Documentação Detalhada do Teste 7 Pay

Visão Geral

O sistema é uma aplicação Flutter desenvolvida para pesquisar e cadastrar endereços. Ele se integra à API ViaCEP para obter informações detalhadas sobre endereços a partir de consultas específicas. A aplicação tem duas telas principais: `AdressPage` para pesquisa e visualização de endereços e `RegisterAdressPage` para filtrar, selecionar e salvar endereços.

Requisitos do Sistema

Hardware

- Dispositivo Android ou iOS para execução do aplicativo.
- Emulador ou dispositivo físico para testes.

Software

- Flutter SDK instalado na máquina.
- Dependências do projeto instaladas (`flutter pub get`).

Configuração do Ambiente

Instalação do Flutter:

- Siga as instruções de instalação do Flutter disponíveis em https://flutter.dev/docs/get-started/install.

Obtenção do Projeto:

- Clone o repositório do projeto:

```
git clone https://github.com/Matheus-o-alves/teste_7_pay
cd nome-do-repositorio
```

Instalação de Dependências:

- No diretório do projeto, execute o seguinte comando para obter as dependências necessárias:
- flutter pub get

Execução da Aplicação

- Execute o seguinte comando no terminal, dentro do diretório do projeto, para iniciar a aplicação no emulador ou dispositivo conectado:
- flutter run

Estrutura do Projeto

A estrutura de arquivos do projeto é organizada da seguinte maneira:

- `lib/`
  - `main.dart`: Ponto de entrada da aplicação.
  - `models/`: Contém o modelo `AdressModel` utilizado para representar um endereço.
  - `data/usecases/`: Contém o serviço `RemoteAdress` responsável por realizar requisições HTTP para obter endereços.
  - `presenter/`: Contém os stores `AdressStore` e `RegisterAdressStore` que gerenciam o estado da aplicação.
  - `pages/`: Contém as páginas da aplicação, sendo `AdressPage` a tela principal e `RegisterAdressPage` a tela de cadastro.
  - `widgets/`: Contém componentes reutilizáveis, como o drawer (`DrawerWidget`).

Tecnologias Utilizadas

A aplicação faz uso de diversas tecnologias e pacotes para fornecer funcionalidades eficientes e uma boa experiência ao usuário:

- Flutter: Framework de desenvolvimento de aplicativos móveis.
- Dart: Linguagem de programação utilizada no Flutter.
- MobX: Biblioteca de gerenciamento de estado.
- Get: Biblioteca para navegação e gerenciamento de rotas.
- http: Pacote para realizar requisições HTTP.
- Provider: Gerenciamento de estado para Flutter.
- Path Provider: Facilita a obtenção de diretórios locais no dispositivo.

Funcionalidades do Sistema

1 - Login Page:

A tela de login é a interface que permite aos usuários inserirem suas credenciais para acessar o sistema. Essa tela foi projetada com ênfase na usabilidade e design agradável, seguindo padrões modernos de UI/UX.

Componentes Principais:

AppBar:

- A barra superior exibe o título "Login" e possui um fundo na cor `0xFF393939`.

Corpo da Tela:

- O corpo da tela contém os seguintes componentes:
  a. Logo:
  - Uma imagem do logo da empresa é exibida na parte central superior da tela.
- b. Campos de Entrada (TextField):
  - Dois campos de entrada para o nome de usuário e senha.
  - Os campos possuem bordas arredondadas para uma estética mais moderna.
  - A cor de fundo é `0xFF393939`, e o texto é exibido em branco (`Colors.white`).
  - Quando o campo está em foco, a borda é destacada em branco.
  - Se houver um erro no campo, a borda do campo fica vermelha.

- c. Botão de Entrar (ElevatedButton):
    - Um botão elevado que permite ao usuário realizar o login.
    - O botão é clicável sempre, mas exibe uma mensagem de erro caso os campos não atendam aos requisitos mínimos.
    - O design do botão é aprimorado com bordas arredondadas e cores atraentes (`0xFFF08F21` quando habilitado e `0xFF393939` quando desabilitado).
    - Quando o login está em andamento, um indicador de progresso é exibido.

