O que é Expo

O Expo é uma plataforma de desenvolvimento que facilita a criação de aplicativos móveis usando JavaScript e React Native. Ele fornece uma série de ferramentas e serviços que tornam o processo de desenvolvimento mais rápido e eficiente, especialmente para desenvolvedores que desejam criar aplicativos para iOS e Android a partir de um único código base.

Para instalar o Expo CLI através do Node.js, você pode usar o gerenciador de pacotes npm (Node Package Manager) no cmd . Aqui está o comando que você deve executar no terminal:

```
npm install -g expo-cli
```

Esse comando instala o Expo CLI globalmente no seu sistema, o que significa que você poderá usá-lo em qualquer lugar do seu terminal.

Após a instalação, você pode verificar se o Expo CLI foi instalado corretamente executando o comando:

```
expo --version
```

Agora, é extremamente importante que o CMD (prompt de comando/terminal de comandos) esteja trabalhando dentro da pasta onde você quer guardar seu projeto. Caso contrário, você corre o risco de perder seu projeto. Se você tem dúvidas de como isso acontece, acesse o PDF do CMD da aula passada.

Criando projeto Expo

Com a pasta selecionada execute o comando: (Lembre-se: a palavra nomedoseuapp é literalmente para você colocar o nome do seu app.)

```
expo init nomedoseuapp
```

Chamando a ferramenta do Expo que você acabou de instalar.

Pedindo para ela iniciar um projeto.

nome do projeto que ela deve criar.

"Expo iniciar um projeto chamado nomedoapp"

Quando você executa o comando expo init nomedoapp, o Expo CLI apresenta opções de templates para escolher como base para o seu projeto. Normalmente, você verá quatro opções principais, que são:

```
? Choose a template: » - Use arrow-keys. Return to submit.
    ----- Managed workflow -----
>   blank                a minimal app as clean as an empty canvas
    blank (TypeScript)   same as blank but with TypeScript configuration
    tabs (TypeScript)    several example screens and tabs using react-navigation and TypeScript
    ----- Bare workflow -----
    minimal              bare and minimal, just the essentials to get you started
```

Managed Workflow: Simplifica o desenvolvimento e configuração, ideal para projetos que podem ser completamente geridos pelo Expo.

Bare Workflow: Oferece maior flexibilidade e controle sobre o código nativo, ideal para projetos que precisam de customizações avançadas ou integrações específicas.

Como precisamos de algumas pastas e estruturas prontas, vamos escolher a segunda opção, blank (TypeScript). Lembre-se de que no terminal o mouse não funciona então você deve utilizar as teclas de seta do teclado e apertar Enter.

```
>   blank (TypeScript)   same as blank but with TypeScript configuration
```

Criando projeto Expo

Caso o seguinte erro aconteça, você deve executar os seguintes comandos para desinstalar o Expo e instalá-lo novamente.

$ expo init is not supported in the local cli please use npx create-expo-app instead

Desinstalar o Expo CLI
Use o seguinte comando para desinstalar o Expo CLI globalmente:.

```
npm uninstall -g expo-cli
```

O npm pode manter arquivos em cache, então é uma boa prática limpá-lo para garantir que tudo relacionado ao Expo CLI seja removido:

```
npm cache clean --force
```

Agora, reinstale o Expo normalmente, o que instalará a versão mais nova, e continue o passo a passo.

Para instalar o Expo CLI através do Node.js, você pode usar o gerenciador de pacotes npm (Node Package Manager) no cmd . Aqui está o comando que você deve executar no terminal:

```
npm install -g expo-cli
```

Esse comando instala o Expo CLI globalmente no seu sistema, o que significa que você poderá usá-lo em qualquer lugar do seu terminal.

Após a instalação, você pode verificar se o Expo CLI foi instalado corretamente executando o comando:

```
expo --version
```

Manipulando o projeto Expo

”Esse processo pode demorar um pouco, pois ele está usando o nome para baixar as dependências que o projeto precisa. (Se você não lembra nada sobre o Node, volte ao PDF do Node.)”

