

# Desenvolvimento de Aplicativo Mobile

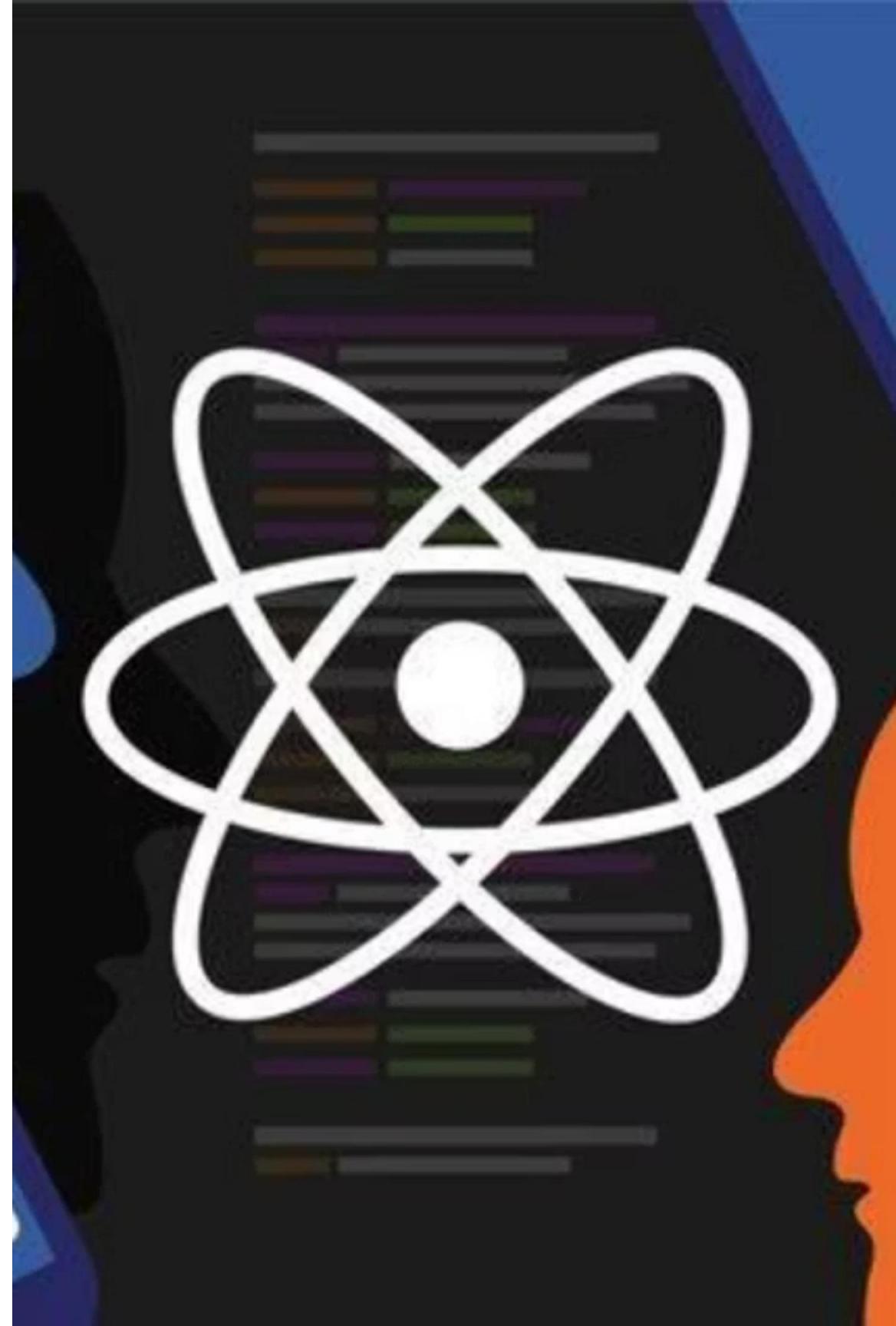
Material base desenvolvido pelo  
Prof. Raphael Barreto