Estado e Lógica:

LoginStore:

- A classe `LoginStore` gerencia o estado e a lógica relacionados à tela de login.
- Ela controla a entrada do nome de usuário e senha.
- Verifica se os campos atendem aos requisitos mínimos (pelo menos 6 caracteres) quando o botão de login é pressionado.
- Realiza a ação de login quando os campos são válidos.

Observadores (Observer):

Observador de Nome de Usuário:

- Monitora as alterações no campo do nome de usuário.
- Exibe uma mensagem de erro se o nome de usuário não atender aos requisitos mínimos.

Observador de Senha:

- Monitora as alterações no campo de senha.
- Exibe uma mensagem de erro se a senha não atender aos requisitos mínimos.

Observador de Botão de Entrar:

- Monitora as alterações no botão de entrar.
- Exibe um indicador de progresso quando o login está em andamento.

Melhorias Futuras:

Estilização Aprimorada:

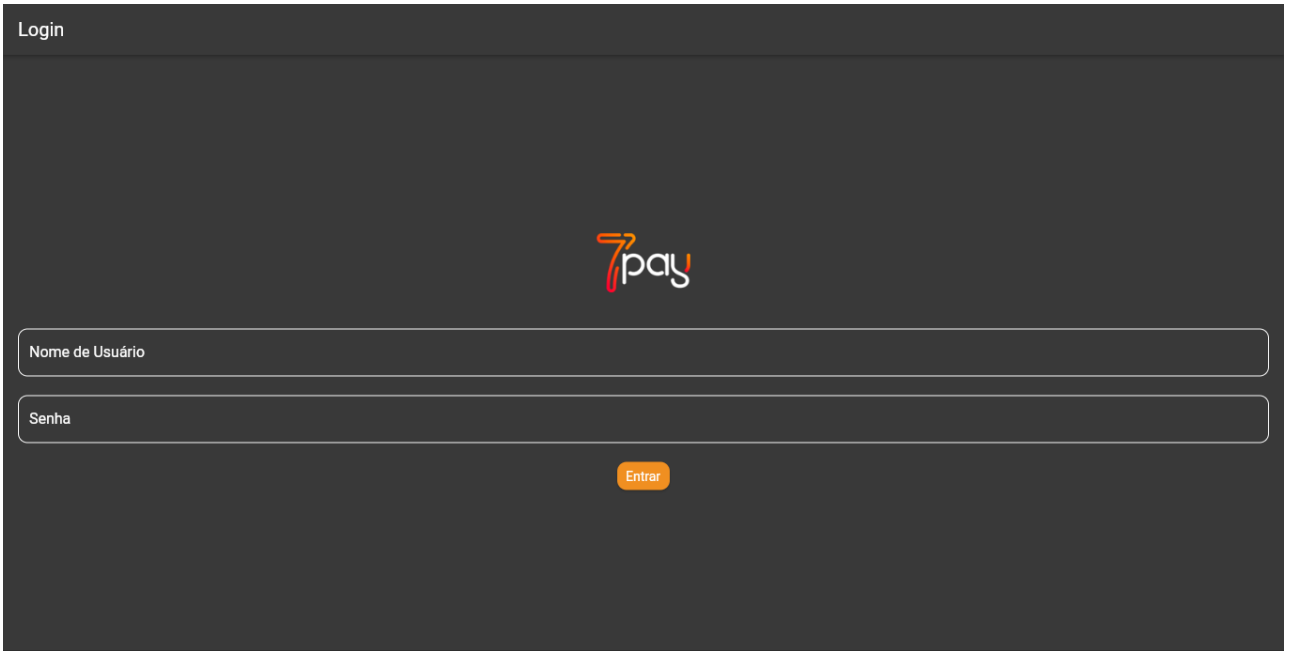
- Adição de animações e transições para uma experiência mais fluida.

Mensagens de Erro Dinâmicas:

- Aprimorar as mensagens de erro para fornecer feedback mais específico.

Adaptação Responsiva:

- Garantir que a tela seja responsiva em diferentes dispositivos.



(Figura 1)

1 - Pesquisar Endereço (AdressPage)

- Descrição:

  Os usuários podem pesquisar endereços informando o UF (Unidade Federativa) e o Bairro.

  Os endereços correspondentes são exibidos em uma tabela, mostrando informações como CEP, Logradouro, Complemento, Bairro, Localidade, UF, Ibge, e se estão selecionados.

