MVC (Model-View-Controller)

Model: Representa os dados e a lógica de negócios da aplicação. Ele gerencia o estado da aplicação e notifica a View sobre alterações.

View: É a interface do usuário, responsável por exibir os dados ao usuário e enviar eventos para o Controller.

Controller: Atua como intermediário entre o Model e a View. Ele processa a entrada do usuário, atualiza o Model e altera a View conforme necessário.

Vantagens:

Separação clara de responsabilidades.

Facilita a manutenção e escalabilidade.

Desvantagens:

Pode levar a Controllers muito grandes e complexos.

MVP (Model-View-Presenter)

Model: Gerencia os dados e a lógica de negócios, semelhante ao MVC.

View: Exibe dados e envia eventos para o Presenter. A View é mais passiva e geralmente não contém lógica de negócios.

Presenter: Interage com o Model e atualiza a View. Ele processa a entrada do usuário e decide como a View deve ser atualizada.

Vantagens:

Melhora a testabilidade, pois a lógica está separada da View.

A View é mais simples e desacoplada da lógica.

Desvantagens:

Pode resultar em uma maior quantidade de código de boilerplate.

Maior complexidade na comunicação entre Presenter e View.

MVVM (Model-View-ViewModel)

Model: Representa os dados e a lógica de negócios.

View: A interface do usuário que exibe os dados e pode ser ligada ao ViewModel.

ViewModel: Abstrai a View e fornece dados e comandos. Ele faz a ligação entre o Model e a View e contém a lógica de apresentação.

Vantagens:

Facilita o binding de dados, reduzindo o código necessário para atualizar a interface.

Melhora a testabilidade e separação de preocupações.

Desvantagens:

Pode ser mais complexo de implementar devido ao binding de dados e comandos.

Requer uma compreensão mais profunda do framework ou biblioteca de binding usada.