**Design Digital – Pesquisa**

**Quais são os padrões mais utilizados na arquitetura mobile?**

Todos eles se diferenciam de acordo com as suas camadas específicas e o modo como elas interagem.

A arquitetura de software aplicada ao ambiente mobile tem como principais benefícios a facilidade de uso e a praticidade na hora de implementar os códigos, possibilitando uma distribuição mais inteligente das responsabilidades na hora de desenvolver a ferramenta. Vamos conhecer outros benefícios.

### MVC (*Model-View-Controller*)

MVC (Model-View-Controller) é um padrão de projeto de software que separa a interface do usuário (View) e das regras de negócio e dados (Model) usando um mediador (Controller) para conectar o modelo à view.

### MVP (*Model-View-Presenter*)

O padrão MVP tem a finalidade de separar a camada de apresentação das camadas de dados e regras de negócio.

**View** - em nosso caso é o formulário que exibirá os dados, não contém regra alguma do negócio a não ser disparar eventos que notificam mudança de estado dos dados que ele exibe e processamento próprio dele, como por exemplo código para fechar o formulário. Um objeto view implementa uma interface que expõe campos e eventos que o presenter necessita.

**Model** - São os objetos que serão manipulados. Um objeto Model implementa uma interface que expõe os campos que o presenter irá atualizar quando sofrerem alteração na view.

**Presenter** - É a ligação entre View e Model, possui papel de mediador entre eles. Ele é encarregado de atualizar o view quando o model é alterado e de sincronizar o model em relação ao view.

### MVP vs. MVC

Na prática, não há muitas diferenças entre as duas propostas de apresentação. Ambos se concentram em separar as responsabilidades entre as camadas e incentivar o desacoplamento da UI (View) com a camada de negócios (Model). As principais diferenças entre eles são:

**Padrão MVC:** o Controller é baseado em comportamentos e podem ser compartilhados entre múltiplas Views, tendo menor burocracia e rápido reaproveitamento.

**Padrão MVP:** pela grande separação entre a View e o Presenter (graças a interface), garante testes mais fáceis. As interfaces são criadas com relacionamento de 1 para 1, ou seja: para cada View existirá um Presenter.

### MVVM (*Model-View-ViewModel*)

O padrão de projeto Model-View-ViewModel (MVVM) foi originalmente criado para aplicativos Windows Presentation Foundation (WPF) usando XAML para separar a interface do usuário (UI) da lógica de negócios e aproveitando ao máximo o data binding (*a vinculação de dados*).  
É composto basicamente dos seguintes elementos:  
**Model:** A camada de modelo é a lógica de negócios que impulsiona a aplicação e quaisquer objetos de negócios;

**View:** Esta camada é a interface do usuário. No caso do desenvolvimento cross plataform, ela inclui qualquer código específico da plataforma para conduzir a interface do usuário da aplicação.

**ViewModel**: Esta camada age como a cola em aplicações MVVM. As camadas ViewModel coordenam as operações entre a view e as camadas model. Uma camada ViewModel irá conter propriedades que a View vai obter ou definir, e funções para cada operação que pode ser feita pelo usuário em cada view. A camada ViewModel também evocará operações sobre a camada **Model**, se necessário.

### VIPER (*View-Interactor-Presenter-Entity-Router*)

O VIPER começou sendo utilizado principalmente nos sistemas iOS, mas hoje também é uma opção recorrente dos desenvolvedores Android. Esse padrão se destina a manter o código organizado e dividido em módulos. Seus componentes são:

* *view* — transfere a entrada do usuário para a camada presenter;
* *interactor* — mantém a lógica de negócio;
* *presenter* — mantém a lógica da view e prepara o conteúdo para exibição;
* *entity* — contém objetos de modelo que são usados pelo *interactor*;
* *router* — responsável pela lógica de navegação.

**Site com uma explicação detalhada e fotos de cada modelo**:( [explicando as arquiteturas](https://medium.com/android-dev-br/arquitetura-limpa-nas-apps-utilizando-viper-no-android-f39e51b44723) )