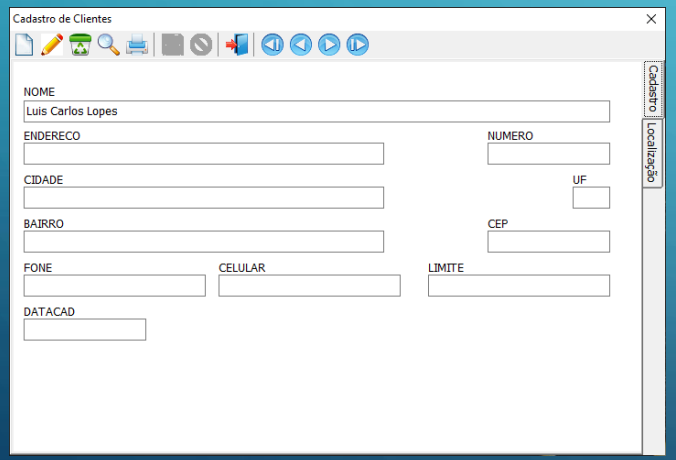
INTRODUÇÃO AO PADRÃO MVC

VEJA NESTE ARTIGO O ESTILO DE ARQUITETURA MVC (MODEL-VIEW-CONTROLLER), BASTANTE IMPORTANTE E MUITO UTILIZADA EM DIVERSOS FRAMEWORKS E PROJETOS DE SOFTWARE.

PADRÃO MVC

View



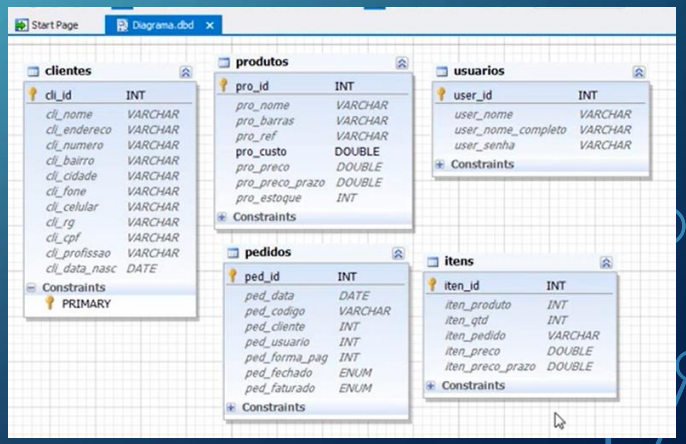
Cliente

Controlers

Procedures
Function
Métodos

Regras de Negócios

Model



Banco de Dados

HISTÓRIA DO MVC

- O Engenheiro Civil Christopher Alexander criou o que se considera o primeiro padrão de projeto em meados da década de 70. É considerado um padrão de projeto uma solução já testada e documentada que resolva um problema específico em projetos distintos. Através do trabalho de Alexander, profissionais da área de desenvolvimento de software utilizaram tais conceitos para iniciar as primeiras documentações de padrões de projetos, tornando-as acessíveis para toda a área de desenvolvimento.

PADRÕES DE PROJETO EM DESENVOLVIMENTO DE SOFTWARE:

