function DetailsScreen() {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Details!</Text>
    </View>
  );
}
function SettingsScreen() {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Settings!</Text>
      <Button
        title="Go to Details"
        onPress={() => navigation.navigate('Details')}
      />
    </View>
  );
}

function Games() {
  return (
```

```
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Settings!</Text>
    </View>
  );
}

function Galery() {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Galery!</Text>
    </View>
  );
}


const HomeStack = createNativeStackNavigator();

function HomeStackScreen() {
  return (
    <HomeStack.Navigator screenOptions={{ headerShown: false }}>
      <HomeStack.Screen
      name="ini"
      component={HomeScreen}
      />
      <HomeStack.Screen name="Details" component={DetailsScreen} />
    </HomeStack.Navigator>
  );
}

const SettingsStack = createNativeStackNavigator();


const Tab = createBottomTabNavigator();

export function Inicio() {
  return (

      <Tab.Navigator
      screenOptions={{
        tabBarStyle:{
          height: 70,
          position:'absolute',
          bottom:16,
          left:16,
          right:16,
          borderRadius:10,
          ...styles.shadow
        }

      }}

      >
        <Tab.Screen
        name="Home"
        component={HomeStackScreen}
        options={{
          headerShown:true,
```

```
          header:()=> (<Header/>),
          tabBarShowLabel:false,
          tabBarIcon:({focused}) => (
            <View style={styles.tabBtn}>
            <Image
            source={require('./../../assets/home.png')} style={styles.Icon}
tintColor={focused ? '#BA63DF' : '#113763'} />
            <Text
            style={{color: focused ? '#BA63DF' : '#113763'}}>HOME</Text>
            </View>)
         }}
        />
        <Tab.Screen
        name="Usuário"
        component={Profile}
        options={{
          headerShown:true,
          tabBarShowLabel:false,
          tabBarIcon:({focused}) => (
            <View style={styles.tabBtn}>
            <Image
            source={require('./../../assets/user.png')} style={styles.Icon}
tintColor={focused ? '#BA63DF' : '#113763'} />
            <Text
            style={{color: focused ? '#BA63DF' : '#113763'}}>USUÁRIO</Text>
            </View>)
         }}
        />
      </Tab.Navigator>

  );
}

export const styles = StyleSheet.create({
  textSection: {
    fontSize:18,
    fontWeight:'600',
    textTransform:'uppercase',
    marginLeft:10
  },
  Pic:{
    width:100,
    height:100,
    borderRadius:10,
  },
  PicR:{
    width:125,
    height:125,
    borderRadius:10,
  },
  button: {
    alignItems: 'center',
    backgroundColor:'#191970',
    borderRadius:10,
    padding:10,
    height:125,
    margin:10
```

```
  },buttonR: {
    alignItems: 'center',
    backgroundColor:'#191970',
    borderRadius:10,
    height:125,
    margin:10
  },
  txt:{
      fontSize:18,
      fontWeight:'500'
  },
  containerHome:{
    padding:10
  },
  Icon:{
    width:30,
    height:30,

  },
  tabBtn:{
    alignItems:'center',
    justifyContent:'center'
  },
  shadow:{
    shadowColor: '#7f5df0',
    shadowOffset:{
      width:0,
      height:10,
    },
    shadowOpacity:0.25,
    shadowRadius:3.5,
    elevation:5
  },
  container: {
    flex: 1,
    marginHorizontal: 16
  },
  item: {
    backgroundColor: '#f9c2ff',
    padding: 20,
    marginVertical: 8
  },
  header: {
    fontSize: 32,
    backgroundColor: '#fff'
  },
  title: {
    fontSize: 24,
    paddingLeft:20
  }
})
```

Vamos entender o que fizemos:

1° Passo: As Importações

```
import * as React from 'react';
import { Button, Text, View, Image, StyleSheet, Alert } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import { useNavigation } from '@react-navigation/native';
import { ScrollView, SectionList } from 'native-base';
import BottomSheet from './../../components/input/BottomSheet'
import { GoogleSignin } from '@react-native-google-signin/google-signin';
import auth from '@react-native-firebase/auth';
import { TouchableOpacity } from 'react-native';
import { Header } from '../../components/header';
import { Profile } from '../Profile';
import { ToggleButton } from 'react-native-paper';
import { Music } from '../MusicScreen';
```

2° Passo: O Código

```
const navigation = useNavigation();

const DATA = [
  {
    title: 'Main dishes',
    data: ['Pizza', 'Burger', 'Risotto'],
  },
  {
    title: 'Sides',
    data: ['French Fries', 'Onion Rings', 'Fried Shrimps'],
  },
  {
    title: 'Drinks',
    data: ['Water', 'Coke', 'Beer'],
  },
  {
    title: 'Desserts',
    data: ['Cheese Cake', 'Ice Cream'],
  },
];
function goToMusicScreen(){
  navigation.navigate('music')
}
function goToRainScreen(){
  navigation.navigate('rainS')
}
function goToForestScreen(){
  navigation.navigate('forestS')
}
function goToOceanScreen(){
```

```
  navigation.navigate('oceanS')
}
function goToCachoeiraScreen(){
  navigation.navigate('cachoeiraS')
}
function HomeScreen() {

  return (
    <ScrollView style={styles.containerHome}>
      <ScrollView style={styles.container}>
      <Text style={styles.textSection}>Músicas para você relaxar</Text>
        <TouchableOpacity style={styles.button} onPress={goToMusicScreen}>
            <Image
            style={styles.Pic}
            source={require('../../assets/music.png')}
          />
          </TouchableOpacity>
      <Text style={styles.textSection}>Recomendações</Text>
      <ScrollView horizontal>
      <TouchableOpacity style={styles.buttonR} onPress={goToForestScreen}>
          <Image
          style={styles.PicR}
          source={require('../../assets/natureza.jpg')}
          />

        </TouchableOpacity>
        <TouchableOpacity style={styles.buttonR} onPress={goToRainScreen}>
          <Image
          style={styles.PicR}
          source={require('../../assets/chuva.jpg')}
          />

        </TouchableOpacity>

        <TouchableOpacity style={styles.buttonR} onPress={ goToCachoeiraScreen}>
          <Image
          style={styles.PicR}
          source={require('../../assets/cachoeira.jpg')}
          />
        </TouchableOpacity>
          <TouchableOpacity style={styles.buttonR} onPress={ goToOceanScreen}>
          <Image
          style={styles.PicR}
          source={require('../../assets/fundoDoMar.jpg')}
          />
          </TouchableOpacity>
      </ScrollView>

            <Text style={styles.textSection}>Registros</Text>
              <TouchableOpacity style={styles.button} onPress={() => Alert.alert('Em
desenvolvimento')}>
                <Image
                style={styles.Pic}
                source={require('../../assets/gallery.png')}
                />
              </TouchableOpacity>
```

```
    </ScrollView>
    </ScrollView>
  );
}
function DetailsScreen() {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Details!</Text>
    </View>
  );
}
function SettingsScreen() {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Settings!</Text>
      <Button
        title="Go to Details"
        onPress={() => navigation.navigate('Details')}
      />
    </View>
  );
}

function Games() {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Settings!</Text>
    </View>
  );
}

function Galery() {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Galery!</Text>
    </View>
  );
}


const HomeStack = createNativeStackNavigator();

function HomeStackScreen() {
  return (
    <HomeStack.Navigator screenOptions={{ headerShown: false }}>
      <HomeStack.Screen
      name="ini"
      component={HomeScreen}
      />
      <HomeStack.Screen name="Details" component={DetailsScreen} />
    </HomeStack.Navigator>
  );
}

const SettingsStack = createNativeStackNavigator();
```

```
const Tab = createBottomTabNavigator();

export function Inicio() {
  return (

      <Tab.Navigator
      screenOptions={{
        tabBarStyle:{
          height: 70,
          position:'absolute',
          bottom:16,
          left:16,
          right:16,
          borderRadius:10,
          ...styles.shadow
        }

