Os padrões de projeto **MVP (Model-View-Presenter)**, **MVVM (Model-View-ViewModel)** e **MVC (Model-View-Controller)** são abordagens de arquiteturas comuns no desenvolvimento de aplicativos mobile. Cada um desses padrões separa a lógica de negócios, a interface do usuário e a lógica de controle em diferentes componentes para facilitar a manutenção e o escalonamento do software.

1. MVC (Model-View-Controller)

- **Model**: Representa os dados e a lógica de negócios da aplicação. É responsável por acessar e manipular os dados, seja por meio de bancos de dados, APIs, etc.
- **View**: É a interface do usuário. A View é responsável por exibir os dados do Model ao usuário e encaminhar as interações do usuário para o Controller.
- **Controller**: Atua como intermediário entre o Model e a View. Recebe as entradas do usuário da View, processa essas entradas (geralmente invocando métodos no Model), e atualiza a View conforme necessário.

Uso Comum: MVC é amplamente utilizado em frameworks como o iOS UIKit e no desenvolvimento web (por exemplo, em frameworks como Rails, Django). Ele permite uma clara separação de preocupações, mas pode levar a uma sobrecarga no Controller em aplicações mais complexas.

2. MVP (Model-View-Presenter)

- **Model**: Similar ao Model no MVC, ele gerencia os dados e a lógica de negócios.
- **View**: Exibe os dados e captura as interações do usuário, mas é mais "burra" do que no MVC, dependendo do Presenter para a lógica de atualização.
- **Presenter**: Recebe entradas da View e se comunica com o Model para obter ou manipular os dados. O Presenter também atualiza a View com os dados processados. Diferente do Controller no MVC, o Presenter possui uma referência direta à View, facilitando a interação.

Uso Comum: MVP é popular no desenvolvimento Android, especialmente em versões anteriores à introdução de frameworks como o Jetpack. Ele oferece uma separação mais clara entre a lógica da interface do usuário e a lógica de apresentação, tornando a View mais fácil de testar.

3. MVVM (Model-View-ViewModel)

- **Model**: Igual aos outros padrões, ele gerencia os dados e a lógica de negócios.
- **View**: Exibe dados e é responsável pela interação com o usuário. No MVVM, a View é mais desacoplada da lógica de apresentação do que no MVP.
- **ViewModel**: Atua como um intermediário entre o Model e a View, expondo métodos e propriedades que a View pode vincular diretamente. A ViewModel não tem conhecimento direto da View e vice-versa, permitindo que a interface do usuário seja alterada sem afetar a lógica da aplicação.

Uso Comum: MVVM é muito popular no desenvolvimento de aplicativos com frameworks como o Android Jetpack (especialmente com Data Binding e LiveData) e o SwiftUI no iOS. Ele facilita o desenvolvimento de interfaces reativas e escaláveis, separando as responsabilidades de maneira que a lógica de negócios e a interface do usuário sejam altamente desacopladas.

Nome: Rodrigo Teixeira Machado de Oliveira