Quando finalizar, irá aparecer as seguintes mensagens no seu terminal de comando:

```
Migrate to using:
› npx create-expo-app --template

√ Choose a template: » blank (TypeScript)  same as blank but with TypeScript configuration
√ Downloaded template.
 Using npm to install packages.
√ Installed JavaScript dependencies.

 Your project is ready!

To run your project, navigate to the directory and run one of the following npm commands.

- cd nomedoseuapp
- npm start # you can open iOS, Android, or web from here, or run them directly with the commands below.
- npm run android
- npm run ios # requires an iOS device or macOS for access to an iOS simulator
- npm run web
```

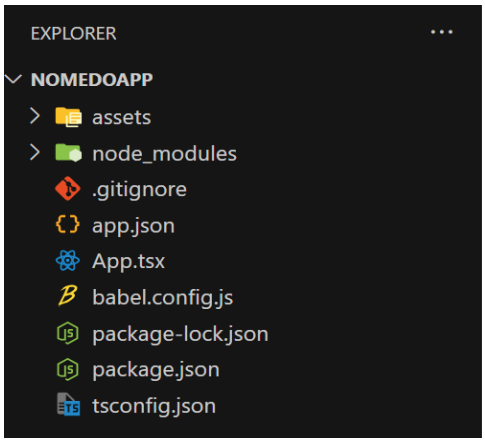
Pronto, agora que o projeto já foi criado, dentro da pasta selecionada você deve verificar se está realmente na pasta do projeto. Ela deve conter os seguintes arquivos. (Caso você não lembre como conferir os arquivos de uma pasta, volte no PDF do CMD.)

```
25/08/2024  18:52    <DIR>          .
25/08/2024  18:52    <DIR>          ..
26/10/1985  05:15               375 .gitignore
25/08/2024  18:51               584 app.json
26/10/1985  05:15               455 App.tsx
25/08/2024  18:51    <DIR>          assets
26/10/1985  05:15               107 babel.config.js
25/08/2024  18:52    <DIR>          node_modules
25/08/2024  18:52           538.278 package-lock.json
25/08/2024  18:51               499 package.json
26/10/1985  05:15                85 tsconfig.json
               7 arquivo(s)        540.383 bytes
               4 pasta(s)    8.547.442.688 bytes disponíveis
```

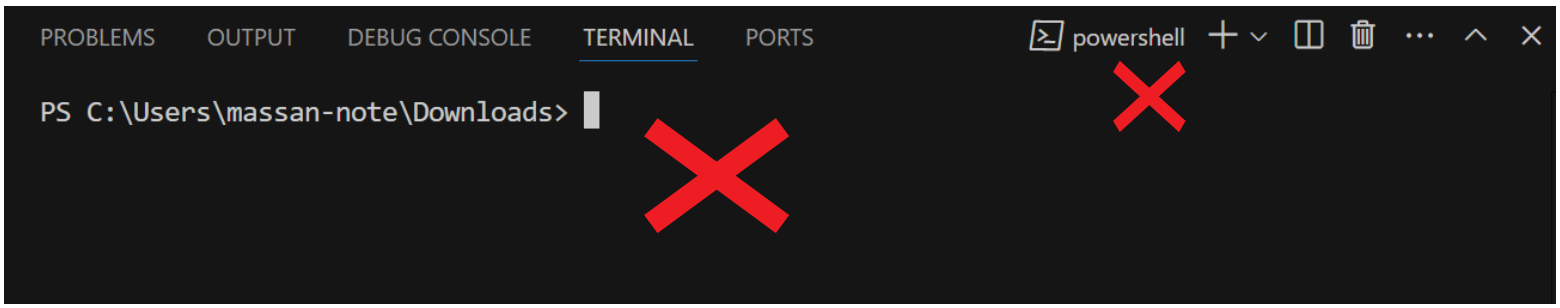
Agora que você conferiu que o projeto realmente está na pasta correta, você deve abrir ele no VS Code . (Se você não lembra como abrir uma pasta que está selecionada no CMD no VS Code, volte ao PDF do CMD.)

Manipulando o projeto Expo

Agora, com o VS Code aberto na pasta correta, temos o controle de todas as pastas do nosso aplicativo, e cada uma delas é essencial para o funcionamento. Mas lembre-se de verificar se você está na pasta correta. Veja como deve ficar a estrutura se estiver tudo certo.