      }}

      >
        <Tab.Screen
        name="Home"
        component={HomeStackScreen}
        options={{
          headerShown:true,
          header:()=> (<Header/>),
          tabBarShowLabel:false,
          tabBarIcon:({focused}) => (
            <View style={styles.tabBtn}>
            <Image
            source={require('./../../assets/home.png')} style={styles.Icon}
tintColor={focused ? '#BA63DF' : '#113763'} />
            <Text
            style={{color: focused ? '#BA63DF' : '#113763'}}>HOME</Text>
            </View>)
         }}
        />
        <Tab.Screen
        name="Usuário"
        component={Profile}
        options={{
          headerShown:true,
          tabBarShowLabel:false,
          tabBarIcon:({focused}) => (
            <View style={styles.tabBtn}>
            <Image
            source={require('./../../assets/user.png')} style={styles.Icon}
tintColor={focused ? '#BA63DF' : '#113763'} />
            <Text
            style={{color: focused ? '#BA63DF' : '#113763'}}>USUÁRIO</Text>
            </View>)
         }}
        />
      </Tab.Navigator>

  );
```

const navigation = useNavigation();: Isso usa o hook useNavigation do React Navigation para obter o objeto de navegação. Esse objeto de navegação é usado para navegar entre as telas do aplicativo.

const DATA = [...]: Aqui, você tem um array de dados DATA, que parece conter informações sobre diferentes seções de um aplicativo, como pratos principais, acompanhamentos, bebidas e sobremesas.

Funções de navegação (goToMusicScreen, goToRainScreen, etc.): Cada uma dessas funções utiliza o objeto de navegação para direcionar o usuário para telas específicas quando os botões associados são pressionados.

Componente HomeScreen: Este é um componente funcional que parece representar a tela inicial do seu aplicativo. Ele usa vários componentes, como ScrollView, TouchableOpacity, Text, e Image, para criar uma interface de usuário.

<ScrollView style={styles.containerHome}>: Este componente ScrollView permite que o conteúdo dentro dele seja rolado verticalmente. Parece que ele contém outros componentes que compõem a tela inicial.

<TouchableOpacity>: É um componente que detecta toques. Quando pressionado, ele executa a função associada, como onPress={goToMusicScreen}.

<ScrollView horizontal>: Um ScrollView que permite a rolagem horizontal. Ele contém uma série de TouchableOpacity que, quando pressionados, executam funções de navegação para diferentes telas.

createNativeStackNavigator: Esta função cria um navegador de pilha nativo, que é usado para gerenciar a navegação empilhada entre as telas.

createBottomTabNavigator: Cria um navegador de abas na parte inferior da tela. Parece que o aplicativo tem duas abas: "Home" e "Usuário" (Profile).

<Tab.Screen>: Configuração das telas dentro do navegador de abas. Ele especifica o componente associado a cada aba, opções de navegação e opções visuais, como ícones e rótulos.

Inicio: Esta é uma função que retorna o componente que representa a navegação principal do aplicativo. Parece ser o ponto de entrada para o seu aplicativo, onde as abas e as respectivas telas são configuradas.

3°Passo: Os Styles

```
export const styles = StyleSheet.create({
  textSection: {
    fontSize:18,
```

```
  fontWeight:'600',
  textTransform:'uppercase',
  marginLeft:10
},
Pic:{
  width:100,
  height:100,
  borderRadius:10,
},
PicR:{
  width:125,
  height:125,
  borderRadius:10,
},
button: {
  alignItems: 'center',
  backgroundColor:'#191970',
  borderRadius:10,
  padding:10,
  height:125,
  margin:10
},buttonR: {
  alignItems: 'center',
  backgroundColor:'#191970',
  borderRadius:10,
  height:125,
  margin:10
},
txt:{
    fontSize:18,
    fontWeight:'500'
},
containerHome:{
  padding:10
},
Icon:{
  width:30,
  height:30,

},
tabBtn:{
  alignItems:'center',
  justifyContent:'center'
},
shadow:{
  shadowColor: '#7f5df0',
  shadowOffset:{
    width:0,
    height:10,
  },
  shadowOpacity:0.25,
  shadowRadius:3.5,
  elevation:5
},
container: {
  flex: 1,
  marginHorizontal: 16
```

```
  },
  item: {
    backgroundColor: '#f9c2ff',
    padding: 20,
    marginVertical: 8
  },
  header: {
    fontSize: 32,
    backgroundColor: '#fff'
  },
  title: {
    fontSize: 24,
    paddingLeft:20
  }
})
```

textSection: Estilo para os textos das seções. Define o tamanho da fonte, peso da fonte, transformação de texto para maiúsculas e margem à esquerda.

Pic e PicR: Estilos para imagens (Pic para imagens menores e PicR para imagens maiores). Define largura, altura e borda arredondada (borda).

button e buttonR: Estilos para os botões. Ambos têm propriedades semelhantes, como alinhamento, cor de fundo, borda arredondada, altura, margem e preenchimento. A diferença parece estar no tamanho, onde button tem tamanho fixo e buttonR parece ter altura flexível.

txt: Estilo para textos. Define o tamanho da fonte e o peso da fonte.

containerHome: Estilo para o contêiner da tela inicial. Define o preenchimento.

Icon: Estilo para ícones. Define largura e altura.

tabBtn: Estilo para os botões das abas. Define alinhamento de itens e justificação de conteúdo.

shadow: Estilo para criar um efeito de sombra. Define a cor da sombra, o deslocamento da sombra, a opacidade da sombra, o raio da sombra e a elevação.

container: Estilo para o contêiner geral. Define flexibilidade, margem horizontal e estilo de item.

item: Estilo para itens dentro de um contêiner. Define a cor de fundo, o preenchimento e a margem vertical.

header: Estilo para cabeçalhos. Define o tamanho da fonte e a cor de fundo.

title: Estilo para títulos. Define o tamanho da fonte e o preenchimento à esquerda.

7° Tela: Perfil

Para a criação dessa tela foi utilizado o seguinte código:

```
import {
    StyleSheet,
    Text,
    View,
    SafeAreaView,
    SectionList,
    StatusBar,
    TouchableOpacity,
    ScrollView,
    Image
  } from 'react-native';
  import { createNativeStackNavigator } from "@react-navigation/native-stack";
  import {useNavigation} from '@react-navigation/native';
  import { GoogleSignin } from '@react-native-google-signin/google-signin';
  import auth from '@react-native-firebase/auth';
```

```
  const navigation = useNavigation();
  const stack = createNativeStackNavigator();

function goToHome(){
    navigation.navigate('home');
    navigation.reset({
        index: 0,
        routes: [{ name: 'home' }],
      });
  }
  function handleGoToEditProfile(){
    navigation.navigate('edit');
  }
  const signOut = async ()=>{
    try{
        let user = await GoogleSignin.isSignedIn()
        if(!user){
            auth().signOut();
            return goToHome()
        }
        await GoogleSignin.revokeAccess();
        await auth().signOut();
        goToHome()
    }catch(error){
        console.error(error);
    }
  }
  export function Profile(){

        let user = auth().currentUser
        var name = user?.displayName
        var email = user?.email
        let userPica = user?.photoURL
    return (
      <View style={styles.containerView}>
        <View style={styles.containerUser}>
          <TouchableOpacity>
            <Image
            style={styles.userPic}
            source={userPica != null?{uri: userPica} : require('../../assets/user.png')}
            />
          </TouchableOpacity>
          <View style={styles.txtUser}>
            <Text style={styles.txtName}>
                {name}
            </Text>
            <Text style={styles.txtEmail}>{email}</Text>
          </View>
        </View>