---

[raphael.b.oliveira@docente.senai.br](mailto:raphael.b.oliveira@docente.senai.br)

# Estrutura de Projetos React Native com Expo

Um guia didático para desenvolvedores iniciantes entenderem como funciona a organização de um projeto React Native criado com Expo





## OBJETIVOS DA AULA

# O Que Vamos Aprender Hoje

## Estrutura de Pastas

Entender como o projeto está organizado e qual a função de cada pasta e arquivo

## Arquivos de Configuração

Conhecer os principais arquivos como package.json, app.json e suas finalidades

## Limpeza do Projeto

Aprender a remover arquivos desnecessários e preparar o ambiente para começar a desenvolver

# Relembrando: O Que Já Fizemos

01

## Criamos o projeto

Executamos o comando `create-expo-app` no terminal

02

# Rodamos o aplicativo

Iniciamos o servidor de desenvolvimento com `npx expo start`

03

## Visualizamos no celular

Usamos o Expo Go para ver o app funcionando (ou o simulador, se preferir)



# A Grande Pergunta Agora

## E agora? Como faço meu próprio app?

Você criou o projeto, rodou ele, viu funcionando... mas e aí? O que já veio pronto? Onde você começa a mexer? Essas são as perguntas que vamos responder agora, explorando cada parte do projeto que o Expo criou para você.

# O Que Aparece ao Abrir o App

Quando você roda o projeto pela primeira vez, ele mostra uma tela explicativa com várias informações úteis:

- Indica qual arquivo você deve editar  
(app/tabs/index.tsx)
  - Dá dicas sobre roteamento e navegação
  - Explica como adicionar imagens e fontes
  - Mostra recursos prontos como Dark Mode
  - Apresenta exemplos de animações

É como um guia de boas-vindas do Expo!

EXPLORER

... index.tsx X

PROJETOS

listade-tarefas > app > (tabs) > index.tsx > HomeScreen

```
1 import { Image } from 'expo-image';
2 import { Platform, StyleSheet } from 'react-native';
3
4 import { HelloWave } from '@/components/hello-wave';
5 import ParallaxScrollView from '@/components/parallax-scroll-view';
6 import { ThemedText } from '@/components/themed-text';
7 import { ThemedView } from '@/components/themed-view';
8 import { Link } from 'expo-router';
9
10 export default function HomeScreen() {
11   return (
12     <ParallaxScrollView
13       headerBackgroundColor={{ light: '#A1CEDC', dark: '#1D3D47' }}
14       headerImage={
15         <Image
16           source={require('@/assets/images/partial-react-logo.png')}
17           style={styles.reactLogo}
18         />
19       }>
20       <ThemedView style={styles.titleContainer}>
21         <ThemedText type="title">Welcome!</ThemedText>
22         <HelloWave />
23       </ThemedView>
24       <ThemedView style={styles.stepContainer}>
25         <ThemedText type="subtitle">Step 1: Try it</ThemedText>
26         <ThemedText>
27           Edit <ThemedText type="defaultSemiBold">app/(tabs)/index.tsx</ThemedText> to see
28           changes.
29           Press{' '}
30           <ThemedText type="defaultSemiBold">
31             {Platform.select({
32               ios: 'cmd + d',
33               android: 'cmd + m',
34               web: 'F12',
35             })}
36           </ThemedText>[' ']
37           to open developer tools.
38       </ThemedText>
39     </ParallaxScrollView>
40   )
41 }
```

# Visão Geral da Estrutura do Projeto

Antes de mergulharmos nos detalhes, vamos ter uma visão panorâmica de como o projeto está organizado. Algumas pastas você já conhece de outros projetos JavaScript, outras são específicas do React Native e do Expo.

