Missão Prática Nível 1 – Mundo 4

Fernanda Canto P. da Costa - 202208379788

Campus de Ipanema

Vamos criar um App – Turma 22.3 – 4ºsemestre

Github: https://github.com/nandacpc/Missao1-Mundo4

Introdução

Documentação referente ao projeto Meeting, desenvolvido utilizando React Native, destinado a simplificar o processo de cadastro e gerenciamento de fornecedores. Este documento visa fornecer uma descrição da funcionalidade principal, arquitetura e componentes do App.

Escopo

O aplicativo Meeting atende às seguintes necessidades:

Cadastro de Fornecedores: Usuários podem inserir informações detalhadas de novos fornecedores, incluindo nome, endereço, contato, categorias de produtos fornecidos e uma imagem representativa.

Listagem de Fornecedores: O aplicativo oferece uma visão geral dos fornecedores cadastrados, com funcionalidades de pesquisa e filtragem baseadas em critérios como categoria ou localização.

Associação de Imagens: Permite aos usuários fazerem upload de logotipos ou fotos relacionadas aos fornecedores, enriquecendo o perfil de cada um.

Experiência de Usuário Intuitiva: Criado para garantir que os usuários possam navegar, adicionar e editar informações de forma eficiente.

Componentes do Aplicativo

O aplicativo Meeting é composto por três componentes principais que facilitam o cadastro e a visualização de fornecedores. Estes componentes são *SupplierForm.js* para o cadastro de novos fornecedores, *SupplierList.js* para listar todos os fornecedores cadastrados, e *SupplierListItem.js* que define como cada fornecedor é apresentado na lista.

SupplierForm.js:

Descrição:
SupplierForm.js é responsável por renderizar o formulário de cadastro de fornecedores no aplicativo. Este formulário coleta informações como nome, endereço, contato, categorias de produtos fornecidos, e uma imagem representativa.

Estado:
O estado do componente contém os campos name, address, contact, categories, e imageUrl, representando as informações do fornecedor.

Funções Importantes:

updateField(key, value): Atualiza o estado do componente com as informações fornecidas pelo usuário.

handleImageSelection(): Abre a galeria de fotos do dispositivo permitindo ao usuário selecionar uma imagem para o fornecedor.

handleSubmit(): Valida os dados do formulário e, se estiverem corretos, adiciona o novo fornecedor ao banco de dados fictício e exibe uma mensagem de sucesso.

SupplierList.js:

Descrição:
SupplierList.js exibe uma lista de todos os fornecedores cadastrados. Inclui funcionalidades de pesquisa e filtragem para encontrar fornecedores específicos baseados em nome, endereço ou categorias.

Estado:
O estado do componente inclui suppliers, que é a lista completa de fornecedores, filteredSuppliers, que é a lista de fornecedores após aplicar o filtro de pesquisa, e searchQuery, que é a consulta de pesquisa atual.

Funções Importantes:

handleSearch(query): Atualiza a consulta de pesquisa e filtra a lista de fornecedores baseando-se no texto de entrada.

SupplierListItem.js:

Descrição:
SupplierListItem.js define a aparência de cada item da lista de fornecedores. Mostra informações como o nome do fornecedor, endereço, contato, e uma imagem.

Propriedades:

supplier: Um objeto contendo as informações do fornecedor a ser exibido.

Layout:
Cada item da lista é exibido em uma linha, com a imagem do fornecedor à esquerda seguida por seus detalhes. O estilo é definido para manter a consistência visual em toda a lista.

Integração com Banco de Dados Fictício

O projeto incorpora um modelo simplificado de armazenamento de dados usando um "banco de dados fictício" para simular a interação com um sistema de banco de dados real. Essa abordagem permite o desenvolvimento e teste do aplicativo sem a necessidade de uma infraestrutura de back-end completa, facilitando a prototipagem rápida e a demonstração de funcionalidades chave do aplicativo.

Estrutura do Banco de Dados Fictício

O banco de dados fictício é implementado como um array JavaScript no arquivo **fakeDatabase.js**. Cada entrada neste array representa um fornecedor, contendo informações como **id**, **name**, **address**, **contact**, **categories**, e **imageUrl**.

```
let fakeDatabase = [ // Exemplo de fornecedor { id: 1, name: "Fornecedor A", address: "Endereço A", contact: "Contato A", categories: "Categoria A", imageUrl: "URL da Imagem A" }, ];
```

Operações do Banco de Dados Fictício

O **fakeDatabase.js** expõe funções para realizar operações CRUD (Create, Read, Update, Delete) básicas, simulando interações comuns com um banco de dados:

Create (Criar): A função **addSupplier** adiciona um novo fornecedor ao array **fakeDatabase**, atribuindo a ele um ID único.

Read (Ler): A função **getSuppliers** recupera a lista completa de fornecedores, oferecendo também a capacidade de filtrar fornecedores com base em critérios específicos, como nome ou categoria.

Implementação da Função de Adição

A função **addSupplier** simula a adição de um novo fornecedor ao banco de dados. Ela aceita um objeto **supplier** como argumento, gera um novo ID baseado no comprimento do array **fakeDatabase**, e adiciona o fornecedor ao array.

```
export const addSupplier = (newSupplier) => { const newId = fakeDatabase.length + 1; const supplierWithId = { ...newSupplier, id: newId }; fakeDatabase.push(supplierWithId); return Promise.resolve(supplierWithId);
```

Implementação da Função de Leitura

A função **getSuppliers** retorna a lista atual de fornecedores. Esta função pode ser expandida para incluir filtragem e ordenação baseada em critérios específicos, como localização ou categoria de produto.

```
export const getSuppliers = () => { return Promise.resolve(fakeDatabase);
```

Conclusão

A documentação do projeto Meeting apresentou uma visão abrangente deste aplicativo React Native, destacando sua funcionalidade de facilitar o cadastro e a gestão de fornecedores.

A implementação do projeto Meeting demonstra não apenas a flexibilidade e eficiência do React Native como uma plataforma de desenvolvimento de aplicativos, mas também enfatiza a importância de uma arquitetura bem pensada na hora de desenvolver um software. O uso de um banco de dados fictício para simular a interação com um sistema de gestão de dados reforça a viabilidade do projeto para prototipagem rápida e testes, enquanto prepara o terreno para futuras integrações com sistemas de back-end reais.