        <View style={styles.containerConfigs}>
          <Text style={styles.title}>Perfil</Text>
            <View style={styles.container}>

              <TouchableOpacity
```

```
              onPress={handleGoToEditProfile}
              >
                <Text style={styles.textBtn}>Editar perfil</Text>
              </TouchableOpacity>
              <TouchableOpacity>
                <Text style={styles.textBtn}>Notificações</Text>
              </TouchableOpacity>
              <TouchableOpacity >
                <Text style={styles.textBtn}>Idioma</Text>
              </TouchableOpacity>
            </View>
          <Text style={styles.title}>Sobre</Text>
            <View style={styles.container}>

              <TouchableOpacity>
                <Text style={styles.textBtn}>Tema</Text>
              </TouchableOpacity>
              <TouchableOpacity>
                <Text style={styles.textBtn}>Informações</Text>
              </TouchableOpacity>
              <TouchableOpacity>
                <Text style={styles.textBtn}>Contatar</Text>
              </TouchableOpacity>
            </View>
          <Text style={styles.title}>Outros</Text>
          <View style={styles.container}>

            <TouchableOpacity>
              <Text style={styles.textBtn}>Relatar problema</Text>
            </TouchableOpacity>
            <TouchableOpacity>
              <Text style={styles.textBtn}>Política</Text>
            </TouchableOpacity>
            <TouchableOpacity
            onPress={
                signOut
            }
            style={styles.logoutBtn}>
              <Text style={styles.textBtn}>Desconectar</Text>
            </TouchableOpacity>
          </View>


        </View>
    </View>
    )

};
  const styles = StyleSheet.create({
    container: {
      borderRadius:10,
      padding:10,
      gap:5,
      backgroundColor:'rgba(36, 39, 96, 0.05)'
    },
    containerView:{
      padding:20,
```

```
    gap:30
  },
  containerUser:{

    justifyContent:'center',
    alignItems:'center',
    gap:20
  },
  txtUser:{
      justifyContent:'center',
      alignItems:'center'
  },
  txtName:{
      fontSize:18,
      fontWeight:'500'
  },
  containerConfigs:{
    gap:10,

  },
  userPic:{
    width:75,
    height:75
  },
  title:{
    fontSize:20
  },
  textBtn:{
    fontWeight:'500',
    fontSize:16,
    color:'black'
  },
});
```

Vamos entender o que fizemos:

1° Passo: As Importações

```
import {
  StyleSheet,
  Text,
  View,
  SafeAreaView,
  SectionList,
  StatusBar,
  TouchableOpacity,
  ScrollView,
  Image

} from 'react-native';
```

2° Passo: O Código

```
const App = () => (

    <View style={styles.containerView}>
      <View style={styles.containerUser}>
        <TouchableOpacity>
          <Image
          style={styles.userPic}
          source={require('user.png')}
          />
        </TouchableOpacity>
        <View style={styles.txtUser}>
          <Text>Allan Matheus</Text>
          <Text>allanMalvadeza@gmail.com</Text>
        </View>
      </View>

      <View style={styles.containerConfigs}>
        <Text style={styles.title}>Perfil</Text>
          <View style={styles.container}>

            <TouchableOpacity>
              <Text style={styles.textBtn}>Editar perfil</Text>
            </TouchableOpacity>
            <TouchableOpacity>
              <Text style={styles.textBtn}>Notificações</Text>
            </TouchableOpacity>
            <TouchableOpacity >
              <Text style={styles.textBtn}>Idioma</Text>
            </TouchableOpacity>
          </View>
        <Text style={styles.title}>Sobre</Text>
          <View style={styles.container}>

            <TouchableOpacity>
              <Text style={styles.textBtn}>Tema</Text>
            </TouchableOpacity>
            <TouchableOpacity>
              <Text style={styles.textBtn}>Informações</Text>
            </TouchableOpacity>
            <TouchableOpacity>
              <Text style={styles.textBtn}>Contatar</Text>
            </TouchableOpacity>
          </View>
        <Text style={styles.title}>Outros</Text>
        <View style={styles.container}>

          <TouchableOpacity>
            <Text style={styles.textBtn}>Relatar problema</Text>
          </TouchableOpacity>
          <TouchableOpacity>
            <Text style={styles.textBtn}>Política</Text>
          </TouchableOpacity>
          <TouchableOpacity style={styles.logoutBtn}>
            <Text style={styles.textBtn}>Desconectar</Text>
          </TouchableOpacity>
        </View>
```

```
    </View>
  </View>

);
```

const App = () => (: Define um componente de função chamado App. Este componente retorna a estrutura da interface do usuário usando JSX.

<View style={styles.containerView}>: Cria um componente de contêiner (View) que aplica estilos definidos em styles.containerView. Este contêiner envolve todo o conteúdo da tela.

<View style={styles.containerUser}>: Cria um contêiner para informações do usuário, como imagem de perfil e detalhes do usuário.

<TouchableOpacity>: Componente de toque que pode ser usado para envolver outros componentes e adicionar uma interação de toque. No entanto, neste caso, não está associado a uma função de pressionar.

<Image>: Exibe uma imagem de perfil do usuário. A propriedade source aponta para a localização da imagem, neste caso, é esperado que esteja em algum lugar no projeto com o nome "user.png".

<View style={styles.txtUser}>: Cria um contêiner para o texto do usuário.

<Text>Allan Matheus</Text>: Exibe o nome do usuário.

<Text>allanMalvadeza@gmail.com</Text>: Exibe o endereço de e-mail do usuário.

<View style={styles.containerConfigs}>: Cria um contêiner para as configurações do perfil, dividido em seções (Perfil, Sobre, Outros).

<Text style={styles.title}>Perfil</Text>: Exibe um título para a seção de perfil.

<View style={styles.container}>: Cria um contêiner para as opções dentro de uma seção.

<TouchableOpacity>: Cria uma opção (Editar perfil) com capacidade de interação de toque.

<Text style={styles.textBtn}>: Exibe o texto da opção.

Repete o padrão para outras opções em diferentes seções (Sobre e Outros).

<TouchableOpacity style={styles.logoutBtn}>: Cria um botão de desconectar com um estilo específico (styles.logoutBtn).

<Text style={styles.textBtn}>Desconectar</Text>: Exibe o texto "Desconectar" dentro do botão.

Fechamento de Tags e Parênteses: Finaliza as tags JSX e a função App.

3°Passo: Os Styles

```
const styles = StyleSheet.create({
  container: {
    borderRadius:10,
    padding:10,
    gap:5,
    backgroundColor:'rgba(36, 39, 96, 0.05)'
  },
  containerView:{
    padding:20,
    gap:20
  },
  containerUser:{
    flexDirection:'row',
    justifyContent:'center',
    alignItems:'center',
    gap:20
  },
  containerConfigs:{
    gap:10
  },
  userPic:{
    width:75,
    height:75
  },
  title:{
    fontSize:20
  },
  textBtn:{
    fontWeight:'500'
  },
});

export default App;
```

container: Este estilo é aplicado ao contêiner que envolve as opções de perfil. As propriedades incluem:

borderRadius: 10: Adiciona bordas arredondadas ao contêiner.

padding: 10: Adiciona um preenchimento interno de 10 unidades.

gap: 5: Define um espaçamento entre os filhos do contêiner.

backgroundColor: 'rgba(36, 39, 96, 0.05)': Define uma cor de fundo com um efeito de transparência.

containerView: Este estilo é aplicado ao contêiner principal que envolve toda a tela. As propriedades incluem:

padding: 20: Adiciona um preenchimento interno de 20 unidades.

gap: 20: Define um espaçamento entre os filhos do contêiner.

containerUser: Este estilo é aplicado ao contêiner que envolve as informações do usuário. As propriedades incluem:

flexDirection: 'row': Estabelece um layout em linha para os elementos filhos.

justifyContent: 'center': Centraliza os elementos horizontalmente.

alignItems: 'center': Centraliza os elementos verticalmente.

gap: 20: Define um espaçamento entre os filhos do contêiner.

containerConfigs: Este estilo é aplicado ao contêiner que envolve as configurações do perfil. A propriedade inclui:

gap: 10: Define um espaçamento entre os filhos do contêiner.

userPic: Este estilo é aplicado à imagem de perfil do usuário. As propriedades incluem:

width: 75: Define a largura da imagem.

height: 75: Define a altura da imagem.

title: Este estilo é aplicado aos títulos das seções. A propriedade inclui:

fontSize: 20: Define o tamanho da fonte para 20 unidades.

textBtn: Este estilo é aplicado ao texto das opções. A propriedade inclui:

fontWeight: '500': Define o peso da fonte como 500.

8° Tela: Edição de Perfil

Para a criação dessa tela foi utilizado o seguinte código:

