



Missão Prática | Nível 1 | Mundo 4

Camilla Gomes – 202302631673@alunos.estacio.br

Campus Nova América

Disciplina: RPG0023 Vamos criar um App – Turma 9001 – 2024.3

Objetivos 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.

Contextualização

A empresa "Meeting" busca criar um aplicativo móvel eficaz para o cadastro de fornecedores, com listas e imagens de alta qualidade, economizando recursos e proporcionando uma excelente experiência ao usuário. A escolha da tecnologia React Native é crucial para estabelecer uma presença sólida no mercado móvel. Nesta atividade, você aprenderá os princípios básicos do React Native.

Requisitos Funcionais:

1. **Cadastro de Fornecedores:** O aplicativo deve permitir o cadastro de fornecedores, incluindo informações detalhadas, como nome, endereço, contato e categorias de produtos fornecidos. Essas informações serão exibidas utilizando componentes como `<Text>`, `<TextInput>`, e `<Image>`.

2. **Listagem de Fornecedores:** Deve ser possível visualizar uma lista de fornecedores cadastrados, com opções de pesquisa e filtragem com base em critérios como categoria ou localização. A lista de fornecedores será exibida utilizando componentes como `<Text>` e `<Image>`.
3. **Associação de Imagens:** O aplicativo deve permitir a associação de imagens aos perfis dos fornecedores. Os usuários devem poder fazer o upload de logotipos ou fotos relacionadas ao fornecedor, utilizando o componente `<Image>`.
4. **Experiência de Usuário Intuitiva:** A interface do aplicativo deve ser intuitiva e fácil de usar, garantindo que os usuários possam navegar, adicionar e editar informações de forma eficiente. Isso será alcançado usando componentes como `<Text>`, `<TextInput>`, e `<Image>`.

Códigos: <https://github.com/Camilla-Alves/Meeting>

Documentação do projeto

App.tsx

Importações

- React: Biblioteca principal.
- NavigationContainer: Gerencia a navegação do app.
- createStackNavigator: Cria o fluxo de navegação em pilhas (Stack Navigator).
- Header: Componente de cabeçalho customizado.
- Cadastro, Lista, EditFornecedor: Telas do app.

Stack

- Define a pilha de navegação usada para alternar entre as telas.

Função App

- NavigationContainer: Envolve toda a navegação.
- Stack.Navigator:
 - initialRouteName: Define que o app começa na tela "Cadastro".
 - screenOptions: Define o cabeçalho customizado Header e o torna visível.

Telas:

1. *Cadastro: Tela inicial. Mostra um formulário de cadastro.*
2. *Lista: Exibe a lista de fornecedores.*
3. *EditFornecedor: Tela para editar um fornecedor existente.*

src/components/Header/index.tsx

Importações

- React: Biblioteca principal.
- View, Text, Image, SafeAreaView, StatusBar, TouchableOpacity: Componentes do React Native para construir a interface.
- useNavigation, useRoute: Hooks do React Navigation para navegação e roteamento.
- Icon: Biblioteca de ícones para React Native.
- styles: Importa os estilos definidos em styles.ts, localizado na mesma pasta.

Componente Header

- Objetivo: Renderiza o cabeçalho do aplicativo, com um botão de voltar condicional, um logo e um título.
- navigation: Hook para navegação, usado para voltar à tela anterior.
- route: Hook para acessar informações sobre a rota atual, para condicionalmente mostrar o botão de voltar se não estiver na tela "Cadastro".

Renderização

- SafeAreaView: Garante que o conteúdo esteja visível dentro das áreas seguras da tela.
- StatusBar: Configura o estilo da barra de status.
- View: Container principal para o cabeçalho.
- TouchableOpacity: Botão de voltar, exibido apenas se não estiver na tela inicial.
- Image: Logo do aplicativo.
- Text: Título do cabeçalho.

Estilos

- styles: Aplica os estilos específicos ao cabeçalho, definidos em styles.ts na mesma pasta.

src/components/Fornecedor/index.tsx

Importações

- React: Biblioteca principal para criar componentes.
- View, Text, TouchableOpacity, Image: Componentes do React Native.

- `styles`: Estilos importados de `src/styles/styles`.

Tipagem

- `FornecedorType`: Dados do fornecedor (nome, endereço, contato, categorias, e imagem opcional).
- `Props`: Inclui dados do fornecedor e funções `onRemove` e `onEdit`.