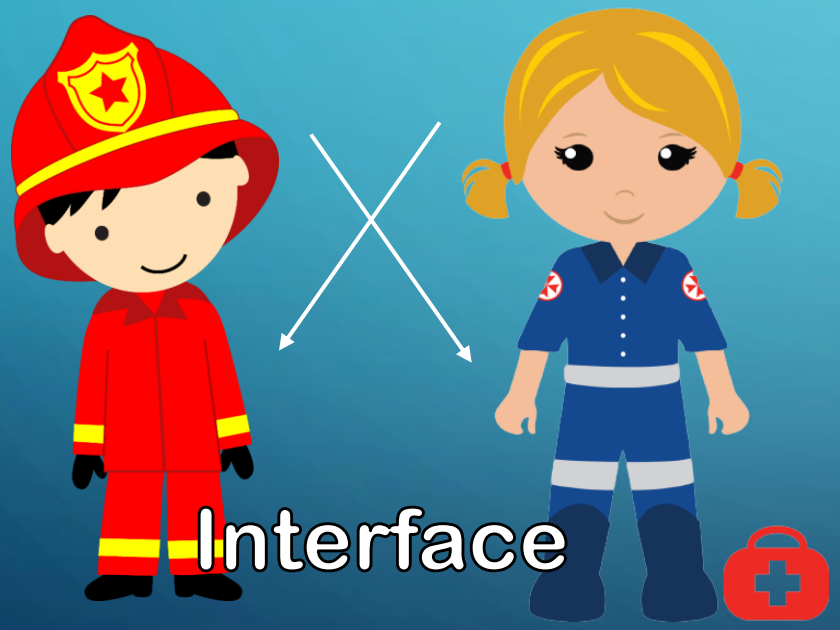
- Baixo acoplamento: é o grau em que uma classe conhece a outra. Se o conhecimento da classe A sobre a classe B for através de sua interface, temos um baixo acoplamento, e isso é bom. Por outro lado, se a classe A depende de membros da classe B que não fazem parte da interface de B, então temos um alto acoplamento, o que é ruim.
- Coesão: quando temos uma classe elaborada com um único e bem focado propósito, dizemos que ela tem alta coesão, e isso é bom. Quando temos uma classe com propósitos que não pertencem apenas a ela, temos uma baixa coesão, o que é ruim.

ALTO ACOPLAMENTO/BAIXO ACOPLAMENTO

Conhece

Classe A

Classe B



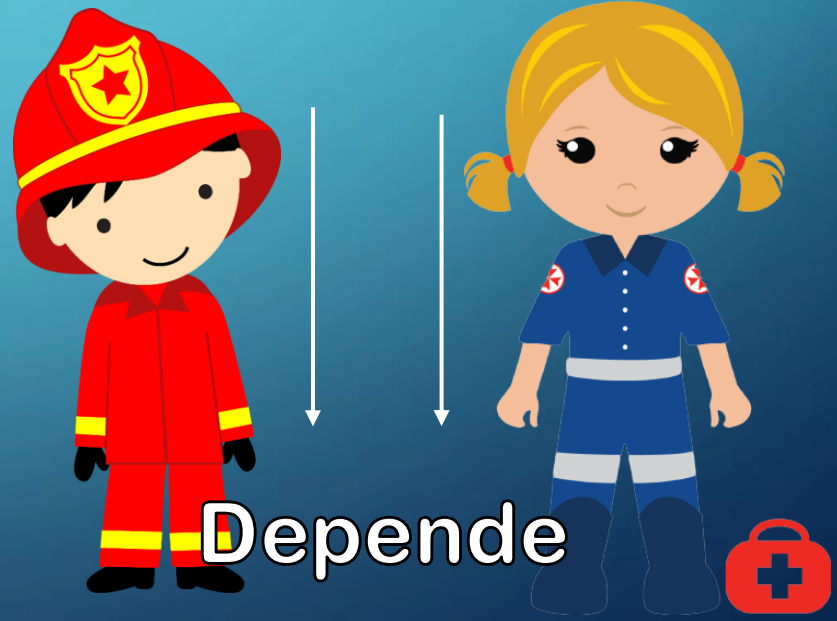
Interface

Alto acoplamento:

Não Conhece

Classe A

Classe B



Depende

Baixo ou nenhum acoplamento:

COESÃO

Classe A

único e bem focado propósito

Alta coesão

Classe B

classe com propósitos que não pertencem apenas a ela

Baixa coesão

PROJETO MVC PODE TRAZER ALGUNS DOS SEGUINTES BENEFÍCIOS:

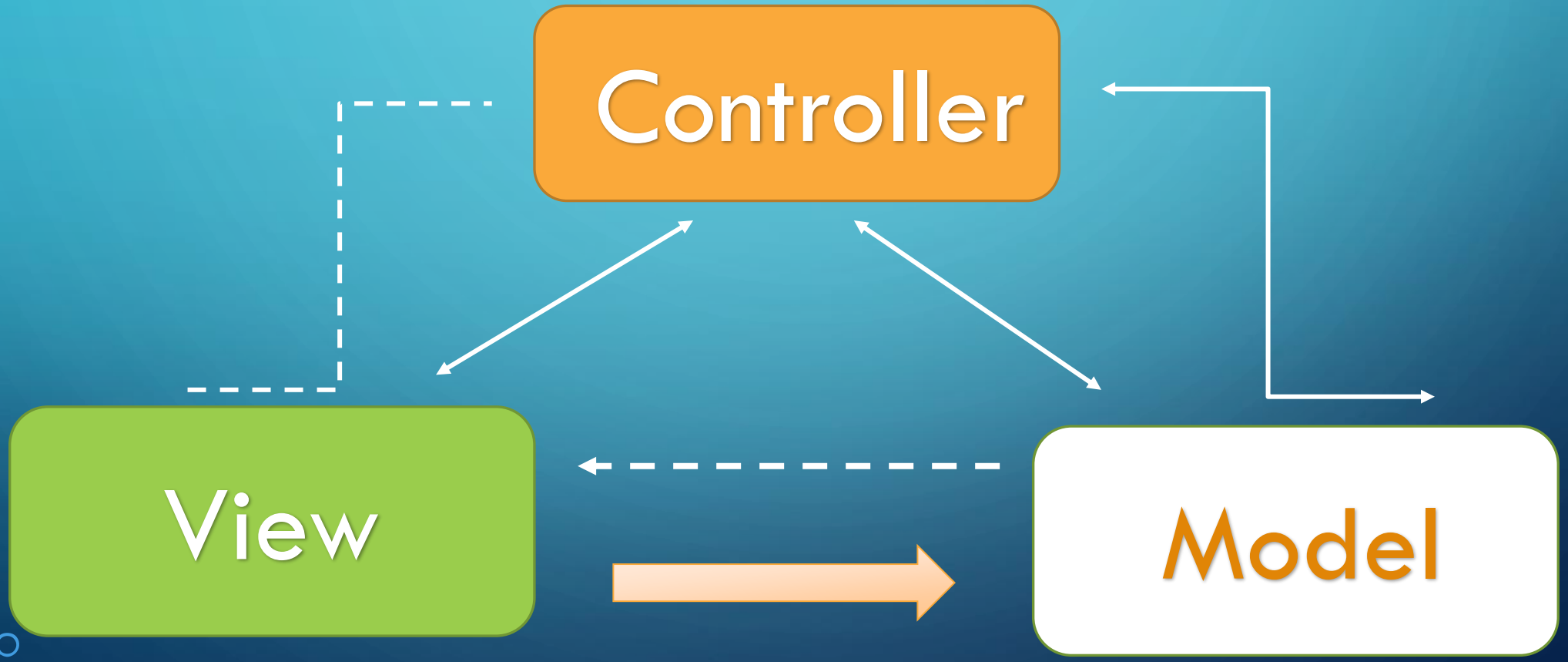
- Aumento de produtividade;
- Uniformidade na estrutura do software;
- Redução de complexidade no código;
- As aplicações ficam mais fácies de manter;
- Facilita a documentação;
- Estabelece um vocabulário comum de projeto entre desenvolvedores;
- Permite a reutilização de módulos do sistema em outros sistemas;
- É considerada uma boa prática utilizar um conjunto de padrões para resolver problemas maiores que, sozinhos, não conseguiriam;
- Ajuda a construir softwares confiáveis com arquiteturas testadas;
- Reduz o tempo de desenvolvimento de um projeto.

O PADRÃO MVC (MODEL-VIEW-CONTROLLER)

- A **utilização do padrão MVC** traz como benefício o isolamento das regras de negócios da lógica de apresentação, que é a interface com o usuário. Isto possibilita a existência de várias interfaces com o usuário que podem ser modificadas sem a necessidade de alterar as regras de negócios, proporcionando muito mais flexibilidade e oportunidades de reuso das classes.
- Uma das características de um padrão de projeto é poder aplicá-lo em sistemas distintos. **O padrão MVC pode ser utilizado em vários tipos de projetos** como, por exemplo, desktop, web e mobile.