```
import {
    ScrollView,
    View,
    FlatList,
    TouchableOpacity,
    StyleSheet,
    Text,
    StatusBar,
    Image,
    Alert,
  } from 'react-native';
  import { Button, List, useTheme,TextInput } from 'react-native-paper';
  import { createNativeStackNavigator } from "@react-navigation/native-stack";
  import {useNavigation} from '@react-navigation/native';
  import auth from '@react-native-firebase/auth';
import { useState } from 'react';
import firestore from '@react-native-firebase/firestore';
import { launchImageLibrary, launchCamera } from "react-native-image-picker";

  const navigation = useNavigation();
  const stack = createNativeStackNavigator();



  export function Edit() {
    return (
        <stack.Navigator screenOptions={{ headerShown: true }}>
            <stack.Screen name="Perfil" component={EditProfile} />
        </stack.Navigator>
    );
  }
   function EditProfile(){
    const [fullName, setFname] = useState('');
    const [email, setEmail] = useState('');
    const [celular, setCelular] = useState('');
    const [birthday, setBirth] = useState('');
    const [genero, setGenero] = useState('');

   function updateDateUser(){
    if(fullName == '' && email == ''){
      Alert.alert('Preencha seus dados.')
      return;
    }

    firestore().collection('users')
              .doc(auth().currentUser.uid)
              .set({
                  fullName: fullName,
                  email: email,
                  celular: celular,
                  dataDeAnivesario: birthday,
                  genero: genero
              },{merge: true}).then(()=>{
                Alert.alert('Informações atualizadas.')
                navigation.goBack()
              })
              .catch((error)=>{
```

```
                Alert.alert(error.message)

            })
 }

 const [selectedImage, setSelectedImage] = useState("");
const [userImage, setImage] = useState(false);


const openImagePicker = () => {
  const options = {
    mediaType: "photo",
    includeBase64: false,
    maxHeight: 2000,
    maxWidth: 2000
  };

  launchImageLibrary(options, (response) => {
    if (response.didCancel) {
      console.log("User cancelled image picker");
    } else if (response.error) {
      console.log("Image picker error: ", response.error);
    } else {
      let imageUri = response.uri || response.assets?.[0]?.uri;
      setSelectedImage(imageUri);
      setImage(true);
    }
  });

};

  return (

    <View style={styles.container}>
     <TouchableOpacity
    onPress={openImagePicker}
     style={styles.button} >
     <Image

     source={
      userImage == true
        ? { uri: selectedImage }
        : require("../../assets/user.png")
    }
    style={styles.userPic}
    resizeMode="contain"
      />

      </TouchableOpacity>

        <View style={styles.formContainer}>

          <Text style={styles.label}>Nome completo</Text>
          <TextInput
          value={fullName}
          onChangeText={value => setFname(value)}
          style={styles.input}
```

```
          />

          <Text style={styles.label}>Email</Text>
          <TextInput
          value={email}
          style={styles.input}
          onChangeText={value => setEmail(value)}
           keyboardType="email-address"
          />

          <Text style={styles.label}>Celular</Text>
          <TextInput
          value={celular}
          style={styles.input}
          onChangeText={value => setCelular(value)}
           keyboardType="numeric"
          />

          <Text style={styles.label}>Data de nascimento</Text>
          <TextInput
          value={birthday}
          style={styles.input}
          onChangeText={value => setBirth(value)}
           keyboardType="numeric"
          />

          <Text style={styles.label}>Gênero</Text>
          <TextInput
          value={genero}
          onChangeText={value => setGenero(value)}
          style={styles.input}
          />
        </View>

      <TouchableOpacity
        onPress={updateDateUser}
        style={styles.buttonSave}>

        <Text style={styles.txt}>SALVAR</Text>
      </TouchableOpacity>


    </View>

  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    marginTop: StatusBar.currentHeight || 0,
    padding:10,
    gap:35
  },
  formContainer: {
    gap:5,
    padding:10,
```

```
    },
    userPic:{
      width:125,
      height:125,
      borderRadius:1000
    },
    button: {
      alignItems: 'center',

    },
    buttonSave: {
      alignItems: 'center',
      backgroundColor: '#BA63DF',
      padding:15,
      borderRadius:5,
    },
    txt:{
        fontWeight:'500',
        fontSize:18,
        color:'#fff'
    },
    input:{
      borderBottomWidth:2,
      borderBottomColor: '#BA63DF',
      backgroundColor:'#DCDCDC',
      borderTopLeftRadius:3,
      borderTopRightRadius:3,
      height:45
    },
    label:{
      fontWeight:'700',
      fontSize:16,
    }
  });

function getAuth() {
    throw new Error('Function not implemented.');
}
```

Vamos entender o que fizemos:

1° Passo: As Importações

```
import {
  ScrollView,
  View,
  FlatList,
  TouchableOpacity,
  StyleSheet,
  Text,
  StatusBar,
  Image,
  TextInput
} from 'react-native';
```

2° Passo: O Código

```
const App = () => {

  return (
    <View style={styles.container}>
     <TouchableOpacity style={styles.button} >
       <Image
       style={styles.userPic}
       source={require('user.png')}
      />
     </TouchableOpacity>

       <View style={styles.formContainer}>

         <Text style={styles.label}>Nome completo</Text>
         <TextInput
         style={styles.input}
         />

         <Text style={styles.label}>Email</Text>
         <TextInput
         style={styles.input}
          keyboardType="email-address"
         />

         <Text style={styles.label}>Celular</Text>
         <TextInput
         style={styles.input}
          keyboardType="numeric"
         />

         <Text style={styles.label}>Data de nascimento</Text>
         <TextInput
         style={styles.input}
          keyboardType="numeric"
         />

         <Text style={styles.label}>Gênero</Text>
         <TextInput
         style={styles.input}
         />
       </View>

      <TouchableOpacity
        style={styles.buttonSave}>
        <Text style={styles.txt}>SALVAR</Text>
```

```
      </TouchableOpacity>


    </View>
  );
};
```

View (linha 4): Isso cria um contêiner de nível superior que agrupa todos os elementos na interface.

`TouchableOpacity` (linha 5): Este é um botão

Text (linhas 12, 16, 20, 24, 28): Componentes de texto que atuam como rótulos para os campos do formulário, indicando o que é esperado em cada campo.

TextInput (linhas 14, 18, 22, 26, 30): Componentes de entrada de texto que permitem que o usuário digite informações nos campos do formulário.

TouchableOpacity (linha 34): Um botão "SALVAR" que o usuário pode pressionar para salvar as informações inseridas no formulário.

3°Passo: Os Styles

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    marginTop: StatusBar.currentHeight || 0,
    padding:20,
    gap:15
  },
  formContainer: {
    gap:5,
  },
  userPic:{
    width:125,
    height:125,
    borderRadius:1000
  },
  button: {
    alignItems: 'center',

  },
  buttonSave: {
    alignItems: 'center',
    backgroundColor: '#BA63DF',
    padding:7,
    borderRadius:5,
```

```
  },
  txt:{
      fontSize:16,
      fontWeight:'500'
  },
  input:{
    padding:5,
    borderBottomWidth:2,
    borderBottomColor: '#BA63DF',
    backgroundColor:'#DCDCDC',
    borderTopLeftRadius:3,
    borderTopRightRadius:3,
  },
  label:{
    fontWeight:'700'
  }
});

export default App;
```

container: Este estilo é aplicado ao contêiner principal que envolve a tela. As propriedades incluem:

flex: 1: Faz com que o contêiner ocupe todo o espaço disponível.

