DOCUMENTO DE ARQUITETURA

Equipe:

508161 – João Paulo Freitas Queiroz

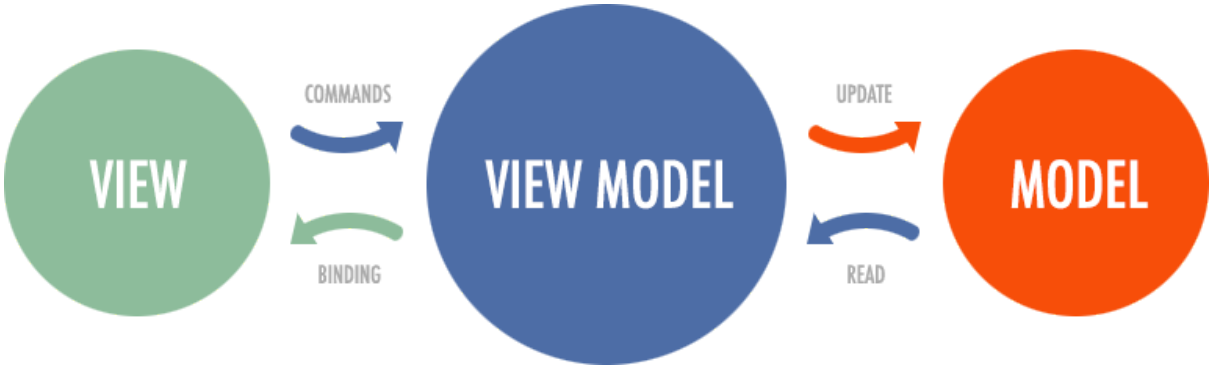
511790 – Matheus Leandro de Melo Silva

473143 - João Lucas de Oliveira Lima

537777 - Eduardo Henrique Brito da Silva

MVVM (Model-View-ViewModel)

MVVM, (Model-View-ViewModel), é um padrão arquitetural de software amplamente utilizado no desenvolvimento de aplicativos de interface de usuário (UI), especialmente em aplicativos para dispositivos móveis. Foi projetado para melhorar a separação da lógica da aplicação da interface do usuário.



O fluxo de dados se dá da seguinte forma:

1. Interação do usuário com a View (por exemplo, um clique de botão)
2. A visualização dispara um evento e o passa para o ViewModel
3. O ViewModel processa esse evento e atualiza o Model de acordo.

Camadas:

Model:

O Model representa os dados e a lógica de negócios do aplicativo. Ele é responsável por armazenar e gerenciar os dados, bem como executar operações relacionadas aos dados. Isso pode incluir a recuperação de dados de um banco de dados, chamadas de API, cálculos e validações de dados.

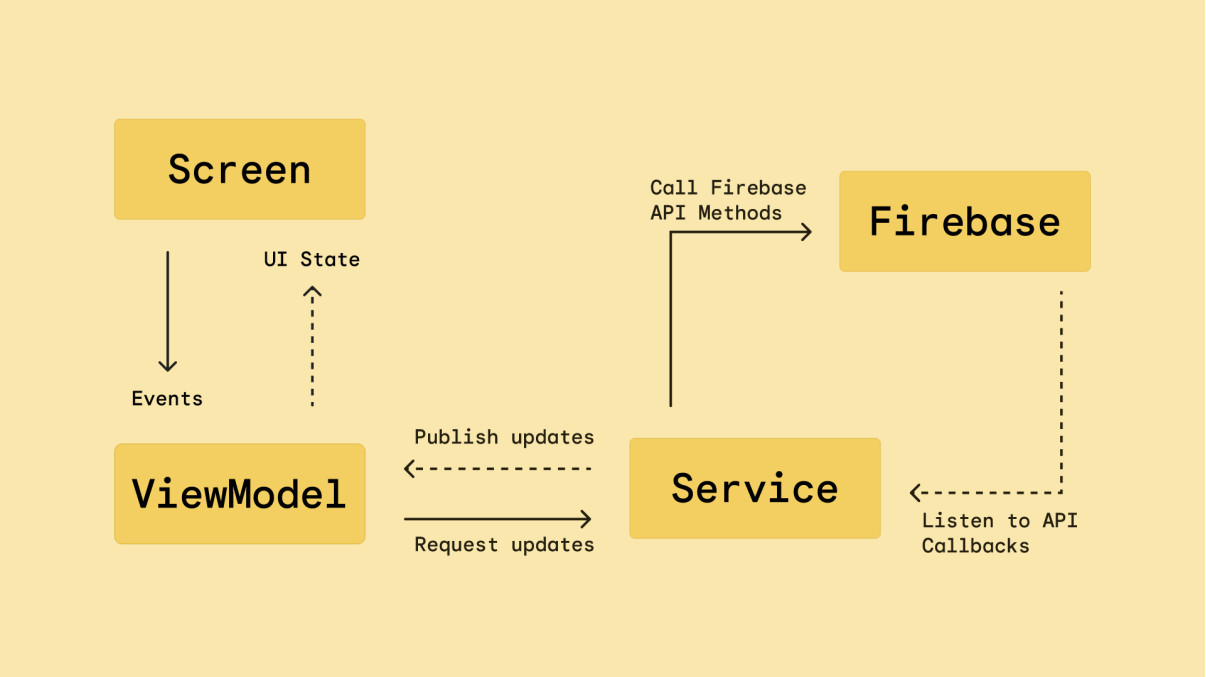
View:

