Dispositivo Móveis

PROFESSOR CRISPIM LUIZ MARTINS

Porque usar arquitetura MVVM. Lembrando que alguns assuntos!

- ***ACOPLAMENTO*** em programação se refere ao grau de interdependência entre módulos ou componentes de um software. *Aqui estão alguns pontos importantes sobre o acoplamento:*
- Dependência entre Módulos: Acoplamento indica quanto um módulo depende de outro para funcionar.
- Baixo Acoplamento: Códigos desacoplados são aqueles de relação fraca, e não dependem de outros para fazer sua funcionalidade básica. Quando há um baixo acoplamento, um componente consegue operar de forma praticamente independente do outro.
- Alto Acoplamento: Alto acoplamento implica em uma conexão forte entre os componentes, que pode torná-los até indistinguíveis. O forte acoplamento pode trazer muitos problemas, como dificuldades de manutenção e gerenciamento, pois qualquer mudança vai afetar toda a cadeia de classes.

Sobre acoplamento, o que é desejável.

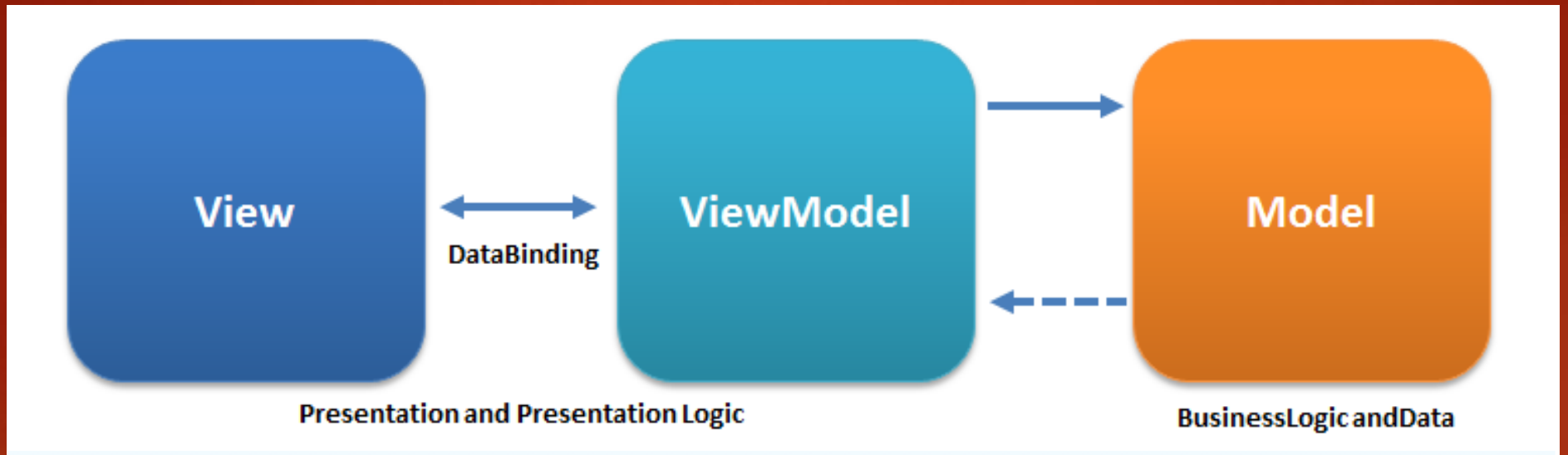
- É sempre desejável um baixo nível de acoplamento, pois isso geralmente se correlaciona com alta coesão e vice-versa. O baixo acoplamento é frequentemente um sinal de um sistema de computador bem estruturado e de um bom design, e quando combinado com alta coesão, suporta os objetivos gerais de alta legibilidade e facilidade de manutenção.