marginTop: StatusBar.currentHeight || 0: Garante que o conteúdo não seja ocultado pela barra de status no topo da tela.

padding: 20: Adiciona um preenchimento interno de 20 unidades.

gap: 15: Define um espaçamento entre os filhos do contêiner.

formContainer: Este estilo é aplicado ao contêiner que envolve elementos do formulário. A propriedade inclui:

gap: 5: Define um espaçamento entre os filhos do contêiner.

userPic: Este estilo é aplicado à imagem de perfil do usuário. As propriedades incluem:

width: 125: Define a largura da imagem.

height: 125: Define a altura da imagem.

borderRadius: 1000: Define um grande raio de borda para criar um formato de círculo. (O valor 1000 é suficientemente grande para tornar a borda circular independentemente do tamanho da imagem).

button: Este estilo é aplicado aos botões gerais. As propriedades incluem:

alignItems: 'center': Centraliza os itens horizontalmente no botão.

buttonSave: Este estilo é aplicado ao botão de salvar. As propriedades incluem:

alignItems: 'center': Centraliza os itens horizontalmente no botão.

backgroundColor: '#BA63DF': Define uma cor de fundo específica para o botão de salvar.

padding: 7: Adiciona um preenchimento interno de 7 unidades.

borderRadius: 5: Adiciona bordas arredondadas ao botão.

txt: Este estilo é aplicado ao texto em geral. As propriedades incluem:

fontSize: 16: Define o tamanho da fonte como 16 unidades.

fontWeight: '500': Define o peso da fonte como 500.

input: Este estilo é aplicado aos campos de entrada (inputs) do formulário. As propriedades incluem:

padding: 5: Adiciona um preenchimento interno de 5 unidades.

borderBottomWidth: 2: Adiciona uma borda na parte inferior com uma largura de 2 unidades.

borderBottomColor: '#BA63DF': Define a cor da borda inferior.

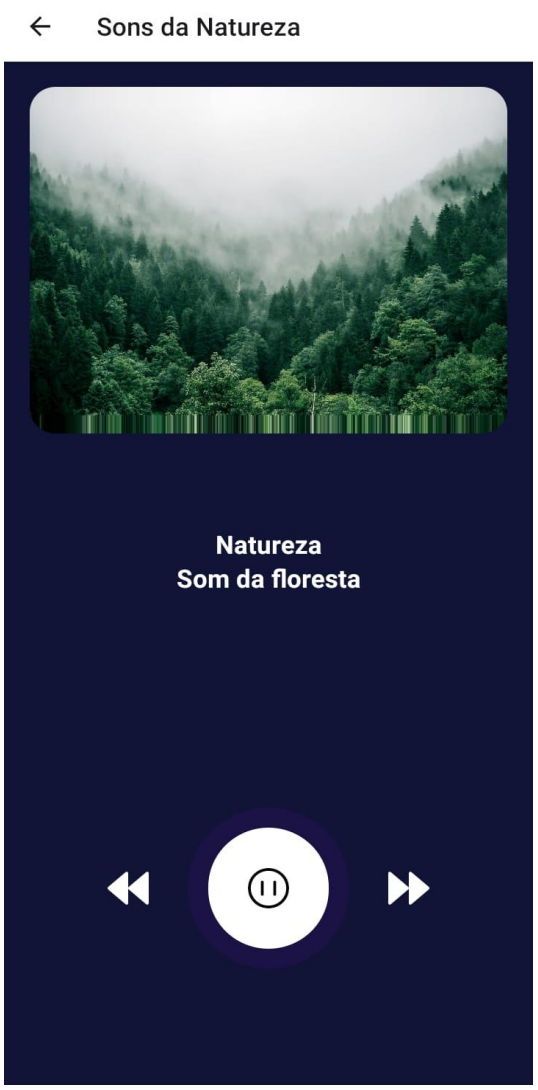
backgroundColor: '#DCDCDC': Define a cor de fundo do campo de entrada.

borderTopLeftRadius: 3 e borderTopRightRadius: 3: Adiciona bordas arredondadas à parte superior do campo de entrada.

label: Este estilo é aplicado aos rótulos (labels) do formulário. A propriedade inclui:

fontWeight: '700': Define o peso da fonte como 700.

9° Tela: Musicoterapia



Para a criação dessa tela foi utilizado o seguinte código:

```
import React, {useRef, useState} from 'react';
import {StatusBar, StyleSheet, Text, View} from 'react-native';
import Video from 'react-native-video';
import { AlbumArt } from '../../components/AlbumArt';
import Control from '../../components/Control';
import SongDetails from '../../components/SngDetails';
import { TRACKS } from '../../components/Tracks';
import { createNativeStackNavigator } from '@react-navigation/native-stack';


const stack = createNativeStackNavigator();

export function ForestScreen() {
```

```
    return (
        <stack.Navigator screenOptions={{ headerShown: true }}>
            <stack.Screen name="Sons da Natureza" component={ForestMusic} />
        </stack.Navigator>
    );
  }

function ForestMusic() {
  const [pause, setPause] = useState(false);
  const [selectedTrack, setSelectedTrack] = useState(0);

  const currentTrack = TRACKS[selectedTrack];

  function onPlay() {
    setPause(false);
  }
  function onPause() {
    setPause(true);
  }
  function onNext() {
    if (selectedTrack == TRACKS.length - 1) {
      setSelectedTrack(0);
    } else {
      setSelectedTrack(selectedTrack + 1);
    }
  }

  function onBack() {
    if (selectedTrack == 0) {
      setSelectedTrack(TRACKS.length - 1);
    } else {
      setSelectedTrack(selectedTrack - 1);
    }
  }

  return (
    <>

      <View style={styles.container}>
        <AlbumArt url={currentTrack.albumArtUrl} />
        <View
          style={{
            flexDirection: 'column',
            justifyContent: 'space-around',
            flex: 1,
          }}>
          <SongDetails
            artistName={currentTrack.artist}
            songName={currentTrack.title}
          />
          <Control {...{pause, onPause, onPlay, onNext, onBack}} />
        </View>

        <Video

          source={require('./../../assets/Forest.mp3')}
          paused={pause}
```

```
          audioOnly
          poster={currentTrack.albumArtUrl}
        />
      </View>
    </>
  );
}

const styles = StyleSheet.create({
  container: {
    backgroundColor: '#111436',
    flex: 1,
    padding:20
  },
});
```

Vamos entender o que fizemos:

1° Passo: As Importações

```
import React, {useRef, useState} from 'react';
import {StatusBar, StyleSheet, Text, View} from 'react-native';
import Video from 'react-native-video';
import { AlbumArt } from '../../components/AlbumArt';
import Control from '../../components/Control';
import SongDetails from '../../components/SngDetails';
import { TRACKS } from '../../components/Tracks';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
```

