Trabalhando com lista no React Native.

Crie um App no https://snack.expo.dev/ e dê o nome de “trabalhando-listas”. Monte a estrutura abaixo:

Primeiro exemplo:

```
import { View, Text, SafeAreaView, StyleSheet } from 'react-native';
function App() {
    return (
      <SafeAreaView style={styles.container}>
      </SafeAreaView>
    );
}
const styles = StyleSheet.create({
    container: {
    },
    texto: {
      fontSize: 30
    }
});
export default App;
```

Uma vez criado o App, vamos criar uma lista para exibir no sistema:

```
let list = [
  {id:1, nome: 'João', idade: 20},
  {id:2, nome: 'Pedro', idade: 30},
  {id:3, nome: 'Maria', idade: 40},
  {id:4, nome: 'Josefina', idade: 50},
];
```

A lista montada é criada dentro do App():

```
function App() {
  let list = [
    {id:1, nome: 'João', idade: 20},
    {id:2, nome: 'Pedro', idade: 30},
    {id:3, nome: 'Maria', idade: 40},
    {id:4, nome: 'Josefina', idade: 50},
  ];
  return (
```

Agora vamos fazer uso da lista:

Pegando o tamanho -> */*<Text>{list.length} Pessoas</Text>*/*

E trabalhando com a lista:

```
    <SafeAreaView style={styles.container}>
      /*<Text>{list.length} Pessoas</Text>*/
      {list.map((user) => (
        <View key={user.id}>
          <Text style={styles.texto}>{user.nome}</Text>
        </View>
      ))}
    </SafeAreaView>
  );
```

Agora que temos todas as partes do sistema, vamos finalizá-lo:

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



function App() {
  let list = [
    {id:1, nome: 'João', idade: 20},
    {id:2, nome: 'Pedro', idade: 30},
    {id:3, nome: 'Maria', idade: 40},
    {id:4, nome: 'Josefina', idade: 50},
  ];
  return (
    <SafeAreaView style={styles.container}>
            /*<Text>{list.length} Pessoas</Text>*/

     {list.map((user) => (
          <View key={user.id}>
            <Text style={styles.texto}>{user.nome}</Text>
          </View>
        ))}
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({
  container: {
  },
  texto: {
    fontSize: 30
  }
});

export default App;
```

Crie um Segundo App, Chamado: Trabalhando-FlatList

Vá até a pasta componentes e crie um novo arquivo chamado: user-item.tsx

Em seguida vamos criar o componente:

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


type Props = {
  nome: string;
  idade: number;
  id: number;
}

export const UserItem = (props: Props) => {
  return (
    <View style={styles.container}>
        <Text style={styles.big}>{props.nome}</Text>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
   margin: 10
  },
  big: {
    fontSize: 28
  }
});
```

Uma vez criado o componente, vamos para o corpo principal do sistema, App.tsx

```
import { View, Text, SafeAreaView, StyleSheet, FlatList } from 'react-native';
import { UserItem } from './components/user-item'

type User = {
  id: number;
  nome: string;
  idade: number;
}

function App() {
  let list = [
    {id:1, nome: 'João', idade: 20},
    {id:2, nome: 'Pedro', idade: 30},
    {id:3, nome: 'Maria', idade: 40},
    {id:4, nome: 'Josefina', idade: 50},
  ];
  return (
    <SafeAreaView style={styles.container}>
      /*  {list.map((user) => (
            <UserItem key={user.id} nome={user.nome} idade={user.idade} id={user.id}
/>
        ))}*/

        <FlatList
            data={list}
            renderItem={({ item } : { item : User}) => (<UserItem  nome={item.nome}
idade={item.idade} id={item.id} />)}
            keyExtractor={item => item.id.toString()}
         />
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({
  container: {
  },
});

export default App;
```

Explicação do App

Explicação Geral

Esse código implementa um aplicativo simples em **React Native** que exibe uma lista de usuários usando o **FlatList**, um componente otimizado para renderizar grandes listas de forma eficiente.

