# Entendendo o Pattern Model View ViewModel MVVM

**Guilherme Bon e Luiza Schefer**

Inversão de controle, desacoplamento, injeção de dependência, são formas de manter um código limpo, fácil de dar manutenção e que seja organizado, nos fazendo assim entender e amadurecer no processo de desenvolvimento de software. E hoje falaremos de um pattern chamado MVVM (Model-View-ViewModel) e poderemos ver na prática alguns destes conceitos.

O MVVM é um pattern que foi criado em 2005, por John Gossman, um dos arquitetos do WPF e Silverlight na Microsoft. O MVVM assemelha-se em alguns aspectos o MVC (Model View Controller) e ao MVP (Model View Presenter), podemos até dizer que o MVVM é uma especialização do MVP adaptado para a arquitetura do WPF E Silverlight. Conceitualmente, o MVVM e o MVP são idênticos, o que os diferencia é que o MVVM é específico para a arquitetura do WPF e Silverlight e o MVP é independente de plataforma. O MVVM, visa estabelecer uma clara separação de responsabilidades em uma aplicação WPF e Silverlight, mantendo uma espécie de façade entre o Modelo de objetos (entenda classes de negócio, serviços externos e até mesmo acesso a banco de dados) e a View que é a interface, com a qual o usuário interage. Para entendermos melhor, como se dá esta separação e visualizar como os componentes interagem dentro deste cenário, observe a figura abaixo:

Responsabilidades e características

View – A responsabilidade da View é definir a aparência ou estrutura que o usuário vê na tela. O ideal é que o codebehind da view, contenha apenas a chamada ao método InitializeComponent dentro do construtor, ou em alguns casos, código que manipule os controles visuais, ou crie animações; algo que é mais difícil de fazer em XAML. A View se liga ao ViewModel, através da propriedade DataContext que é setada para a classe ViewModel correspondente à aquela View.

O MVVM permite a você ter uma visão, da clara separação da Interface com o usuário(View), sua lógica de apresentação (ViewModel) e os seus Dados(Model). E trabalhando desta forma, temos separação de responsabilidades, desacoplamento e conseguimos evoluir e manter melhor as nossas aplicações.

Mais informações em: https://www.devmedia.com.br/entendendo-o-pattern-model-view-viewmodel-mvvm/18411