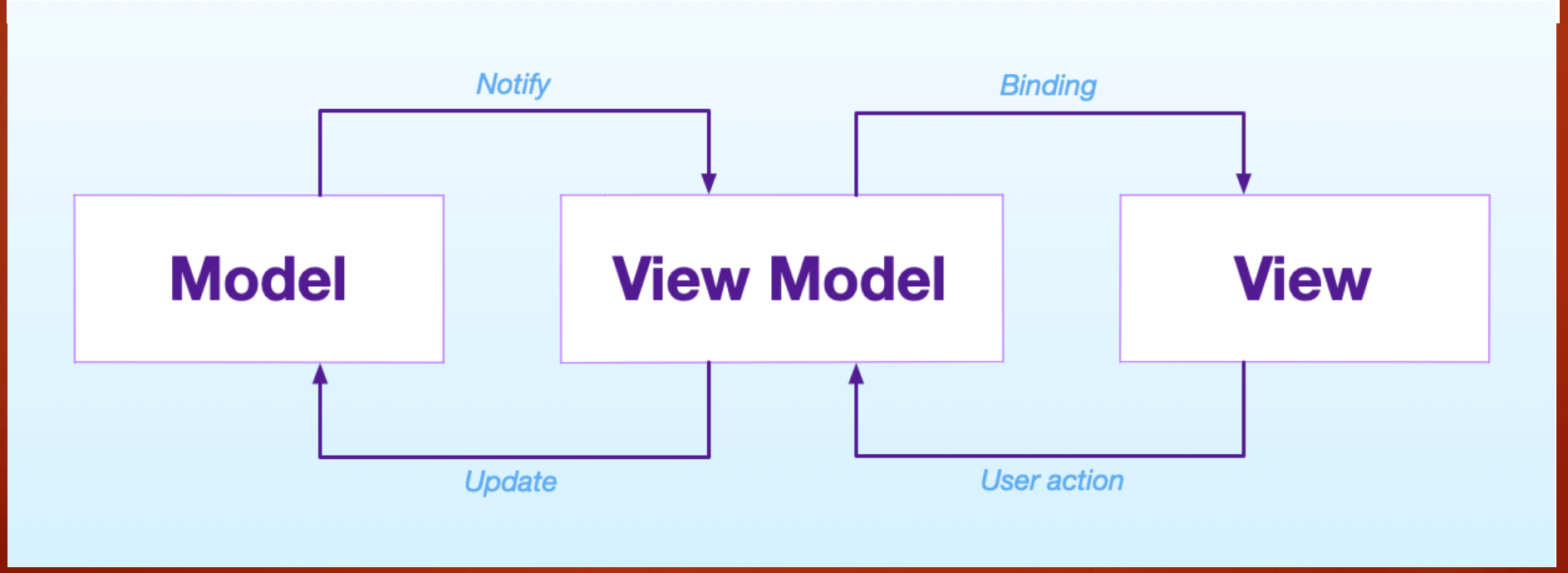
Interface do Usuário (IU)

- Os componentes da Interface do Usuário (IU) são os elementos que ajudam as pessoas a interagir com um software, site ou aplicativo. Eles geralmente se enquadram em uma das quatro categorias
- CONTROLES DE ENTRADA: Esses elementos de IU interativos permitem que os usuários insiram os dados no sistema. Você pode usar uma ampla variedade de controles de entrada, como botões, campos de texto, alternadores, caixas de seleção e muito mais.
- COMPONENTES DE NAVEGAÇÃO: Elementos como controle deslizante, breadcrumbs, campo de pesquisa, paginação, tags e ícones ajudam as pessoas a navegar por um site ou produto.

- COMPONENTES INFORMATIVOS: Os elementos da UI que ajudam a compartilhar informações com os usuários incluem, mas não se limitam a: notificações, barras de progresso, dicas de ferramentas, caixas de mensagens e janelas pop-up.
- CONTAINERS: Eles mantêm os elementos da página juntos.
- A IU se refere aos pontos de interação entre o usuário e um dispositivo ou software, como botões, ícones, janelas, etc.

Imagem do que estaremos falando.





Arquitetura MVVM

- A arquitetura MVVM (Model-View-ViewModel) é altamente recomendada pela equipe de desenvolvedores do Google e do Android. Ela possui várias vantagens, como tornar o projeto vagamente acoplado, facilitar a manutenção, simplificar a adição de novos recursos ou a remoção de recursos existentes, e tornar o código muito testável.
- Aqui está uma breve descrição dos componentes da arquitetura MVVM:

- View: Esta parte da arquitetura ajuda a construir a interface de usuário e é a única parte com a qual os usuários podem interagir diretamente.
- ViewModel: O objeto ViewModel atua como um intermediário entre View e Model, o que significa que fornece dados para os componentes da IU, como fragmentos ou atividades. Próximo Slide falaremos mais!!!
- Model: Representa os dados e a lógica de negócios da aplicação.
- Além disso, existem alguns frameworks que podem ajudar a implementar o MVVM. O Google fornece sua estrutura composta por vários componentes diferentes que você pode usar e construir para tornar seu trabalho mais simples.

ViewModel – Mais um pouco!

- O ViewModel é uma classe que atua como um intermediário entre a View e o Model. Ele não tem nenhuma ligação direta com a View, ou seja, não possui nenhuma referência a ela. Sua principal função é fornecer os dados e executar as ações que a View necessita.
- Aqui estão algumas características importantes do ViewModel:
- Persistência de Dados: O ViewModel armazena o estado em cache e mantém essas informações mesmo após mudanças de configuração. Isso significa que a interface não precisa buscar dados novamente ao navegar entre diferentes atividades ou implementar mudanças de configuração, como ao girar a tela.
- Acesso à Lógica de Negócios: O ViewModel encapsula a lógica de negócios correspondente e expõe o estado à interface.
- Desacoplamento: O ViewModel permite que você crie um modelo de acordo com o que você precisar na sua view, permitindo que você não tenha acoplamento com o seu domínio

- Um exemplo usando uma model e uma ViewModel, está em C#.
- Model

```
public class Estudante
{
    public int Id { get; set; }
    public string Nome { get; set; }
    public string Curso { get; set; }
}
```

O Model é uma representação dos dados que o aplicativo irá lidar.

- ViewModel

```
public class EstudanteViewModel
{
    public string Nome { get; set; }
    public string Curso { get; set; }
}
```

O ViewModel é uma classe que contém os dados que queremos exibir na nossa View. Ele pode conter dados de um ou mais Models.