Função Fornecedor

- **Objetivo**: Exibir detalhes do fornecedor com opções para edição e remoção.
 - **Imagem**: Mostra a imagem do fornecedor se `imagemURL` estiver disponível.
 - **Detalhes**: Exibe nome, endereço, contato e categorias.
 - **Botões**:
 - **Editar**: Aciona `onEdit` para modificar o fornecedor.
 - **Remover**: Aciona `onRemove` para excluir o fornecedor.

Estilos

- `styles`: Importado de `src/styles/styles`. Define o layout e a aparência do componente, incluindo margens, cores e alinhamento, para manter a consistência visual com o restante do app.

`src/screens/Cadastro/index.tsx`

Importações

- `React`, `useState`: Biblioteca principal para criar componentes e gerenciar estados.
- `View`, `Text`, `TextInput`, `TouchableOpacity`, `Alert`: Componentes do React Native para construir a interface.
- `AsyncStorage`: Biblioteca para armazenamento persistente de dados no dispositivo.
- `styles`: Estilos importados de `src/styles/styles` para estilizar o componente.
- `FornecedorType`: Tipo de dados importado de `src/components/Fornecedor`.

Função Cadastro

- **Objetivo**: Permitir o cadastro de novos fornecedores com validação e armazenamento dos dados.
 - **Estados**: Usa hooks de estado para gerenciar os valores dos campos (nome, endereço, contato, categorias, `imagemURL`).

- `handleAddFornecedor`:
 - Validação: Verifica se todos os campos obrigatórios estão preenchidos. Exibe um alerta se algum campo estiver vazio.
 - Armazenamento: Lê e atualiza a lista de fornecedores no `AsyncStorage`. Adiciona o novo fornecedor se não houver um fornecedor com o mesmo nome.
 - Navegação: Redireciona para a tela "Lista" após a adição bem-sucedida do fornecedor.
 - Reset: Limpa os campos do formulário após a adição.

Renderização

- Formulário:
 - `TextInput`: Campos para inserir o nome, endereço, contato, categorias e URL da imagem do fornecedor.
 - Botões:
 - Adicionar: Aciona `handleAddFornecedor` para adicionar o fornecedor.
 - Lista de Fornecedores: Navega para a tela "Lista" onde os fornecedores cadastrados são exibidos.

Estilos

- `styles`: Importado de `src/styles/styles`. Aplica o design ao formulário e botões, garantindo uma aparência consistente com o restante do aplicativo.

`src/screens/Lista/index.tsx`

Importações

- `React`, `useState`, `useEffect`, `useCallback`: Biblioteca principal e hooks para estados e efeitos.
- `useFocusEffect`: Executa efeitos quando a tela ganha foco.
- `AsyncStorage`: Armazenamento assíncrono para dados persistentes.
- `Picker`: Componente para seleção de categorias e localizações.
- `View`, `Text`, `TextInput`, `FlatList`, `Alert`, `TouchableOpacity`: Componentes do React Native.
- `styles`: Estilos importados de `src/styles/styles`.
- `Fornecedor`, `FornecedorType`: Componente e tipo de dados do fornecedor.

Função Lista

- Objetivo: Tela para listar, filtrar, editar e remover fornecedores.
 - Estados:
 - fornecedores: Lista carregada do armazenamento.
 - search: Texto da busca para filtrar por nome.
 - selectedCategory e selectedLocation: Valores selecionados para filtragem.
 - categories e locations: Categorias e localizações disponíveis para filtragem.
 - Funções:
 - loadFornecedores: Carrega fornecedores do AsyncStorage e atualiza categorias e localizações.
 - useFocusEffect: Atualiza fornecedores toda vez que a tela ganha foco.
 - useEffect: Adiciona um novo fornecedor se o parâmetro novoFornecedor for atualizado.
 - handleSearch: Atualiza o texto da busca.
 - handleCategoryChange e handleLocationChange: Atualizam as categorias e localizações selecionadas.
 - filteredFornecedores: Filtra fornecedores baseado na busca, categoria e localização.
 - handleRemove: Remove fornecedor após confirmação.
 - handleEdit: Navega para a tela de edição do fornecedor.

