### **Estrutura:**

### Foi utilizado MVVM com Clean Architecture que traz uma série de benefícios ao desenvolvimento de software, especialmente em termos de manutenção, escalabilidade e testabilidade. Essa abordagem modular e bem definida facilita a colaboração em equipe, a integração de novas funcionalidades e a garantia de qualidade do código. **Algumas Libs utilizadas no projeto:** Dagger Hilt, Coroutines e Retrofit além de algumas referente a testes que falo no final do arquivo.

### **Documentação técnica:**

### **Data Transfer Objects (DTOs)**

#### **ExchangeDetailsDto**

1. **Descrição**: Representa os detalhes de uma exchange retornados pela API.
2. **Campos**:
   1. exchangeId: Identificador da exchange.
   2. name: Nome da exchange.
   3. rank: Rank da exchange.
   4. volume1dayUsd: Volume de negociação em um dia em USD.
   5. volume1hrsUsd: Volume de negociação em uma hora em USD.
   6. volume1mthUsd: Volume de negociação em um mês em USD.
   7. website: URL do site da exchange.

#### **ExchangeDto**

1. **Descrição**: Representa uma exchange retornada pela API.
2. **Campos**:
   1. volume1dayUsd: Volume de negociação em um dia em USD.
   2. exchangeId: Identificador da exchange.
   3. name: Nome da exchange.

#### **ExchangeIconDto**

1. **Descrição**: Representa o ícone de uma exchange retornado pela API.
2. **Campos**:
   1. assetId: Identificador do ativo.
   2. exchangeId: Identificador da exchange.
   3. url: URL do ícone.

### **Mappers**

#### **Mapper Functions**

1. **Descrição**: Funções de extensão para converter DTOs em modelos de domínio.
2. **Funções**:
   1. ExchangeDetailsDto.toExchangeDetails(): Converte ExchangeDetailsDto para ExchangeDetails.
   2. ExchangeIconDto.toExchangeIcon(): Converte ExchangeIconDto para ExchangeIcon.
   3. ExchangeDto.toExchange(): Converte ExchangeDto para Exchange.

### **Remote**

#### **ApiKeyInterceptor**

1. **Descrição**: Interceptor que adiciona a API key ao header das requisições.
2. **Métodos**:
   1. intercept(chain: Interceptor.Chain): Adiciona a chave da API no header da requisição.

#### **ExchangeMarketBitcoinApi**

1. **Descrição**: Interface que define os endpoints da API.
2. **Métodos**:
   1. getExchanges(): Obtém a lista de exchanges.
   2. getExchangeById(exchangeId: String): Obtém os detalhes de uma exchange pelo ID.
   3. getExchangeIcon(iconSize: Int): Obtém os ícones das exchanges.

### **Repository**

#### **CoinRepositoryImpl**

1. **Descrição**: Implementação do repositório que interage com a API.
2. **Métodos**:
   1. getExchanges(): Obtém a lista de exchanges.
   2. getExchangeById(exchangeId: String): Obtém os detalhes de uma exchange pelo ID.
   3. getExchangeIcon(iconSize: Int): Obtém os ícones das exchanges.

### **Dependency Injection**

#### **AppModule**

1. **Descrição**: Módulo de configuração do Dagger Hilt.
2. **Métodos**:
   1. provideOkHttpClient(): Configura e fornece um OkHttpClient.
   2. provideCoinMarketBitcoinApi(): Configura e fornece a API.
   3. provideCoinRepository(): Fornece a implementação do repositório.

### **Domain Models**

#### **Exchange**

1. **Descrição**: Modelo de domínio para uma exchange.
2. **Campos**:
   1. volume1dayUsd: Volume de negociação em um dia em USD.
   2. exchangeId: Identificador da exchange.
   3. name: Nome da exchange.
   4. iconUrl: URL do ícone da exchange (opcional).

#### **ExchangeDetails**

1. **Descrição**: Modelo de domínio para os detalhes de uma exchange.
2. **Campos**:
   1. exchangeId: Identificador da exchange.
   2. name: Nome da exchange.
   3. rank: Rank da exchange.
   4. volume1dayUsd: Volume de negociação em um dia em USD.
   5. volume1hrsUsd: Volume de negociação em uma hora em USD.
   6. volume1mthUsd: Volume de negociação em um mês em USD.
   7. website: URL do site da exchange.

#### **ExchangeIcon**

1. **Descrição**: Modelo de domínio para o ícone de uma exchange.
2. **Campos**:
   1. assetId: Identificador do ativo.
   2. exchangeId: Identificador da exchange.
   3. url: URL do ícone.

### **Use Cases**

#### **GetExchangeUseCase**

1. **Descrição**: Caso de uso para obter detalhes de uma exchange.
2. **Métodos**:
   1. invoke(exchangeId: String): Retorna um fluxo de Resource<ExchangeDetails>.

#### **GetExchangeIconUseCase**

1. **Descrição**: Caso de uso para obter ícones das exchanges.
2. **Métodos**:
   1. invoke(iconSize: Int): Retorna um fluxo de Resource<List<ExchangeIcon>>.

#### **GetExchangeListWithIconUseCase**

1. **Descrição**: Caso de uso para obter a lista de exchanges com ícones.
2. **Métodos**:
   1. invoke(): Retorna um fluxo de Resource<List<Exchange>>.

#### **GetExchangesUseCase**

1. **Descrição**: Caso de uso para obter a lista de exchanges.
2. **Métodos**:
   1. invoke(): Retorna um fluxo de Resource<List<Exchange>>.

### **Presentation**

#### **ExchangeDetailScreen**

1. **Descrição**: Tela de detalhes da exchange.
2. **Componentes**:
   1. ExchangeDetailsFooter: Componente que exibe o rodapé com informações adicionais.
   2. VolumeField: Componente que exibe campos de volume.

#### **ExchangeDetailViewModel**

1. **Descrição**: ViewModel para a tela de detalhes da exchange.
2. **Métodos**:
   1. getExchange(exchangeId: String): Obtém os detalhes da exchange.

#### **ExchangeListScreen**

1. **Descrição**: Tela de listagem das exchanges.
2. **Componentes**:
   1. ExchangeListItem: Componente que exibe um item da lista de exchanges.

#### **ExchangeListViewModel**

1. **Descrição**: ViewModel para a tela de listagem das exchanges.
2. **Métodos**:
   1. getExchangesWithIcons(): Obtém a lista de exchanges com ícones.

### **Shared Components**

#### **ExchangeStateError**

1. **Descrição**: Componente que exibe uma mensagem de erro.

#### **ExchangeStateLoading**

1. **Descrição**: Componente que exibe um indicador de carregamento.

### **Utils**

#### **Constants**

1. **Descrição**: Constantes usadas na aplicação.

#### **Resource**

1. **Descrição**: Classe selada para representar o estado dos dados (sucesso, erro, carregando).

#### **Extensions**

1. **Descrição**: Funções de extensão para manipulação de erros e conversão de DTOs para modelos de domínio.

### **Application**

#### **ExchangeApplication**

1. **Descrição**: Classe Application configurada com Dagger Hilt.

### **Testes Unitários e de UI**

### Para garantir a funcionalidade e a qualidade do código, foram realizados testes unitários e de UI utilizando bibliotecas como MockK, Mockito e Jetpack Compose Testing etc.