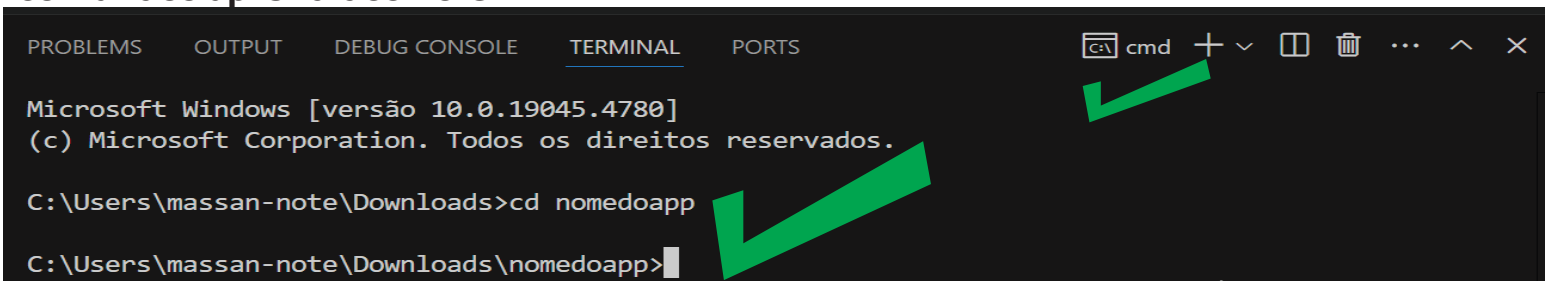
Você deve se acostumar a abrir o VS Code na pasta correta, ou sempre terá algum problema relacionado a isso. Um dos problemas pode acontecer quando você abrir o terminal de comando no VS Code. Agora, abra o terminal de comando no VS Code pressionando CTRL+J.



Repare que o caminho está incorreto e o tipo de terminal também. O terminal deve ser o CMD (Prompt de Comando) e não o PowerShell, e o caminho deve ser o nome da pasta do seu projeto.

Para trocar de terminal, clique na setinha ao lado do nome do terminal incorreto e selecione o 'Command Prompt (CMD)'. Se não quiser ficar com dois terminais abertos, clique na lixeira no terminal que você não quer utilizar.

Para corrigir o caminho da pasta do seu projeto, você deve fazer a troca utilizando os comandos aprendidos no CMD



Manipulando o projeto Expo

O App.tsx é essencialmente o coração do seu aplicativo React Native, servindo como o ponto de entrada onde você configura a estrutura básica, navegação, e outros aspectos globais da aplicação. Ele controla o que os usuários verão primeiro ao abrir o aplicativo e gerencia a lógica principal por trás da UI.

Agora, podemos dar início ao nosso aplicativo em React Expo. Execute no terminal de comando do Vscode o comando npx expo start.

O primeiro componente que devemos entender é o <View>. Ela funciona como uma caixa, e dentro dela podemos definir como os elementos serão organizados. Também podemos definir o tamanho dessa caixa e o espaço que ela ocupa na tela. Além disso, podemos ter uma view dentro de outra view.

```
<View style={styles.container}>
  <Text>Open up App.tsx to start working on your app!</Text>
  <StatusBar style="auto" />
</View>
```

Mais acima, temos a importação dos componentes. Repare que o componente <View> está lá também. Isso acontece porque, sempre que vamos utilizar um componente, devemos primeiro importá-lo.

```
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';
```

Exercício: Explorando o Componente View

Vamos criar uma interface simples que utilize o componente View para organizar elementos na tela, experimentando diferentes propriedades e estilos.

Passo 1: Configuração Inicial

1 - Crie um novo projeto Expo ou abra um projeto existente.

2 - No arquivo App.tsx, importe os componentes necessários:.

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

Passo 2: Criar a Estrutura Básica

Dentro do componente App, utilize o componente View para organizar dois blocos de texto (Text).

Cada bloco de texto deve estar dentro de uma View separada.

```
export default function App() {
  return (
    <View style={styles.container}>
      <View style={styles.box1}>
        <Text>Box 1</Text>
      </View>
      <View style={styles.box2}>
        <Text>Box 2</Text>
      </View>
    </View>
  );
}
```

Passo 3: Estilização

Defina um estilo básico para o View principal (container) e para as View internas (box1 e box2).

Use propriedades como flex, padding, margin, backgroundColor, e alignItems para modificar o layout e visualização das View.

Exercício: Explorando o Componente View

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#f0f0f0',
  },
  box1: {
    width: 100,
    height: 100,
    backgroundColor: 'lightblue',
    justifyContent: 'center',
    alignItems: 'center',
    marginBottom: 20,
  },
  box2: {
    width: 100,
    height: 100,
    backgroundColor: 'lightcoral',
    justifyContent: 'center',
    alignItems: 'center',
  },
});
```

Passo 4: Teste Tambem

Flexbox: Modifique as propriedades justifyContent e alignItems do container para ver como os View internos são posicionados.

Dimensões: Alterne entre valores de largura (width) e altura (height) fixos e proporcionais (flex) para os View internos.

Estilização Avançada: Adicione propriedades como borderRadius, shadowColor, shadowOffset, e elevation para explorar como estilos podem modificar a aparência dos componentes View.