2° Passo: O Código

```
const stack = createNativeStackNavigator();

export function ForestScreen() {
    return (
        <stack.Navigator screenOptions={{ headerShown: true }}>
            <stack.Screen name="Sons da Natureza" component={ForestMusic} />
        </stack.Navigator>
    );
  }

function ForestMusic() {
  const [pause, setPause] = useState(false);
  const [selectedTrack, setSelectedTrack] = useState(0);

  const currentTrack = TRACKS[selectedTrack];

  function onPlay() {
    setPause(false);
  }
  function onPause() {
    setPause(true);
```

```
  }
  function onNext() {
    if (selectedTrack == TRACKS.length - 1) {
      setSelectedTrack(0);
    } else {
      setSelectedTrack(selectedTrack + 1);
    }
  }

  function onBack() {
    if (selectedTrack == 0) {
      setSelectedTrack(TRACKS.length - 1);
    } else {
      setSelectedTrack(selectedTrack - 1);
    }
  }

  return (
    <>

      <View style={styles.container}>
        <AlbumArt url={currentTrack.albumArtUrl} />
        <View
          style={{
            flexDirection: 'column',
            justifyContent: 'space-around',
            flex: 1,
          }}>
          <SongDetails
            artistName={currentTrack.artist}
            songName={currentTrack.title}
          />
          <Control {...{pause, onPause, onPlay, onNext, onBack}} />
        </View>

        <Video

          source={require('./../../assets/Forest.mp3')}
          paused={pause}
          audioOnly
          poster={currentTrack.albumArtUrl}
        />
      </View>
    </>
  );
```

<View style={styles.container}>: Cria um contêiner principal que envolve todos os elementos da tela. O estilo é definido pela folha de estilos styles.container.

<AlbumArt url={currentTrack.albumArtUrl} />: Renderiza um componente chamado AlbumArt e passa a propriedade url com a URL da capa do álbum atual (currentTrack.albumArtUrl). Presumivelmente, este componente renderiza a capa do álbum.

<View> com controles de reprodução e detalhes da música:

A propriedade flexDirection: 'column' estabelece um layout de coluna para os elementos internos.
justifyContent: 'space-around' distribui o espaço ao redor dos elementos filhos, dando a eles um espaçamento uniforme.
flex: 1 faz com que este contêiner ocupe todo o espaço disponível.
<SongDetails artistName={currentTrack.artist} songName={currentTrack.title} />: Renderiza um componente chamado SongDetails e passa as propriedades artistName e songName com informações sobre o artista e o título da música atual.

<Control {...{pause, onPause, onPlay, onNext, onBack}} />: Renderiza um componente chamado Control e passa várias funções de controle de reprodução como props, bem como o estado pause. Isso sugere que o componente Control pode conter botões ou elementos para pausar, retomar, avançar e retroceder a reprodução da música.

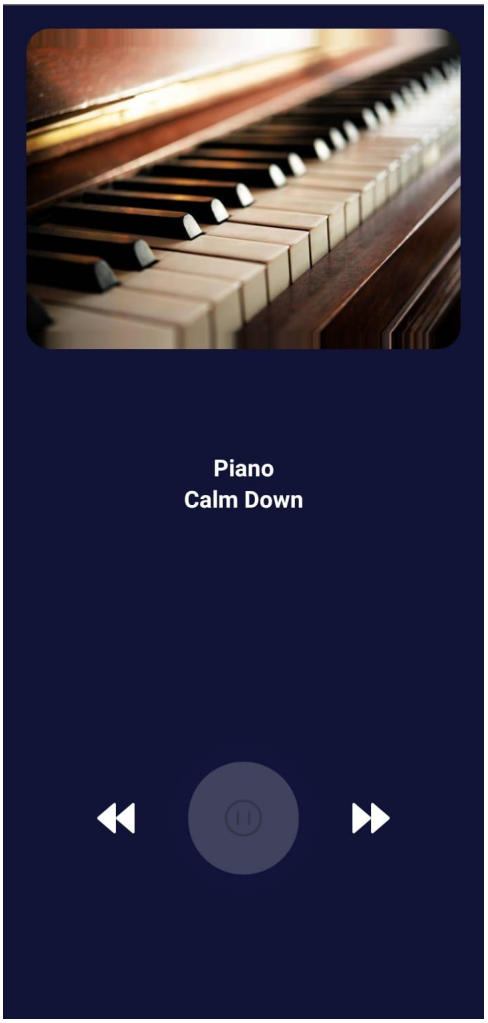
<Video>: Renderiza um componente de vídeo. Parece estar usando a biblioteca react-native-video ou algo similar. Algumas propriedades notáveis incluem:

source: Define a fonte do vídeo. No caso, é um arquivo de áudio (Forest.mp3) localizado em '../../../assets/'.
paused={pause}: Controla se o vídeo está pausado ou em reprodução, com base no estado pause.
audioOnly: Indica que o vídeo é apenas uma faixa de áudio.
poster={currentTrack.albumArtUrl}: Define a imagem de pôster do vídeo, que parece ser a capa do álbum

3°Passo: Os Style

```
const styles = StyleSheet.create({
  container: {
    backgroundColor: '#111436',
    flex: 1,
    padding:20
  },
});
```

10° Tela: Musicoterapia - Tela Principal



Para a criação dessa tela foi utilizado o seguinte código:

```
import React, {useRef, useState} from 'react';
import {StatusBar, StyleSheet, Text, View} from 'react-native';
import Video from 'react-native-video';
import { AlbumArt } from '../../components/AlbumArt';
import Control from '../../components/Control';
import SongDetails from '../../components/SngDetails';
import { TRACKS } from '../../components/Tracks';

export function Music() {
  const [pause, setPause] = useState(false);
  const [selectedTrack, setSelectedTrack] = useState(0);

  const currentTrack = TRACKS[selectedTrack];

  function onPlay() {
    setPause(false);
  }
  function onPause() {
    setPause(true);
  }
  function onNext() {
    if (selectedTrack == TRACKS.length - 1) {
```

```
      setSelectedTrack(0);
    } else {
      setSelectedTrack(selectedTrack + 1);
    }
  }

  function onBack() {
    if (selectedTrack == 0) {
      setSelectedTrack(TRACKS.length - 1);
    } else {
      setSelectedTrack(selectedTrack - 1);
    }
  }

  return (
    <>

      <View style={styles.container}>
        <AlbumArt
url={'https://media.istockphoto.com/id/641307550/photo/piano-keyboard-of-an-old-music-instrument-close-up.jpg?s=612x612&
w=0&k=20&c=dLbQfwsFKNZU6bZuHO6BVftTFPF2OA--h_31N3PFOMQ='} />
        <View
          style={{
            flexDirection: 'column',
            justifyContent: 'space-around',
            flex: 1,
          }}>
          <SongDetails
            artistName={'Piano'}
            songName={'Calm Down'}
          />
          <Control {...{pause, onPause, onPlay, onNext, onBack}} />
        </View>

        <Video

          source={require('./../../assets/ocean.mp3')}
          paused={pause}
          audioOnly
          poster={currentTrack.albumArtUrl}
        />
      </View>
    </>
  );
}

const styles = StyleSheet.create({
  container: {
    backgroundColor: '#111436',
    flex: 1,
    padding:20
  },
});
```