  É possível filtrar os endereços informando o UF e o Bairro desejados.

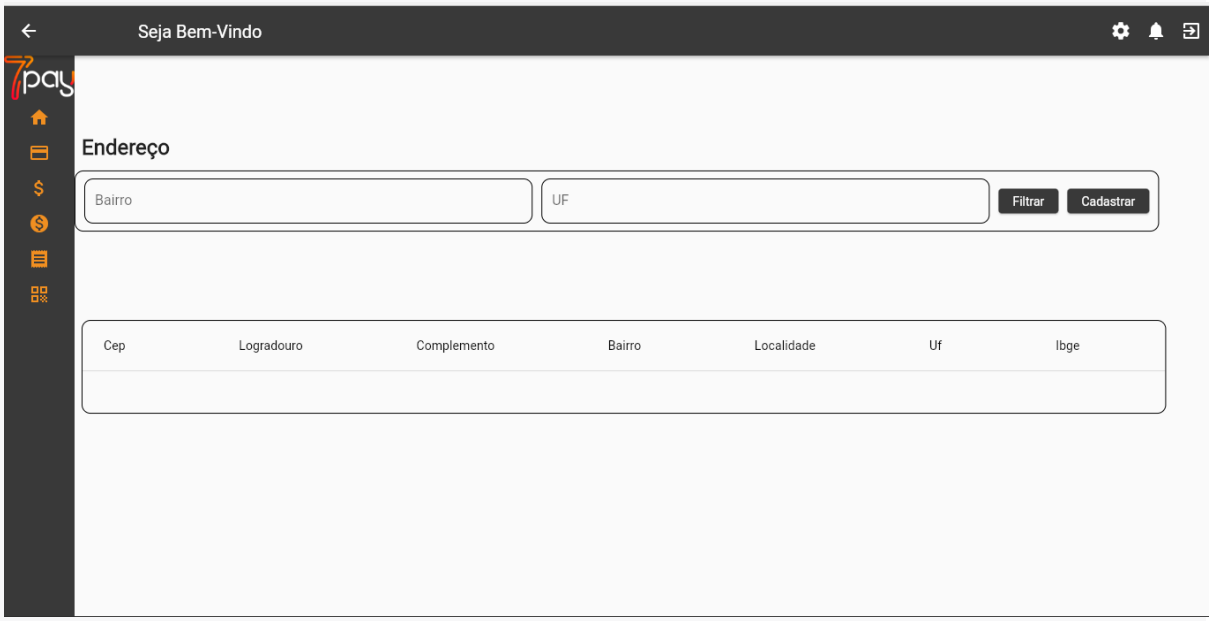
  Os endereços filtrados são exibidos em tempo real na tabela.

- Fluxo:

  O usuário interage com a tela principal (AdressPage) para pesquisar e visualizar endereços.

  A classe `AdressStore` gerencia o estado da tela principal, incluindo a lista de endereços, filtros e mensagens de erro.

A classe `RemoteAdress` é responsável por fazer requisições HTTP para obter endereços a partir da API ViaCEP.
Os endereços são exibidos na tabela, e os usuários podem filtrar os resultados.



(Figura 2)

3 - Cadastrar Endereço (RegisterAdressPage)

