**Documentação: Aplicativo Banco Digital**

**Visão Geral**

O Banco Digital é um aplicativo móvel desenvolvido usando o framework Flutter, que simula as funcionalidades básicas de um banco digital. O aplicativo oferece uma interface amigável para os usuários realizarem operações bancárias comuns, como login, visualização de saldo e extrato, transferências e consulta de cotações de moedas.

**Arquitetura**

O aplicativo segue uma arquitetura baseada em rotas nomeadas para gerenciamento de navegação entre telas, com separação de responsabilidades entre:

* **UI (Interface do Usuário)**: Widgets e telas
* **Lógica de Negócios**: Classes de serviços e controladores
* **Modelos de Dados**: Representação dos dados manipulados pelo aplicativo

**Estrutura de Pastas**

lib/

├── main.dart # Ponto de entrada do aplicativo

├── routes/ # Configuração de rotas nomeadas

│ └── app\_routes.dart

├── models/ # Modelos de dados

│ ├── user.dart

│ ├── transaction.dart

│ └── currency.dart

├── screens/ # Telas do aplicativo

│ ├── login\_screen.dart

│ ├── home\_screen.dart

│ ├── currency\_screen.dart

│ └── transfer\_screen.dart

├── widgets/ # Widgets reutilizáveis

│ ├── custom\_button.dart

│ ├── custom\_input.dart

│ └── transaction\_card.dart

├── services/ # Serviços

│ ├── auth\_service.dart

│ ├── currency\_service.dart

│ └── transaction\_service.dart

└── utils/ # Funções utilitárias

├── validators.dart

└── formatters.dart

**Telas Principais**

**1. Tela de Login**

A tela de login permite que os usuários acessem o aplicativo usando credenciais (email/senha).

**Funcionalidades:**

* Campo de entrada para email
* Campo de entrada para senha (com visualização opcional da senha)
* Botão de login
* Opção para autenticação biométrica (impressão digital/facial)

**2. Tela Principal (Home)**

A tela principal exibe informações sobre a conta do usuário após o login bem-sucedido.

**Funcionalidades:**

* Saldo da conta
* Lista de transações recentes
* Menu de acesso rápido para:
  + Transferência
  + Cotação
  + Perfil do usuário
  + Outras operações

**3. Tela de Cotação**

Esta tela permite aos usuários visualizarem as cotações atualizadas de diversas moedas.

**Funcionalidades:**

* Lista de moedas com valores atualizados
* Gráfico de histórico de cotações
* Campo de busca para encontrar moedas específicas
* Opção para atualizar cotações em tempo real

**Integração com API:**

* Consumo da API de cotações para obter valores atualizados
* Tratamento de respostas e erros da API
* Cache local para reduzir o número de requisições

**4. Tela de Transferência**

Esta tela permite aos usuários realizarem transferências de dinheiro para outros usuários ou contas.

**Funcionalidades:**

* Campo para seleção do destinatário
* Campo para valor da transferência
* Campo opcional para descrição/motivo
* Confirmação da transferência com resumo
* Comprovante digital com opção de compartilhamento

**Plugins Utilizados**

1. **flutter\_secure\_storage**: Armazenamento seguro de tokens e informações sensíveis
2. **local\_auth**: Autenticação biométrica
3. **http**: Requisições HTTP para API de cotações
4. **share\_plus**: Compartilhamento de comprovantes
5. **camera**: Acesso à câmera para escaneamento de QR Code
6. **fl\_chart**: Visualização de gráficos para cotações
7. **image\_picker**: Seleção de imagens para perfil do usuário

**Gerenciamento de Estado**

O aplicativo utiliza uma combinação de:

* **Provider**: Para gerenciamento de estado global
* **ValueNotifier**: Para estados locais em widgets específicos

**Rotas Nomeadas**

O sistema de navegação é baseado em rotas nomeadas para facilitar a transição entre telas:

// Exemplo de configuração de rotas

final Map<String, WidgetBuilder> routes = {

'/': (context) => LoginScreen(),

'/home': (context) => HomeScreen(),

'/currency': (context) => CurrencyScreen(),

'/transfer': (context) => TransferScreen(),

};

// Exemplo de rota com argumentos

'/transfer\_confirm': (context) => TransferConfirmScreen(

args: ModalRoute.of(context)!.settings.arguments as TransferArgs,

),

**Integração com API de Cotações**

A API de cotações é acessada através do serviço CurrencyService, que realiza requisições HTTP para obter dados atualizados sobre as moedas.

// Exemplo simplificado de serviço de cotação

class CurrencyService {

final String apiUrl = 'https://api.exemplo.com/cotacoes';

Future<List<Currency>> getCurrencies() async {

final response = await http.get(Uri.parse(apiUrl));

if (response.statusCode == 200) {

final List<dynamic> data = json.decode(response.body);

return data.map((item) => Currency.fromJson(item)).toList();

} else {

throw Exception('Falha ao carregar cotações');

}

}

}

**Segurança**

O aplicativo implementa as seguintes medidas de segurança:

* Autenticação segura com validação de campos
* Armazenamento criptografado de dados sensíveis
* Timeout de sessão para proteção contra acesso não autorizado
* Opção de autenticação biométrica para maior segurança

**Responsividade**

O layout do aplicativo é adaptável a diferentes tamanhos de tela, utilizando:

* Widgets flexíveis (Expanded, Flexible)
* MediaQuery para adaptação baseada no tamanho da tela
* LayoutBuilder para layouts condicionais

**Geração de APK Otimizado**

Para a geração do APK otimizado, são utilizadas as seguintes técnicas:

* R8 para minificação de código
* Compressão de recursos
* Remoção de código não utilizado
* Suporte a diferentes arquiteturas (arm64-v8a, armeabi-v7a)

**Melhorias Futuras**

* Implementação de temas claro/escuro
* Suporte para múltiplos idiomas
* Sincronização offline
* Notificações push para transações
* Integração com Open Banking APIs