PLAYLIST

```
export const TRACKS = [
    {
      title: 'Som da floresta',
      artist: 'Natureza',
      albumArtUrl:
'https://images.unsplash.com/photo-1542273917363-3b1817f69a2d?auto=format&fit=crop&q=80&w=1000&ixlib=rb-4.0.3&ixid=
M3wxMjA3fDB8MHxzZWFyY2h8MTh8fGZvcmVzdHxlbnwwfHwwfHx8MA%3D%3D',
      audioUrl:'./../../assets/Forest.mp3',
    },
    {
```

```
    title: 'Som das ondas',
    artist: 'Natureza',
    albumArtUrl:

'https://images.unsplash.com/photo-1439405326854-014607f694d7?ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8NXx8
b2NlYW58ZW58MHx8MHx8fDA%3D&w=1000&q=80',
    audioUrl:
      './../../assets/SleepyOcean.mp3',
  },
  {
    title: 'Som dos animais',
    artist: 'Natureza',
    albumArtUrl: 'https://img.freepik.com/fotos-gratis/vista-aerea-das-arvores-verdes-vibrantes-na-floresta_181624-49828.jpg',
    audioUrl:
      './../../assets/Forest.mp3',
  },
  {
    title: 'Relax',
    artist: 'Piano',
    albumArtUrl: 'https://t4.ftcdn.net/jpg/03/20/62/91/360_F_320629162_0b0WbH58gUb1d6GSygA36FiFRg9ueoAU.jpg',
    audioUrl:
      './../../assets/ocean.mp3',
  },
  {
    title: 'Calm Down',
    artist: 'Piano',
    albumArtUrl:
'https://media.istockphoto.com/id/641307550/photo/piano-keyboard-of-an-old-music-instrument-close-up.jpg?s=612x612&w=0&
k=20&c=dLbQfwsFKNZU6bZuHO6BVftTFPF2OA--h_31N3PFOMQ=',
    audioUrl:
      'http://www.costellopsychology.com/relax/Brainwave%20Nature/Sleepy%20Ocean.mp3',
  },
];
```

Vamos entender o que fizemos:

1° **Passo:** As Importações

```
import React, {useRef, useState} from 'react';
import {StatusBar, StyleSheet, Text, View} from 'react-native';
import Video from 'react-native-video';
import { AlbumArt } from '../../components/AlbumArt';
import Control from '../../components/Control';
import SongDetails from '../../components/SngDetails';
import { TRACKS } from '../../components/Tracks';
```

2° **Passo:** O Código

```
export function Music() {
  const [pause, setPause] = useState(false);
  const [selectedTrack, setSelectedTrack] = useState(0);

  const currentTrack = TRACKS[selectedTrack];

  function onPlay() {
    setPause(false);
  }
  function onPause() {
```

```
    setPause(true);
  }
  function onNext() {
    if (selectedTrack == TRACKS.length - 1) {
      setSelectedTrack(0);
    } else {
      setSelectedTrack(selectedTrack + 1);
    }
  }

  function onBack() {
    if (selectedTrack == 0) {
      setSelectedTrack(TRACKS.length - 1);
    } else {
      setSelectedTrack(selectedTrack - 1);
    }
  }

  return (
    <>

      <View style={styles.container}>
        <AlbumArt
url={'https://media.istockphoto.com/id/641307550/photo/piano-keyboard-of-an-old-music-instrument-close-up.jpg?s=612x612&
w=0&k=20&c=dLbQfwsFKNZU6bZuHO6BVftTFPF2OA--h_31N3PFOMQ='} />
        <View
          style={{
            flexDirection: 'column',
            justifyContent: 'space-around',
            flex: 1,
          }}>
          <SongDetails
            artistName={'Piano'}
            songName={'Calm Down'}
          />
          <Control {...{pause, onPause, onPlay, onNext, onBack}} />
        </View>

        <Video

          source={require('./../../assets/ocean.mp3')}
          paused={pause}
          audioOnly
          poster={currentTrack.albumArtUrl}
        />
      </View>
    </>
  );
}
```

Estado Local (useState):

const [pause, setPause] = useState(false);: Usa o Hook useState para gerenciar o estado local pause, que controla se a reprodução da música está pausada ou não.
const [selectedTrack, setSelectedTrack] = useState(0);: Usa o Hook useState para gerenciar o estado local selectedTrack, que representa o índice da faixa de música atual.
Funções de Controle (onPlay, onPause, onNext, onBack):

onPlay(): Define pause como false para iniciar a reprodução.
onPause(): Define pause como true para pausar a reprodução.

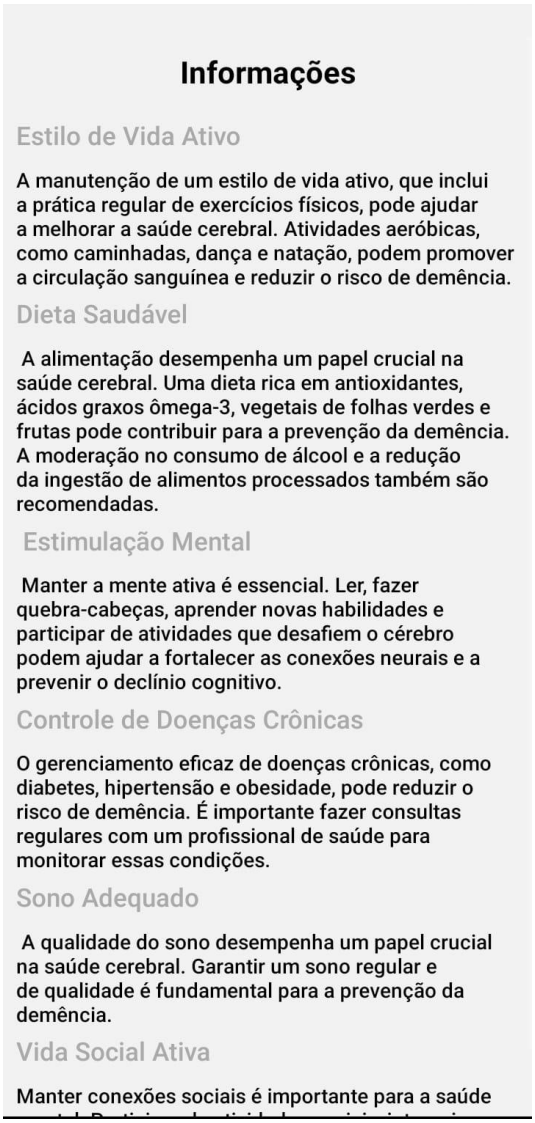
onNext(): Avança para a próxima faixa, voltando para a primeira se estiver na última.
onBack(): Retrocede para a faixa anterior, indo para a última se estiver na primeira.
Renderização do Componente Music:

AlbumArt: Renderiza um componente chamado AlbumArt e passa uma URL de imagem para a capa do álbum.
SongDetails: Renderiza um componente chamado SongDetails e passa informações sobre o artista ('Piano') e o nome da música ('Calm Down').
Control: Renderiza um componente chamado Control e passa as funções de controle e o estado pause como props.
Video: Renderiza um componente de vídeo (áudio apenas), usando a fonte do arquivo de áudio 'ocean.mp3'. A reprodução é controlada pelo estado pause.

3°Passo: Os Styles

```
const styles = StyleSheet.create({
  container: {
    backgroundColor: '#111436',
    flex: 1,
    padding:20
  },
});
```