## CONFIGURAÇÕES

# Pasta .expo (Pode Ignorar)

 **Importante:** Você não precisa mexer nessa pasta! Ela contém configurações internas do Expo.

A pasta `.expo` é criada automaticamente pelo framework Expo e contém:

- Configurações de build do projeto
- Cache de desenvolvimento
- Arquivos temporários

O ponto (.) no início do nome indica que é um arquivo/pasta oculto no sistema. No VS Code, você pode até configurar para não exibir essas pastas, mantendo seu explorador de arquivos mais limpo.

```
  "name": "lista-de-tarefas",
  "main": "expo-router/entry",
  "version": "1.0.0",
  ▷Debug
  "scripts": {
    "start": "expo start",
    "reset-project": "node ./scripts/reset-project.js",
    "android": "expo start --android",
    "ios": "expo start --ios",
    "web": "expo start --web",
    "lint": "expo lint"
  },
  "dependencies": {
    "@expo/vector-icons": "^15.0.3",
    "@react-navigation/bottom-tabs": "^7.4.0",
    "@react-navigation/elements": "^2.6.3",
    "@react-navigation/native": "^7.1.8",
    "expo": "~54.0.32",
    "expo-constants": "~18.0.13",
    "expo-font": "~14.0.11",
    "expo-haptics": "~15.0.8",
    "expo-image": "~3.0.11",
    "expo-linking": "~8.0.11",
    "expo-router": "~6.0.22",
    "expo-splash-screen": "~31.0.13",
    "expo-status-bar": "~3.0.9",
    "expo-symbols": "~1.0.8",
    "expo-system-ui": "~6.0.9",
    "expo-web-browser": "~15.0.10",
    "react": "19.1.0",
    "react-dom": "19.1.0",
    "react-native": "0.81.5",
    "react-native-gesture-handler": "~2.28.0",
    "react-native-worklets": "0.5.1",
    "react-native-reanimated": "~4.1.1",
    "react-native-safe-area-context": "~5.6.0",
  }
}
```

# package.json: O Coração do Projeto

Se você já trabalhou com React ou Node.js, esse arquivo é um velho conhecido!

O package.json gerencia todas as dependências e scripts do projeto.

## Scripts Disponíveis

Comandos para iniciar o app em diferentes plataformas: `start`, `android`, `ios`, `web`

## Dependências

Lista de todas as bibliotecas que o projeto usa, instaladas automaticamente pelo Expo

## DevDependencies

Ferramentas de desenvolvimento como TypeScript, Jest e Babel

# Dependências Principais do Projeto

## As Três Bibliotecas Essenciais

```
"react": "19.1.0",
"react-dom": "19.1.0",
"react-native": "0.81.5",
```

Essas são as únicas bibliotecas realmente necessárias para um projeto React Native funcionar. Todo o resto é opcional, mas o Expo já traz várias ferramentas úteis pré-instaladas.

## Bibliotecas Extras do Expo

- `@expo/vector-icons`: Ícones prontos
- `expo-router`: Sistema de navegação
- `expo-font`: Fontes customizadas
- `react-native-gesture-handler`: Gestos no celular
- `react-native-reanimated`: Animações

# Entendendo as Bibliotecas do Expo



## Gesture Handler

Permite trabalhar com gestos complexos no celular: pinça para zoom, arrastar elementos, deslizar para os lados. Tudo isso já vem pronto!



## Reanimated

Biblioteca poderosa para criar animações suaves e performáticas. Desde simples transições até animações complexas.



## Router

Sistema de navegação entre telas baseado em arquivos, similar ao Next.js. Muito mais simples que o React Navigation tradicional.

## Splash Screen

Gerencia aquela tela que aparece quando você abre o app pela primeira vez, enquanto ele está carregando.

TYPESCRIPT

## Arquivos de Configuração TypeScript

O Expo cria o projeto já preparado para TypeScript por padrão. Você vai encontrar:

- **tsconfig.json**: Configurações do compilador TypeScript
- **expo-env.d.ts**: Definições de tipos específicas do Expo (não edite este arquivo!)