A View é a camada responsável pela apresentação da interface de usuário. Ela exibe os dados do Model e interage com o usuário. A View geralmente consiste em elementos de interface, como botões, caixas de texto e gráficos. No MVVM, a View é o que o usuário vê e interage diretamente. No nosso projeto a camada de UI(View) é representada pela Screen que envia eventos para os ViewModels, além disso a Screen também precisa observar estados e atualizar os dados mostrados, tudo isso através do uso dos UiState.

ViewModel:

O ViewModel atua como um intermediário entre a Model e a View. Ele contém a lógica de apresentação e a lógica de interação com o usuário. O ViewModel é responsável por fornecer os dados necessários para a View e processar as ações do usuário. Ele é projetado para ser independente da View e, portanto, não deve conter código específico da interface do usuário. No nosso projeto a camada de ViewModel vai chamar métodos dos

Services gerenciando toda a lógica de negócio. O Services é responsável por chamar os métodos de diferentes APIs do Firebase que nós vamos usar e retornar o resultado para os ViewModels.



Arquitetura do compose

No Compose cada tela tem um arquivo Screen e um arquivo ViewModel. A maioria das telas também utiliza uma classe UiState para saber de que forma os dados são trocados entre o ViewModel e os Composables da tela. A camada de ViewModel contém toda a lógica de negócio que deve ser isolada dos composables. O Compose segue o *declarative UI model* no qual a screen observa o estado do objeto e quando percebe que o estado dele mudou o framework automaticamente reexecuta as funções de composable, processo esse chamado de “recomposição”.

Vantagens:

Separação das regras de negócio:

O MVVM ajuda a separar as regras de negócio, mantendo o código da UI separado da lógica da aplicação.

Fácil de manter e testar:

O MVVM facilita a utilização de testes unitários permitindo testar cada componente separadamente. Pode-se utilizar o design pattern de injeção de dependência para injetar objetos simulados em seu código, para que não precise depender de dependências externas, como rede ou banco de dados, durante o teste.

Reutilização de Código:

Como o ViewModel é independente da View, ele pode ser reutilizado em várias partes da interface do usuário. Isso reduz a duplicação de código e melhora a consistência em todo o aplicativo..

Colaboração Mais Fácil:

O MVVM é um padrão de arquitetura bem definido e amplamente aceito, o que facilita a colaboração entre desenvolvedores, designers e equipes multidisciplinares. Cada membro da equipe pode se concentrar em sua área de especialização sem interferir nas outras.

Ligação de Dados (Data Binding):

Bibliotecas que automatizam a sincronização entre a View e o ViewModel, reduzindo a necessidade de código boilerplate para atualizar a interface do usuário quando os dados mudam.

Conclusão:

O MVVM não apenas ajuda a criar aplicativos Android mais organizados e testáveis, mas também promove a colaboração eficaz entre equipes de desenvolvedores e designers. Além disso, melhora a manutenção de código legado e é uma escolha sólida para projetos de médio a grande porte, onde a escalabilidade e a qualidade do código são cruciais. Portanto, ao considerar todos esses benefícios, o MVVM é uma escolha inteligente para o desenvolvimento de sistemas Android.