Renderização

- Busca e Filtros:
 - TextInput: Campo para buscar fornecedores por nome.
 - Picker: Seletores para categorias e localizações.
- Lista de Fornecedores:
 - FlatList: Exibe fornecedores filtrados. Usa o componente Fornecedor para cada item.
 - ListEmptyComponent: Mensagem se não houver fornecedores.
- Botão de Cadastro:
 - TouchableOpacity: Navega para a tela de cadastro.

Estilos

- `styles`: Importado de `src/styles/styles`. Aplica formatação e layout aos elementos da tela, garantindo consistência visual.
-

src/screens/EditFornecedor/index.tsx

Importações:

- `React`: Biblioteca principal para criar o componente.
- `useState`, `useEffect`: Hooks do React para gerenciar estado e efeitos colaterais.
- `View`, `Text`, `TextInput`, `TouchableOpacity`: Componentes do React Native para layout e interação.
- `styles`: Importa os estilos do arquivo separado (`src/styles/styles`).
- `AsyncStorage`: Para armazenar e recuperar dados localmente.

Propriedades:

- `route`: Recebe o fornecedor a ser editado via parâmetros de navegação.
- `navigation`: Utilizado para navegar de volta após salvar as alterações.

Estados:

- `nome`, `endereço`, `contato`, `categorias`, `imagemURL`: Dados do fornecedor.

Função Principal:

- `handleSave`:
 - Descrição: Atualiza os dados do fornecedor no `AsyncStorage`.
 - Funcionamento:
 1. Obtém a lista atual de fornecedores.
 2. Atualiza o fornecedor com os novos dados.
 3. Armazena a lista atualizada no `AsyncStorage`.
 4. Navega de volta para a tela anterior.
 - Erro: Exibe um alerta se a atualização falhar.

Renderização:

- Formulário de Edição:
 - Campos para editar nome, endereço, contato, categorias e URL da imagem.
 - Botão "Salvar" que chama `handleSave` ao ser pressionado.

Estilos:

- Estilos importados: Utiliza `styles` de `src/styles/styles` para manter a consistência e permitir a reutilização em diferentes componentes.
-

Conclusão:

O app "Meeting" foi desenvolvido para tornar o cadastro e gerenciamento de fornecedores mais fácil e direto. Com React Native, o foco foi criar uma ferramenta prática e que funcione bem, garantindo uma experiência tranquila para o usuário.

Durante o desenvolvimento, acabei adicionando alguns recursos extras além do que foi pedido. Para isso, precisei pesquisar e explorar soluções novas, o que me ajudou a conhecer e entender melhor o React Native e suas ferramentas.

O app está funcionando conforme o esperado e atende aos requisitos principais. Ele é fácil de usar e eficaz para o cadastro e gerenciamento de fornecedores. As soluções adotadas e as tecnologias utilizadas garantem uma experiência de usuário eficiente e demonstram um bom domínio dos conceitos do React Native.

Demonstrações do Aplicativo:

No primeiro print, temos a tela inicial do aplicativo, que é a tela de cadastro de fornecedores. Nela também é possível ir direto para a listagem de fornecedores.

No segundo print, temos a tela de listagem de fornecedores, onde o usuário pode pesquisar por nome ou selecionar de acordo com a categoria ou localização. Também é possível editar ou remover o fornecedor.

No terceiro print, damos foco nas opções de pesquisa e filtragem que foram mencionadas anteriormente.

No quarto print, temos a tela onde é possível editar as informações do fornecedor.



Meeting

Cadastro

Adicionar

Lista de Fornecedores

8:08



Meeting

Fornecedores

Pesquisar por nome

Todas as Categorias



Todas as Localizações



Nome: Seu Ze

Endereço: Rio de Janeiro

Contato: ze@gmail.com

Categorias: Frutas

[Editar](#)

[Remover](#)

Cadastrar

8:08



Meeting

Fornecedores

Pesquisar por nome

Sucos



Todas as Localizações

Rio de Janeiro

Minas Gerais

Nome: Carlitos

Endereço: Minas Gerais

Contato: 99999-9999

Categorias: Sucos

Editar

Remover

Cadastrar

8:08



Meeting

Editar Fornecedor

Seu Ze

Rio de Janeiro

ze@gmail.com

Frutas

<https://cdn-icons-png.flaticon.com/512/0/460.png>

Salvar