FAKE MEGA STORE - DOCUMENTAÇÃO TÉCNICA COMPLETA (Versão

1. INTRODUÇÃO GERAL AO PROJETO

O aplicativo Fake Mega Store é um aplicativo educacional desenvolvido em Flutter com o objetivo de simular uma loja virtual de roupas e produtos diversos, utilizando a FakeStoreAPI como fonte de dados.

O foco do projeto é didático, permitindo estudar: - Consumo de API REST. - Navegação entre telas usando rotas nomeadas. - Gerenciamento manual de estado. - Estruturação de código por responsabilidade. - Interação do usuário com widgets fundamentais do Flutter.

Todos os dados do app são mantidos em memória através da classe AppState. Nenhum dado é persistido localmente ou em banco de dados.

2. PUBSPEC.YAML E DEPENDÊNCIAS

O arquivo pubspec.yaml define as dependências e configurações gerais do projeto Flutter.

Dependências utilizadas: - flutter (SDK principal) - http

O pacote http é responsável por: - Realizar requisições GET para a FakeStoreAPI - Converter respostas JSON em objetos Dart

Após qualquer modificação no pubspec.yaml, deve-se executar: flutter pub get

Não são utilizados pacotes de persistência ou gerenciamento avançado de estado, reforçando o caráter educacional do projeto.

3. ESTRUTURA DE PASTAS

lib/ main.dart rotas.dart

estado/ app_state.dart

modelos/ produto.dart

servicos/ api_service.dart

pages/ login_page.dart registro_page.dart loja_page.dart detalhes_produto_page.dart carrinho_page.dart perfil_page.dart

4. DESCRIÇÃO DOS ARQUIVOS

main.dart - Responsável por iniciar o aplicativo. - Configura MaterialApp: título, tema, rotas e tela inicial. - Instancia AppState e o distribui às telas.

rotas.dart - Centraliza todos os nomes de rotas do app. - Evita uso de strings mágicas.

app_state.dart Classe responsável por armazenar o estado global.

Variáveis relevantes: - nomeUsuario - emailUsuario - cidadeEntrega - estadoEntrega - cepEntrega - bairroEntrega - ruaEntrega - numeroEntrega - itensCarrinho

Funções principais: - fazerLogin() - fazerLogout() - atualizarPerfil() - adicionarAoCarrinho() - removerUmaUnidadeDoCarrinho() - esvaziarCarrinho() - totalCarrinho() - quantidadeTotalCarrinho()

Todas as telas acessam este estado como referência compartilhada.

produto.dart Modelo que representa o objeto Produto.

Contém: - id - titulo - descricao - preco - imagemUrl - categoria

Utiliza o método fromJson para conversão dos dados vindos da API.

api_service.dart Classe responsável pela comunicação com a FakeStoreAPI.

Função principal: - buscarProdutos()

Responsável por: - Realizar requisição GET - Decodificar JSON - Retornar lista de produtos - Tratar erros básicos.

5. TELAS PRINCIPAIS

LoginPage - Entrada simulada do usuário. - Campos de e-mail e senha. - Botões de navegação.

RegistroPage - Formulário de cadastro. - Validação básica dos dados.

LojaPage - Exibe produtos utilizando FutureBuilder. - Uso de GridView.builder. - Navegação para detalhes do produto.

DetalhesProdutoPage - Mostra todos os dados do produto. - Botão para adicionar ao carrinho.

CarrinhoPage - Lista os produtos selecionados. - Permite ajustar quantidades. - Finalização simulada de compra.

PerfilPage - Permite visualizar e editar dados. - Campos com validação de números.

6. PRINCIPAIS WIDGETS UTILIZADOS

- Scaffold - AppBar - Column / Row - TextField - ElevatedButton - GestureDetector - GridView.builder - ListView.builder - FutureBuilder - SnackBar - AlertDialog

7. CHECKLIST - ANÁLISE TÉCNICA

Fluxo e rotas compreensíveis: SIM Parâmetros entre telas: SIM Uso de setState: SIM Gerenciadores reativos avançados: NÃO GestureDetector: SIM Slider / Radio: NÃO ListView.builder com API: SIM Estados de loading e erro: SIM Persistência local: NÃO SnackBar funcional: SIM Aplicação compilável: SIM

8. CONSIDERAÇÕES FINAIS

Este projeto demonstra domínio dos fundamentos do Flutter para aplicações educacionais. Ele é adequado como material de estudo e referência prática para alunos que estão iniciando no desenvolvimento mobile com Flutter.