- A comunicação entre interfaces e regras de negócios é definida através de um controlador, que separa as camadas. Quando um evento é executado na interface gráfica, como um clique em um botão, a interface se comunicará com o controlador, que por sua vez se comunica com as regras de negócios.
- Imagine uma aplicação financeira que realiza cálculos de diversos tipos, como os de juros. Você pode inserir valores para os cálculos e também escolher que tipo de cálculo será realizado. Isto tudo é feito pela interface gráfica, que para o **modelo MVC é conhecida como View**. No entanto, o sistema precisa saber que você está requisitando um cálculo, e para isso, terá um botão no sistema que quando clicado gera um evento.

MODELO (MODEL), VISÃO (VIEW) E CONTROLADOR (CONTROLLER).



MODEL OU MODELO

- Essa classe também é conhecida como Business Object Model (objeto modelo de negócio). **Sua responsabilidade é gerenciar e controlar a forma como os dados se comportam por meio das funções, lógica e regras de negócios estabelecidas.**

- Ele é o detentor dos dados que recebe as informações do Controller, válida se ela está correta ou não e envia a resposta mais adequada.

CONTROLLER OU CONTROLADOR

- **A camada de controle é responsável por intermediar as requisições enviadas pelo View com as respostas fornecidas pelo Model**, processando os dados que o usuário informou e repassando para outras camadas.
- Numa analogia bem simplista, o controller operaria como o "maestro de uma orquestra" que permite a comunicação entre o detentor dos dados e a pessoa com vários questionamentos no MVC.

VIEW OU VISÃO

- **Essa camada é responsável por apresentar as informações de forma visual ao usuário**. Em seu desenvolvimento devem ser aplicados apenas recursos ligados a aparência como mensagens, botões ou telas.

Seguindo nosso processo de comparação o View está na linha de frente da comunicação com usuário e é responsável transmitir questionamentos ao controller e entregar as respostas obtidas ao usuário. É a parte da interface que se comunica, disponibilizando e capturando todas as informação do usuário.

- Primeiramente o controlador (Controller), que interpreta as entradas do mouse ou do teclado enviadas pelo usuário e mapeia essas ações do usuário em comandos que são enviados para o modelo (Model) e/ou para a janela de visualização (View) para efetuar a alteração apropriada;
- Por sua vez, o modelo (Model) gerencia um ou mais elementos de dados, responde a perguntas sobre o seu estado e responde a instruções para mudar de estado. O modelo sabe o que o aplicativo quer fazer e é a principal estrutura computacional da arquitetura, pois é ele quem modela o problema a ser resolvido;
- Por fim, a visão (View) gerencia a área retangular do display e é responsável por apresentar as informações para o usuário através de uma combinação de gráficos e textos. A visão não sabe nada sobre o que a aplicação está atualmente fazendo, pois tudo que ela realmente faz é receber instruções do controle e informações do modelo e então exibi-las. A visão também se comunica de volta com o modelo e com o controlador para reportar o seu estado.

- Neste artigo, vimos o que é a arquitetura de software, quais são seus elementos, o que são as suas interações. Também vimos mais detalhadamente o padrão de arquitetura de software MVC (Model-View-Controller) que é muito utilizado nos projetos de software. Analisamos os seus elementos e como eles interagem entre si . Além disso, vimos as principais vantagens de adotar um padrão como o MVC que se caracteriza pela facilidade na obtenção de múltiplas visões dos mesmos dados, desacoplagem da interface da lógica da aplicação, entre outras vantagens. No entanto, vimos que definir a arquitetura de um software é ainda mais do que escolher o seu padrão arquitetural, precisamos definir a tecnologia (J2EE ou .Net), linguagens a integrar, forma de persistência, interfaces com o usuário e muito mais. Dessa forma, devemos sempre reunir o máximo de pessoas experientes possíveis, conhecimentos, análise do projeto, todos os seus requisitos funcionais e não funcionais, a fim de definirmos a melhor arquitetura possível para o software que está sendo desenvolvido.