# Vamos criar um App

**Walber do Carmo Farias, 202303752547**

**Polo de Abaetetuba (PA)**
**Nível 1 – Vamos criar um App– 2023.1 – 2024.1**

## Objetivo da Prática

- Configurar o ambiente de desenvolvimento React Native;
- Implementar a funcionalidade de entrada de texto em um componente React Native;
- Implementar um Componente de Lista Dinâmica (ScrollView);
- Implementar componentes React Native para exibir informações de forma dinâmica em listas;
- Empregar elementos visuais em um aplicativo React Native.

## 1º Missão Prática

- Criando a função **adicionarFornecedor,** usando um recurso chamado Hooks (o useState é o hook), que ajuda a lidar com dados dentro de um componente

    **const [fornecedores, setFornecedores] = useState([ ]);**

    A primeira parte cria uma variável chamada fornecedores, que vai guardar um vetor de fornecedores

    setFornecedores é a função que muda o valor dessa lista quando necessário.

    O useState([]) define que o vetor começa vazio (com um array vazio []).

    **const [nome, setNome] = useState(' ');**

    Temos uma variável chamada nome, que guarda o nome de um fornecedor.

    setNome serve para mudar o valor do nome.

    O useState(' ') diz que o nome começa como uma string vazia (' ')

    E assim por diante com os demais campos…

```
const adicionarFornecedor = () => {
    if (nome && cnpj && telefone && imagem) {
        setFornecedores(prevFornecedores => [
            ...prevFornecedores,
            {
                id: Math.random().toString(),
                nome,
                cnpj,
                telefone,
                imagem,
            },
        ]);

        setNome('');
        setCnpj('');
        setTelefone('');
        setImagem(null);
    } else {
        Alert.alert('Erro', 'Preencha todos os campos, incluindo a imagem!');
    }
};
```

- O código cria uma função chamada **solicitarPermissao**, no qual será responsável por solicitar permissão do usuário para acessar a galeria de fotos, afinal, o usuário precisa selecionar uma foto para o cadastro do fornecedor.

  O termo **async** significa "assíncrono", ou seja, estamos definindo que a função **solicitarPermissao** será assíncrona, pois pode demorar um tempo para o usuário aceitar ou recusar a permissão de acesso a galeria de fotos.

  Logo após temos uma constante chamada status, que recebe o retorno da função requestMediaLibraryPermissionsAsync().

  Caso o usuário aceite a solicitação de acesso, a função retornará valor 'granted'.

  Temos nessa mesma linha o uso da palavra await (que significa aguardar), ou seja, como é uma função assíncrona (que pode demorar para executar), usamos a palavra await para dizer que esta demora deve ser aguardada para que depois o aplicativo continue sua execução.

  Caso a permissão seja negada (status diferente de 'granted'), o aplicativo mostrará um alerta dizendo 'permissão necessária para acessar suas fotos.'

```
32  const solicitarPermissao = async () => {
33      const { status } = await ImagePicker.requestMediaLibraryPermissionsAsync();
34      if (status !== 'granted') {
35          Alert.alert('Permissão necessária', 'permissão necessária para acessar suas fotos.');
36      }
37  };
```

- Criação da função **selecionarImagem,** que faz a chamada da função solicitarPermissao, antes de ser executada. A variável resultado irá guardar a imagem selecionada. Caso o resultado seja diferente de ‘canceled’, significa que a imagem foi selecionada corretamente, e então poderá ser setada. Além disso, foi utilizado a biblioteca image-picker para permitir que o usuário realize o upload de imagem.

```
const selecionarImagem = async () => {
    await solicitarPermissao();

    let resultado = await ImagePicker.launchImageLibraryAsync({
        mediaTypes: ImagePicker.MediaTypeOptions.Images,
        allowsEditing: true,
        aspect: [4, 3],
        quality: 1,
    });

    if (!resultado.canceled) {
        setImagem(resultado.assets[0].uri);
    }
};
```

- Criação da função **renderizarFornecedor**, para que seja possível exibir as informações e a imagem associada ao fornecedor. Caso o fornecedor não tenha imagem, irá aparecer no lugar a mensagem “sem imagem”.

```
const renderizarFornecedor = ({ item }) => (
    <View>
        {item.imagem ? (
            <Image style={styles.imagemPreview} source={{ uri: item.imagem }}
        ) : (
            <Text>Sem imagem</Text>
        )}
        <Text >Nome: {item.nome}</Text>
        <Text>CNPJ: {item.cnpj}</Text>
        <Text>Telefone: {item.telefone}</Text>
    </View>
);
```

- Criação do componente **App(),** o qual irá retornar uma visualização (View) contendo o formulário de cadastro de fornecedor com entrada para nome, cnpj, telefone e imagem do fornecedor. Ao clicar no botão Cadastrar Fornecedor, a função **adicionarFornecedor** é acionada.

```
64    return (
65        <View style={styles.container}>
66            <Text>cadastro de fornecedores</Text>
67            <TextInput placeholder='nome' value={nome} onChangeText={setNome} />
68            <TextInput placeholder='cnpj' value={cnpj} onChangeText={setCnpj} />
69            <TextInput placeholder='telefone' value={telefone} onChangeText={setTelefone} />
70            <Button title='Selecionar Imagem' onPress={selecionarImagem} />
71            <Button title='Cadastrar fornecedor' onPress={adicionarFornecedor} />
72            <FlatList data={fornecedores} keyExtractor={(item) => item.id} renderItem={renderizarFornecedor}
73
74
75
76        </View>
77    );
78  }
79
```

- Criação da constante **styles** que cria um objeto de estilo. Dentro do objeto, temos outros dois objetos (container e imagemPreview) com seus respectivos atributos e valores correspondentes ao CSS.

```
80  const styles = StyleSheet.create({
81      container: {
82          flex: 1,
83          backgroundColor: '#fff',
84          alignItems: 'center',
85          justifyContent: 'center',
86          padding: 100,
87      },
88      imagemPreview: {
89          width: 100,
90          height: 100,
91      }
92  });
```

**Resultados**

cadastro de fornecedores
nome
cnpj
telefone

SELECIONAR IMAGEM

CADASTRAR FORNECEDOR


Nome: Coca cola
CNPJ: 2618726712/000-1
Telefone: (11) 3237-7332


Nome: Pepsi
CNPJ: 2172172121/000-2
Telefone: (19) 3233-3282


## Conclusão

No desenvolvimento do aplicativo de cadastro de fornecedores, foi possível utilizar os principais conceitos de React Native. Entre eles, componente de <Text> para

exibição de texto, <TextInput> para permitir que o usuário faça entrada de texto, <Image> para exibição de imagem, <FlatList> para exibir uma lista e o uso de recurso como o StyleSheet para aplicação de estilo nos componentes utilizados.