

PESQUISA

Pablo Valentin

Temas

Test Driven Development (TDD)

Model View View-Model (MVVM)

Model View Presenter (MVP)

Single Page Application (SPA)

TDD

Test Driven Development

O que é?

Consiste no Desenvolvimento Orientado a Testes, onde para cada funcionalidade de um projeto é criado um teste, que se inicia antes da implementação da mesma e termina após ela.

Etapas

- Definição da funcionalidade analisada;
- Criação do teste antes da implementação (Teoria);
- Erro previsto (Red);
- Implementação da funcionalidade;
- Adaptação do teste para seu pleno funcionamento (Green);
- Revisão do código visando melhorias e padronização (Refactor).

Red (Falha)

Nesta etapa, o teste é aplicado sem a existência do código da funcionalidade. É previsto que ocorra um erro, que será corrigido após a implementação da mesma.

Green (Êxito)

Aqui, o teste anterior é adaptado, já existindo o código da funcionalidade, o que garante que ela será implementada com sucesso.

Refactor (Revisão)

Sendo a última etapa, aqui a funcionalidade já foi implementada com sucesso, e o código é revisado, visando padronizá-lo e adicionar melhorias, caso seja possível.

Por que utilizar?

- Correção facilitada de bugs, devido aos testes;
- Desenvolvimento e integração contínua, pois os problemas são identificados e resolvidos mais rapidamente;
- Código mais limpo;
- Etc.

Fontes de Pesquisa

<https://www.devmmedia.com.br/test-driven-development-tdd-simples-e-pratico/18533>

MVVM

Model View View-Model

O que é?

É um padrão de arquitetura de projeto de software, baseado nos paradigmas POO (Programação Orientada a Objetos) e POE (Programação Orientada a Eventos), onde uma coisa acontece após outra já ter acontecido.

Características

- Atualização da interface em tempo real;
- Separação do código em várias camadas, permitindo melhor gerenciamento do código;
- Hierarquia de camadas bem definida (Nem todas as camadas interagem entre si.);
- Etc.

Camadas

- **Model** (Lógica de negócios da aplicação.);
- **View** (Interface gráfica.);
- **View-Model** (Manipulação e padronização dos dados da Model e atualização da View.).

Model

Nesta camada são criados arquivos que definem os componentes (Classes) do projeto e suas características (Atributos, campos, métodos, etc.).

View

Aqui, é onde são desenvolvidos os arquivos da interface gráfica, ou seja, tudo o que o usuário (Cliente) irá visualizar.

View-Model

Camada intermediária entre a View e a Model, que é responsável por obter dados do banco de dados, através da Model, e fornecer, obter e manipular dados da View, de forma indireta (Data Binding.), garantindo que ela sempre fique atualizada em tempo real. Também é comum existirem gerenciadores de estado nesta camada, que interceptam eventos acionados pela interação do usuário com a View (O cálculo do valor de uma compra, por exemplo, com base na quantidade escolhida pelo cliente.).

Por que utilizar?

- Melhor gerenciamento de projetos de médio e grande porte;
- Código mais estruturado;
- Maior facilidade para analisar erros, devido a separação do código em camadas;
- Etc.

Exemplos da aplicação da View-Model (Eventos)

- Funções que aumentam e diminuem a quantidade de um produto, após o usuário pressionar botões;
- Cálculo em tempo real do valor total de uma compra, com base na interação do usuário;
- Etc.

Fontes de Pesquisa

<https://www.devmedia.com.br/entendendo-o-pattern-model-view-viewmodel-mvvm/18411>

<https://www.youtube.com/watch?v=B2pJWtSyVFA> <https://www.youtube.com/watch?v=A197N1n0qSI>

<https://pt.stackoverflow.com/questions/80601/o-que-%C3%A9-a-program%C3%A7%C3%A3o-orientada-a-eventos>

<https://medium.com/@laylaemanuele/o-que-%C3%A9-mvvm-quais-s%C3%A3o-os-pr%C3%B3s-e-contras-15c022f8cb99>

<https://chatgpt.com/> (Explique o padrão de projeto arquitetural MVVM.)

MVP

Model View Presenter

O que é?

É um padrão arquitetural de projeto, muito parecido com o MVVM, porém um pouco menos complexo.

Características

- A atualização não é em tempo real, ou seja, precisa ser especificada no código;
- Diferente do MVVM, onde a View-Model não tem acesso direto a View, no MVP, a Presenter tem;
- Etc.

Camadas

- **Model** (Lógica de negócios da aplicação.);
- **View** (Interface gráfica.);
- **Presenter** (Camada de transporte de dados entre a View e a Model.).

Model

Nesta camada são criados arquivos que definem os componentes (Classes) do projeto, suas características (Atributos, campos, métodos, etc.) e gerenciadores de eventos, que interceptam alterações na View e atualizam os dados da Model, caso seja necessário.

View

Aqui, é onde são desenvolvidos os arquivos da interface gráfica, ou seja, tudo o que o usuário (Cliente) irá visualizar.

Presenter

Camada intermediária entre a View e a Model, que é responsável por obter dados do banco de dados, através da Model, e fornecer, obter e manipular dados da View, de forma direta.

Por que utilizar?

Pelos mesmo motivos do **MVVM**. São eles:

- Melhor gerenciamento de projetos de médio e grande porte;
- Código mais estruturado;
- Maior facilidade para analisar erros, devido a separação do código em camadas;
- Etc.

Fontes de Pesquisa

<https://www.devmmedia.com.br/o-padrao-mvp-model-view-presenter/3043>

SPA

Single Page Application

O que é?

É um conceito/funcionalidade muito utilizado por frameworks modernos, como React, Vue, Angular, etc. Consiste em definir uma única página HTML como raíz da aplicação, que é atualizada dinamicamente, em tempo de execução, para exibir o que o usuário deseja visualizar, ou seja, não é necessário que o navegador precise se atualizar para renderizar um novo conteúdo.

Características

- Divisão do HTML em componentes, garantindo reaproveitamento;
- Uma única página raíz para toda a aplicação;
- Atualização da página raíz, mas não do navegador (Fluidez);
- Gerenciamento de alteração de estado da página por meio de recursos de frameworks, o que pode dificultar a atualização da página raíz (Maior complexidade.), mas com a possibilidade de maior desempenho;
- Etc.

Por que utilizar?

- Maior sensação de fluidez pelo usuário;
- Evita repetição desnecessária de código (Reaproveitamento);
- Etc.

Fontes de Pesquisa

<https://www.devmedia.com.br/ja-ouviu-falar-em-single-page-applications/39009>

<https://developer.mozilla.org/en-US/docs/Glossary/SPA>

OBRIGADO!