 **Vai usar JavaScript?** Você pode deletar esses arquivos e renomear as extensões .tsx para .jsx

# app.json: Configurações do Aplicativo



## Ícones

Define os ícones do app para diferentes plataformas (iOS, Android, Web)



## Cores do Sistema

Cor da barra de status, splash screen e tema geral do aplicativo



## Splash Screen

Imagen e configurações da tela de carregamento inicial



## Outras Configs

Nome do app, versão, orientação da tela e muito mais

Todas essas configurações afetam como o aplicativo vai aparecer no celular do usuário. Vamos personalizá-las mais para frente!

# Arquivos Auxiliares

---

## .gitignore

Lista de arquivos que não devem ser enviados para o repositório Git (como node\_modules e arquivos de build)

## package-lock.json

Trava as versões exatas de cada dependência instalada, garantindo que o projeto funcione igual em qualquer máquina

## README.md

Documentação básica do Expo explicando como instalar dependências e rodar o projeto

# node\_modules: A Pasta Pesada

Essa pasta contém **todos os arquivos** das bibliotecas que seu projeto usa. É aqui que ficam o React, React Native, Expo e todas as outras dependências.

Características importantes:

- Gerada automaticamente ao rodar `npm install`
- Pode ter centenas de megabytes de tamanho
- Nunca deve ser enviada para o Git (já está no `.gitignore`)
- Pode ser deletada e recriada a qualquer momento

Se algo estranho acontecer, tente deletar `node_modules` e rodar `npm install` novamente!

```
import { > splash
import {
import { StatusBar } from 'expo-status-bar';
import 'react-native-reanimated';

import { useColorScheme } from '@hooks/use-color-scheme';

export const unstable_settings = {
  anchor: '(tabs)',
};

export default function RootLayout() {
  const colorScheme = useColorScheme();

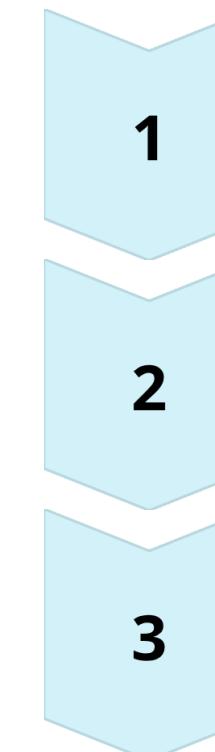
  return (
    <ThemeProvider value={colorScheme === 'dark' ? 'dark' : 'light'}>
      <Stack>
        <Stack.Screen name="(tabs)" options={{>
          <Stack.Screen name="modal" options={{>
            </Stack>
            <StatusBar style="auto" />
          </ThemeProvider>
        )};

```

#### PASTAS DO CÓDIGO

## A Pasta /app: Onde a Mágica Acontece

Esta é a pasta mais importante do projeto! É aqui que você vai criar suas telas e toda a lógica do aplicativo. O Expo Router usa essa pasta para entender a navegação do app automaticamente.



### \_layout.jsx

Arquivo especial que define o layout base do app

### /tabs

Pasta com as telas de navegação por abas

### Suas telas

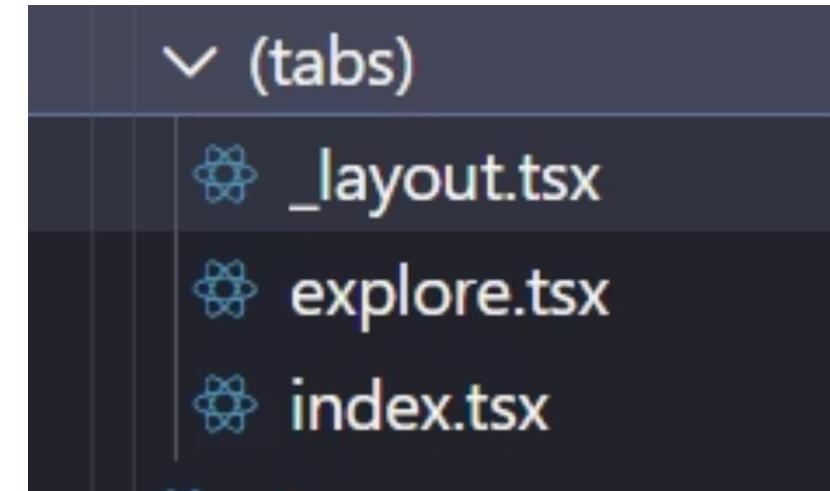
Você vai criar novos arquivos aqui para cada tela do app

