Padrões de Projeto Mobile: MVP, MVVM, MVC

1. Modelo MVC (Model-View-Controller)

O **MVC** é um dos padrões de projeto mais antigos e amplamente utilizados. Ele separa a aplicação em três componentes principais:

- **Model (Modelo):** Representa os dados e a lógica de negócios. Ele não depende da interface do usuário e contém as regras e operações que manipulam os dados.
- **View (Visão):** Responsável pela apresentação da interface gráfica para o usuário. A View apenas exibe a informação, sem ter lógica de negócios.
- **Controller (Controlador):** Atua como intermediário entre o Model e a View. Recebe as entradas do usuário através da View, processa essas entradas (geralmente interagindo com o Model), e atualiza a View de acordo.

Vantagens:

- Separação clara das responsabilidades.
- Facilita a manutenção e a evolução do código.
- Permite reutilização de componentes.

Desvantagens:

- A View e o Controller podem ficar fortemente acoplados.
- Pode se tornar difícil de gerenciar em aplicativos maiores.

2. Modelo MVP (Model-View-Presenter)

O **MVP** é uma evolução do MVC, com o objetivo de resolver alguns dos problemas do MVC em aplicações mais complexas. Ele também é dividido em três componentes:

- **Model (Modelo):** Similar ao MVC, trata dos dados e da lógica de negócios.
- **View (Visão):** Também exibe a interface para o usuário, mas aqui, a View é mais passiva, apenas respondendo a comandos do Presenter.
- **Presenter (Apresentador):** Atua como intermediário, mas aqui ele tem mais responsabilidade. O Presenter contém toda a lógica para manipular a View e interagir com o Model. Ele atualiza a View com os dados do Model.

Vantagens:

- Maior desacoplamento entre a View e o Presenter, facilitando a realização de testes.
- A lógica da interface é separada do código de exibição, facilitando a manutenção.

Desvantagens:

- O Presenter pode se tornar muito grande e complexo.

3. Modelo MVVM (Model-View-ViewModel)

O **MVVM** é amplamente utilizado em frameworks como o Angular e tecnologias como o Xamarin. Ele é uma extensão do MVP com algumas diferenças sutis e importantes:

- **Model (Modelo):** Gerencia os dados e a lógica de negócios, como nos outros padrões.
- **View (Visão):** A interface gráfica do usuário, que observa mudanças nos dados da ViewModel.
- **ViewModel:** Interage com o Model e expõe os dados para a View. A grande diferença é que a ViewModel possui dados observáveis e comandos que a View pode escutar ou executar.

Vantagens:

- A ViewModel permite a reutilização do código, pois pode ser usada com diferentes Views.
- Suporta data binding, o que simplifica a interação entre a View e a ViewModel.
- Facilita o teste unitário, pois a ViewModel não depende da View.

Desvantagens:

- Pode ser mais complicado de implementar e entender.

Referencias

https://cwi.com.br/blog/design-patterns-android/