11° Tela: Informações



Para a criação dessa tela foi utilizado o seguinte código:

```
import { useState } from "react";
import React from "react";
import {
  ScrollView,
  View,
  FlatList,
  TouchableOpacity,
  StyleSheet,
  Text,
  StatusBar,
  Image,
  TextInput
} from "react-native";
import { launchImageLibrary, launchCamera } from "react-native-image-picker";
import { black } from "react-native-paper/lib/typescript/styles/themes/v2/colors";

export function About(){
  return (
```

```
    <ScrollView style={styles.container}>
      <Text style={styles.title}>Informações</Text>

          <Text style={styles.titleParagrafo}>Estilo de Vida Ativo</Text>
        <Text style={styles.paragrafo}>A manutenção de um estilo de vida ativo, que inclui a prática regular de exercícios físicos, pode ajudar a melhorar a saúde cerebral. Atividades aeróbicas, como caminhadas, dança e natação, podem promover a circulação sanguínea e reduzir o risco de demência.
        </Text>

        <Text style={styles.titleParagrafo}>Dieta Saudável</Text>
            <Text style={styles.paragrafo}> A alimentação desempenha um papel crucial na saúde cerebral. Uma dieta rica em antioxidantes, ácidos graxos ômega-3, vegetais de folhas verdes e frutas pode contribuir para a prevenção da demência. A moderação no consumo de álcool e a redução da ingestão de alimentos processados também são recomendadas.

        </Text>

        <Text style={styles.titleParagrafo}> Estimulação Mental</Text>
            <Text style={styles.paragrafo}> Manter a mente ativa é essencial. Ler, fazer quebra-cabeças, aprender novas habilidades e participar de atividades que desafiem o cérebro podem ajudar a fortalecer as conexões neurais e a prevenir o declínio cognitivo.

        </Text>

        <Text style={styles.titleParagrafo}>Controle de Doenças Crônicas</Text>
            <Text style={styles.paragrafo}>O gerenciamento eficaz de doenças crônicas, como diabetes, hipertensão e obesidade, pode reduzir o risco de demência. É importante fazer consultas regulares com um profissional de saúde para monitorar essas condições.

        </Text>

        <Text style={styles.titleParagrafo}>Sono Adequado</Text>
            <Text style={styles.paragrafo}> A qualidade do sono desempenha um papel crucial na saúde cerebral. Garantir um sono regular e de qualidade é fundamental para a prevenção da demência.

        </Text>

        <Text style={styles.titleParagrafo}>Vida Social Ativa</Text>
            <Text style={styles.paragrafo}>Manter conexões sociais é importante para a saúde mental. Participar de atividades sociais, interagir com amigos e familiares e manter relacionamentos fortes pode ajudar a prevenir a demência.
        </Text>
    </ScrollView>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    marginTop: StatusBar.currentHeight || 0,
    padding: 10,
  columnGap:100
  },
  title: {
    fontSize:24,
    fontWeight:'700',
    justifyContent:"center",
    alignSelf:"center",
    marginBottom:20,
    color:'black'
  },
  titleParagrafo:{
    fontSize:20,
    fontWeight:'600',
    marginBottom:10,
    color:'#aaa'
  },
  paragrafo:{
```

```
    fontSize:16,
    fontWeight:'500',
    marginBottom:5,
    color:'#000'
  }
});
```

Vamos entender o que fizemos:

1° Passo: As Importações

```
import { useState } from "react";
import React from "react";
import {
  ScrollView,
  View,
  FlatList,
  TouchableOpacity,
  StyleSheet,
  Text,
  StatusBar,
  Image,
  TextInput
} from "react-native";
import { launchImageLibrary, launchCamera } from "react-native-image-picker";
import { black } from "react-native-paper/lib/typescript/styles/themes/v2/colors";
```

2° Passo: O Código

```
export function About(){
  return (
    <ScrollView style={styles.container}>
      <Text style={styles.title}>Informações</Text>

          <Text style={styles.titleParagrafo}>Estilo de Vida Ativo</Text>
        <Text style={styles.paragrafo}>A manutenção de um estilo de vida ativo, que inclui a prática regular de exercícios físicos, pode ajudar a melhorar a saúde cerebral. Atividades aeróbicas, como caminhadas, dança e natação, podem promover a circulação sanguínea e reduzir o risco de demência.
        </Text>

        <Text style={styles.titleParagrafo}>Dieta Saudável</Text>
            <Text style={styles.paragrafo}> A alimentação desempenha um papel crucial na saúde cerebral. Uma dieta rica em antioxidantes, ácidos graxos ômega-3, vegetais de folhas verdes e frutas pode contribuir para a prevenção da demência. A moderação no consumo de álcool e a redução da ingestão de alimentos processados também são recomendadas.

        </Text>

        <Text style={styles.titleParagrafo}> Estimulação Mental</Text>
            <Text style={styles.paragrafo}> Manter a mente ativa é essencial. Ler, fazer quebra-cabeças, aprender novas habilidades e participar de atividades que desafiem o cérebro podem ajudar a fortalecer as conexões neurais e a prevenir o declínio cognitivo.

        </Text>

        <Text style={styles.titleParagrafo}>Controle de Doenças Crônicas</Text>
```

```
        <Text style={styles.paragrafo}>O gerenciamento eficaz de doenças crônicas, como diabetes, hipertensão e obesidade, pode reduzir o risco de demência. É importante fazer consultas regulares com um profissional de saúde para monitorar essas condições.

      </Text>

      <Text style={styles.titleParagrafo}>Sono Adequado</Text>
        <Text style={styles.paragrafo}> A qualidade do sono desempenha um papel crucial na saúde cerebral. Garantir um sono regular e de qualidade é fundamental para a prevenção da demência.

      </Text>

      <Text style={styles.titleParagrafo}>Vida Social Ativa</Text>
        <Text style={styles.paragrafo}>Manter conexões sociais é importante para a saúde mental. Participar de atividades sociais, interagir com amigos e familiares e manter relacionamentos fortes pode ajudar a prevenir a demência.
      </Text>
    </ScrollView>
  );
};
```

<ScrollView style={styles.container}>: Cria um componente de rolagem (ScrollView) que envolve todos os elementos da tela. O estilo é definido pela folha de estilos styles.container.

<Text style={styles.title}>Informações</Text>: Exibe um título "Informações" utilizando o estilo styles.title.

Diversos Parágrafos de Texto (<Text>): Cada parágrafo de informação é exibido usando a tag <Text>. Aqui estão alguns exemplos:

Estilos (styles): A folha de estilos define estilos para os diferentes elementos do componente, incluindo títulos (title e titleParagrafo), parágrafos (paragrafo), e o contêiner geral (container). Esses estilos são usados para controlar o layout, tamanho do texto, cores e outras propriedades visuais dos elementos.

3°Passo: Os Styles

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    marginTop: StatusBar.currentHeight || 0,
    padding: 10,
  columnGap:100
  },
  title: {
    fontSize:24,
    fontWeight:'700',
    justifyContent:"center",
    alignSelf:"center",
    marginBottom:20,
    color:'black'
  },
  titleParagrafo:{
    fontSize:20,
    fontWeight:'600',
    marginBottom:10,
    color:'#aaa'
  },
  paragrafo:{
```

```
    fontSize:16,
    fontWeight:'500',
    marginBottom:5,
    color:'#000'
  }
});
```

container:

flex: 1: Faz com que o contêiner ocupe todo o espaço disponível na tela.
marginTop: StatusBar.currentHeight || 0: Garante que o conteúdo não seja ocultado pela barra de status no topo da tela.
padding: 10: Adiciona um preenchimento interno de 10 unidades.
columnGap: 100: Define um espaçamento entre as colunas de 100 unidades (embora columnGap não seja uma propriedade padrão do React Native; talvez você quis dizer marginHorizontal ou marginVertical).
title:

fontSize: 24: Define o tamanho da fonte como 24 unidades.
fontWeight: '700': Define o peso da fonte como 700 (negrito).
justifyContent: "center": Centraliza o conteúdo verticalmente.
alignSelf: "center": Centraliza o título horizontalmente em relação ao próprio elemento.
marginBottom: 20: Adiciona uma margem inferior de 20 unidades.
color: 'black': Define a cor do texto como preto.
titleParagrafo:

fontSize: 20: Define o tamanho da fonte como 20 unidades.
fontWeight: '600': Define o peso da fonte como 600.
marginBottom: 10: Adiciona uma margem inferior de 10 unidades.
color: '#aaa': Define a cor do texto como cinza claro.
paragrafo:

fontSize: 16: Define o tamanho da fonte como 16 unidades.
fontWeight: '500': Define o peso da fonte como 500.
marginBottom: 5: Adiciona uma margem inferior de 5 unidades.
color: '#000': Define a cor do texto como preto.