# Estrutura Inicial das Telas

## Pasta /tabs

O projeto inicial vem com uma estrutura de navegação por abas (tabs), muito comum em apps móveis. Você encontra:

- `index.tsx`: A tela inicial (Home)
- `explore.tsx`: Segunda tela de exemplo
- `_layout.tsx`: Configuração das tabs



Cada arquivo `.tsx` representa uma tela diferente do aplicativo.

# Outras Pastas Importantes



## /hooks

Custom hooks reutilizáveis. O Expo já inclui hooks para temas, cores e muito mais



## /constants

Valores constantes usados no app, como a paleta de cores padrão



## /components

Componentes reutilizáveis em várias partes do app, como botões e cards customizados



## /assets

Imagens, fontes e outros arquivos estáticos do projeto

## A Pasta /assets em Detalhes

### Imagens

Logo do React Native, ícones adaptativos para iOS e Android, splash screen

### Fontes

O Expo já vem com algumas fontes customizadas prontas para usar (`SpaceMono-Regular.ttf`)

É nesta pasta que você vai colocar suas próprias imagens e fontes quando começar a personalizar o app!

HORA DE LIMPAR!

## Por Que Limpar o Projeto?

# Vamos começar do zero!

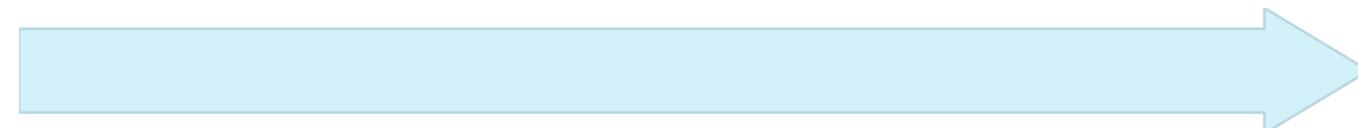
O projeto inicial vem com muito código de exemplo que você provavelmente não vai usar. Para aprender melhor, é mais interessante começar com um projeto limpo e ir adicionando as coisas aos poucos.

Duas opções para limpar:

1. Usar o script de reset que o Expo fornece
2. Limpar manualmente (mais controle!)

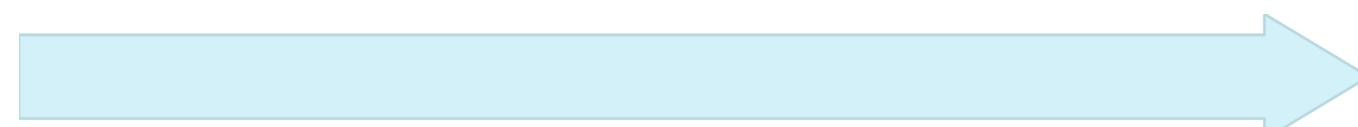


# Limpeza Manual: Passo a Passo



## Remover TypeScript (se for usar JS)

Delete `tsconfig.json` e `expo-env.d.ts`



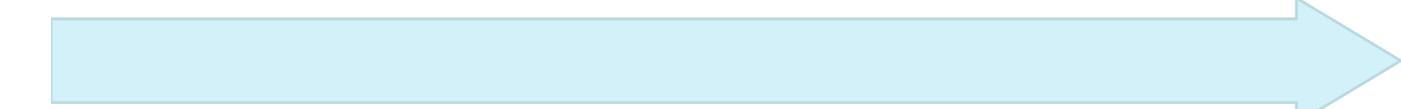
## Limpar assets

Delete fontes e imagens de exemplo, mas mantenha a pasta  
`/assets` vazia



## Limpar pastas auxiliares

Delete as pastas `/scripts`, `/hooks`, `/constants` e `/components`



## Simplificar /app

Delete `/tabs`

em `app/` mantenha apenas o `_layout.tsx` e `index.jsx` e renomeie a extensão para `.jsx`