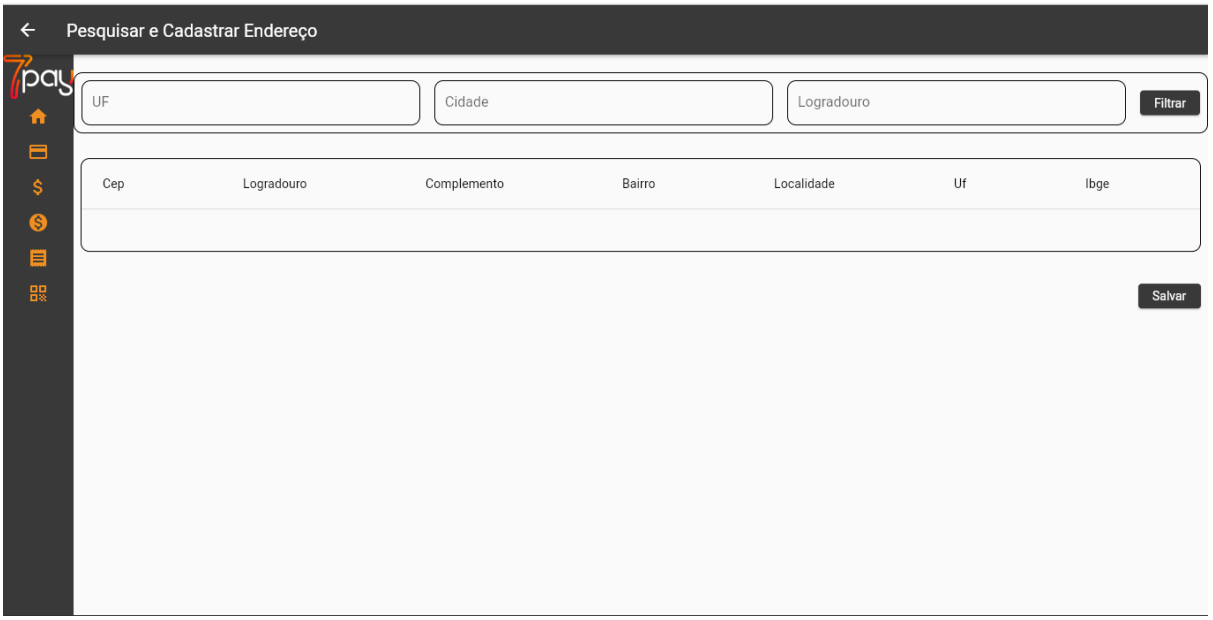
- Descrição:

  Os usuários podem cadastrar endereços clicando no botão "Cadastrar" na tela principal (AdressPage).
  São redirecionados para a tela de cadastro (RegisterAdressPage), onde podem filtrar e selecionar endereços.
  Os endereços selecionados são exibidos em uma tabela, mostrando as mesmas informações da pesquisa.
  Os usuários podem salvar os endereços selecionados, clicando no botão "Salvar".
  Os endereços salvos são exibidos novamente na tela principal (AdressPage).

- Fluxo:

  O usuário clica no botão "Cadastrar" na tela principal (AdressPage).
  Ele é redirecionado para a tela de cadastro (RegisterAdressPage).

Na tela de cadastro, o usuário filtra endereços usando UF, Bairro e Logradouro.
Os endereços filtrados são exibidos em uma tabela.
O usuário seleciona os endereços desejados clicando nas checkboxes.
Os endereços selecionados são exibidos na tabela com uma marcação visual.
O usuário clica no botão "Salvar" para salvar os endereços selecionados.
Os endereços salvos são exibidos novamente na tela principal (AdressPage).



(Figura 3)

Fluxo de Dados

O usuário interage com a tela principal (AdressPage) para pesquisar e visualizar endereços.
A classe `AdressStore` gerencia o estado da tela principal, incluindo a lista de endereços, filtros e mensagens de erro.
A classe `RemoteAdress` é responsável por fazer requisições HTTP para obter endereços a partir da API ViaCEP.
Os endereços são exibidos na tabela, e os usuários podem filtrar os resultados.

Fluxo de Cadastro

O usuário clica no botão "Cadastrar" na tela principal (AdressPage).
Ele é redirecionado para a tela de cadastro (RegisterAdressPage).
Na tela de cadastro, o usuário filtra endereços usando UF, Bairro e Logradouro.
Os endereços filtrados são exibidos em uma tabela.
O usuário seleciona os endereços desejados clicando nas checkboxes.
Os endereços selecionados são exibidos na tabela com uma marcação visual.
O usuário clica no botão "Salvar" para salvar os endereços selecionados.
Os endereços salvos são exibidos novamente na tela principal (AdressPage).

Observações

- A aplicação utiliza o padrão MobX para gerenciamento de estado, proporcionando reatividade nas atualizações da interface.
- O pacote Get é utilizado para navegação e gerenciamento de rotas.
- O sistema possui um conjunto de testes para verificar o comportamento das funcionalidades principais.

Nota: Certifique-se de que a máquina possui as dependências necessárias e os pré-requisitos instalados antes de executar a aplicação. Para obter mais informações, consulte a documentação oficial do Flutter: https://flutter.dev/docs