Ele é composto por dois arquivos principais:

1. **user-item.tsx** – Define um componente para exibir um item de usuário.
2. **App.tsx** – Define a estrutura principal da aplicação e exibe a lista de usuários.

Explicação do Arquivo user-item.tsx

Este arquivo contém um componente chamado **UserItem**, que representa um item da lista de usuários.

Código

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

⮕ Importação de módulos

- View: Componente que serve como um contêiner para organizar outros elementos na tela.
- Text: Componente usado para exibir textos na tela.
- StyleSheet: API usada para criar estilos para os componentes, semelhante ao CSS.

```
type Props = {
  nome: string;
  idade: number;
  id: number;
}
```

⮕ Definição do tipo das propriedades (Props)

- Define os **parâmetros que o componente UserItem pode receber**:
    - nome: Nome do usuário (**string**).
    - idade: Idade do usuário (**number**).
    - id: Identificação do usuário (**number**).

```
export const UserItem = (props: Props) => {
```

⮕ Definição do Componente UserItem

- Recebe as propriedades (**props**) do tipo Props.

```
 return (
  <View style={styles.container}>
    <Text style={styles.big}>{props.nome}</Text>
  </View>
 );
```

➞ Estrutura do Componente

- View: Serve como contêiner para os elementos.
- Text: Exibe o nome do usuário (props.nome).
- **Uso de estilos (styles.big)** para definir o tamanho do texto.

```
const styles = StyleSheet.create({
 container: {
  margin: 10
 },
 big: {
   fontSize: 28
 }
});
```

➞ Estilos do Componente

- container: Define um espaçamento de **10 pixels** ao redor do item.
- big: Define que o tamanho da fonte do nome será **28 pixels**.

Explicação do Arquivo App.tsx

Este é o arquivo principal da aplicação, onde a lista de usuários é exibida.

Código

```
import { View, Text, SafeAreaView, StyleSheet, FlatList } from 'react-native';
import { UserItem } from './components/user-item'
```

➞ Importação de módulos

- SafeAreaView: Garante que o conteúdo não ultrapasse áreas seguras do sistema operacional (como notch e barras de status).
- FlatList: Componente eficiente para exibir listas longas.
- UserItem: Importa o **componente UserItem** criado anteriormente.

```
type User = {
  id: number;
  nome: string;
  idade: number;
}
```

➞ Definição do tipo User

- Define a estrutura de um objeto **usuário**, com as propriedades id, nome e idade.

```
function App() {
  let list = [
    {id:1, nome: 'João', idade: 20},
    {id:2, nome: 'Pedro', idade: 30},
    {id:3, nome: 'Maria', idade: 40},
    {id:4, nome: 'Josefina', idade: 50},
  ];
```

➞ Lista de Usuários (list)

- Um **array de objetos** contendo os dados dos usuários.

```
  return (
    <SafeAreaView style={styles.container}>
      <FlatList
        data={list}
        renderItem={({ item } : { item : User}) => (<UserItem  nome={item.nome} idade={item.idade} id={item.id} />)}
        keyExtractor={item => item.id.toString()}
      />
    </SafeAreaView>
  );
```

➞ Renderização da Lista (FlatList)

- **data={list}** → Define a lista de usuários que será renderizada.
- **renderItem={({ item }) => (<UserItem nome={item.nome} idade={item.idade} id={item.id} />)}**

- Para cada item da lista, ele cria um componente UserItem, passando nome, idade e id como propriedades.

- **keyExtractor={item => item.id.toString()}**
  - Define um identificador único para cada item da lista.

```
const styles = StyleSheet.create({
  container: {
  },
});
```

➞ Estilos

- O objeto styles poderia ter mais configurações, mas está vazio neste exemplo.

Resumo Geral

- O App.tsx exibe uma **lista de usuários**.
- O UserItem exibe **cada usuário individualmente**.
- O FlatList cuida da **renderização eficiente** da lista.