# Convertendo para JavaScript (\_layout.jsx e index.jsx)

## Antes (TypeScript)

```
// _layout.tsx

import { Stack } from 'expo-router';

export default function RootLayout() {
  return <Stack />;
}
```

## Depois (JavaScript)

```
// _layout.jsx

import { Text } from 'react-native';

export default function RootLayout() {
  return <Text>Olá Mundo!</Text>;
}
```

Note que mudamos a extensão de `.tsx` para `.jsx` e simplificamos o conteúdo!

# Primeiro Componente Funcionando

Depois de limpar tudo e simplificar o `_layout.jsx`, seu app vai mostrar apenas um texto simples: "Olá Mundo!"

Pode parecer que "quebramos" tudo, mas na verdade agora você tem total controle sobre cada linha de código do projeto!

 **Dica:** Salve o arquivo e veja a atualização automática no celular/simulador. É o Fast Refresh em ação!

# Testando o Hot Reload

## Faça uma mudança no código

Altere o texto de "Olá Mundo!" para qualquer outra coisa

## Salve o arquivo (Ctrl+S)

O app deve atualizar automaticamente em poucos segundos

## Se não atualizar

No simulador: pressione R para reload. No celular: balance o dispositivo e escolha "Reload"

## Estrutura Mínima Necessária

# O que você realmente precisa?



Com o arquivo `_layout.jsx`



Configurações do app



Dependências e scripts



Para suas imagens (pode estar vazia por enquanto)

Com apenas isso, você já tem um projeto React Native funcional e pronto para começar a desenvolver!

# O Arquivo `_layout`: Por Que Esse Nome?

O Expo Router (sistema de navegação baseado em arquivos) usa convenções específicas de nomenclatura:

- Arquivos com `_` na frente são especiais
- `_layout.jsx` define o layout base das rotas
- Similar ao `App.jsx` do React tradicional
- É o ponto de entrada do aplicativo



Vamos entender melhor essa convenção quando falarmos sobre navegação em aulas futuras!

# Comparação: React vs React Native

## React Web

```
// App.jsx

function App() {
  return (
    <div>
      <h1>Olá Mundo</h1>
    </div>
  );
}
```

Usa HTML padrão: div, h1, p, span, etc.

## React Native

```
// _layout.jsx

function RootLayout() {
  return (
    <Text>Olá Mundo</Text>
  );
}
```

Usa componentes específicos: Text, View, Image, etc.

Na próxima aula vamos explorar esses componentes nativos em detalhes!

 RESUMO

# O Que Aprendemos Hoje

## Estrutura de Pastas

Entendemos cada pasta e arquivo do projeto Expo, desde configurações até código

## Dependências

Conhecemos as bibliotecas essenciais e as ferramentas extras que o Expo fornece

## Limpeza do Projeto

Aprendemos a remover código de exemplo e deixar apenas o necessário

## Primeiro Componente

Criamos nosso primeiro componente funcional e vimos o hot reload em ação

## Desafio Prático

# Mãos à Obra!



### Crie seu projeto

Execute `npx create-expo-app meu-app` no terminal



### Limpe o código

Siga o passo a passo desta aula para deixar o projeto limpinho



### Teste no celular

Rode o app no Expo Go e experimente fazer mudanças no código

Tire um momento para explorar cada pasta e arquivo. Quanto mais familiarizado você ficar com a estrutura agora, mais fácil será desenvolver depois!



# Perguntas?

Entre em contato:

[raphael.b.oliveira@docente.senai.br](mailto:raphael.b.oliveira@docente.senai.br)

Obrigado pela atenção! Estou à disposição para esclarecer dúvidas e ajudar no seu aprendizado.

**Firjan SENAI**

