TRABALHO DE ARQUITETURA DE SOFTWARE

Padrão de Arquitetura MVC - Model View Controller

Itens a serem preenchidos com conteúdo abaixo.

[JONAS DEYVID]HISTÓRICO

MVC - The Smalltalk Years

O Engenheiro Civil Christopher Alexander criou base para 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 possa resolver um problema específico em projetos distintos. Através do trabalho de Alexander, profissionais da área de desenvolvimento de software utilizam tais conceitos para iniciar as primeiras documentações de padrões de projetos, tornando-as acessíveis para toda a área de desenvolvimento.

O padrão arquitetural MVC foi inventado por Trygve Reenskaug enquanto ele era um cientista visitante no grupo Smalltalk no famoso Centro de Pesquisas da Xerox em Palo Alto. Ele escreveu seu primeiro artigo no MVC em 1978. Ele originalmente o chamou de padrão Thing Model View Editor, mas ele rapidamente mudou o nome do padrão para o padrão Model View Controller.

Você pode ler seus documentos originais aqui:

http://heim.ifi.uio.no/~trygver/2007/MVC_Originals.pdf

Reenskaug estava tentando resolver o problema de representar (modelar) sistemas complexos do mundo real, como “o projeto e a construção de uma ponte principal, uma estação de energia ou uma plataforma de produção de petróleo offshore”. Um humano tem um modelo mental desses sistemas e um computador tem um modelo digital. Como você faz a ponte entre o modelo mental e o modelo digital?

Ele originalmente definiu Modelos, Visualizações e Controladores assim:

MODELOS - Modelos representam conhecimento. Um modelo poderia ser um único objeto (um tanto desinteressante), ou poderia ser alguma estrutura de objetos.

O modelo (Model) é utilizado para manipular informações de forma mais detalhada, sendo recomendado que, sempre que possível, se utilize dos modelos para realizar consultas, cálculos e todas as regras de negócio do nosso site ou sistema. É o modelo que tem acesso a toda e qualquer informação sendo essa vinda de um banco de dados, arquivo XML.

- VISÕES - Uma visão é uma representação (visual) de seu modelo. Normalmente, destacaria certos atributos do modelo e suprimir outros. Está, portanto, agindo como um filtro de apresentação .É responsável por tudo que o usuário final visualiza, toda a interface, informação, não importando sua fonte de origem, é exibida graças a camada de visão. Pode ter mecanismos que impeçam o usuário de cometer erros, todavia, não podem conter regras de negócio.

CONTROLADORES - Um controlador é o link entre um usuário e o sistema. Ele fornece ao usuário informações, organizando visualizações relevantes para se apresentar em locais apropriados na tela. Ele fornece meios para a saída do usuário, apresentando ao usuário menus ou outros meios de fornecer comandos e dados. O controlador recebe essa saída do usuário, converte-a nas mensagens apropriadas e passa essas mensagens para uma ou mais das visualizações. Em resumo, é a controladora que executa uma regra de negócio (modelo) e repassa a informação para a visualização (visão).

[MAIKE]O MVC é utilizado em muitos projetos devido à arquitetura que possui, o que possibilita a divisão do projeto em camadas muito bem definidas. Cada uma delas, o Model, o Controller e a View, executa o que lhe é definido e nada mais do que isso.

Justificativa

Com o aumento da complexidade das aplicações desenvolvidas, sempre visando a programação orientada a objeto, torna-se relevante a separação entre os dados e a apresentação das aplicações. Desta forma, alterações feitas no layout não afetam a manipulação de dados, e estes poderão ser reorganizados sem alterar o layout.

Esse padrão resolve este problema através da separação das tarefas de acesso aos dados e lógica de negócio, lógica de apresentação e de interação com o utilizador, introduzindo um componente entre os dois: o controlador.

QUANDO DEVE SER UTILIZADO

Com o aumento da complexidade dos sistemas/sites desenvolvidos hoje, essa arquitetura tem como foco dividir um grande problema em vários problemas menores e de menor complexidade. Dessa forma, qualquer tipo de alterações em uma das camadas não interfere nas demais, facilitando a atualização de layouts, alteração nas regras de negócio e adição de novos recursos. Em caso de grandes projetos, o MVC facilita muito a divisão de tarefas entre a equipe.

É indicado usar o MVC quando se quer ter as seguintes vantagens:

- Facilidade de reaproveitamento de código;
- Facilidade na manutenção e adição de recursos;
- Maior integração da equipe e/ou divisão de tarefas;
- Diversas tecnologias estão adotando essa arquitetura;
- Facilidade em manter o seu código sempre limpo;

O gerenciamento da complexidade se torna fácil devido à divisão da aplicação em Model, View e Controller. Esses componentes são

independentes, sendo, então, possível o desenvolvimento paralelo. A inclusão de novos clientes é realizada de forma simples, apenas incluindo seus visualizadores e controles, obtendo-se um melhor reaproveitamento do código. Não só a inclusão, mas também a customização e ou a substituição é facilitada, pois o MVC é completamente extensível. Desenvolvedores que conhecem MVC terão facilidade com o código, pois o mesmo segue um padrão mais legível (legibilidade).

QUANDO NÃO DEVE SER UTILIZADO

O MVC não é apropriado para todas as situações. O design e a implementação de três tipos distintos de componentes, juntamente com suas várias formas de interação, podem ser caros e esse custo pode não fazer sentido para interfaces de usuário relativamente simples. Além disso, a correspondência entre as abstrações do MVC e os kits de ferramentas de interface de usuário comercial não são perfeitos. A view e o controller dividem a entrada e saída, mas essas funções são frequentemente combinadas em widgets individuais. Isso pode resultar em um incompatibilidade entre a arquitetura e o kit de ferramentas da interface com o usuário.

Um exemplo seria uma single page estática com intuito único de mostrar uma companhia ou um produto. Só é preciso do html, css e javascript, e talvez algum framework. Utilizar um MVC nesse caso seria desperdício de recursos.

Um outro exemplo seria um sistema desktop para adicionar os votos das refeições feitas pelos comensais no RU, onde a única funcionalidade do sistema seria, adicionar os votos e gerar porcentagens. Neste caso também não seria recomendado o uso do MVC.

[JONAS LOPES]EXEMPLOS

Aplicação Financeira

Imagine uma aplicação financeira que realiza cálculos de diversos tipos, entre eles os de juros. Você pode inserir valores para os cálculos e também escolher que tipo de cálculo será realizado. Isto tudo você faz 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, você terá um botão no sistema que quando clicado gera um evento.

Este evento pode ser uma requisição para um tipo de cálculo específico como o de juros simples ou juros compostos. Fazem parte da requisição neste caso os valores digitados no formulário, como também a seleção do tipo de cálculo que o usuário quer executar sobre o valor informado. O evento do botão é como um pedido a um intermediador que prepara as informações para então enviá-las para o cálculo. Este intermediador nós chamamos de Controller. O controlador é o único no sistema que conhece o responsável pela execução do cálculo, neste caso a camada que contém as regras de negócios. Esta operação matemática será realizada pelo Model assim que ele receber um pedido do Controller.

O Model realiza a operação matemática e retorna o valor calculado para o Controller, que também é o único que possui conhecimento da existência da camada de visualização. Tendo o valor em “mãos”, o intermediador o repassa para a interface gráfica que exibirá para o usuário. Caso esta operação deva ser registrada em uma base de dados, o Model se encarrega também desta tarefa.

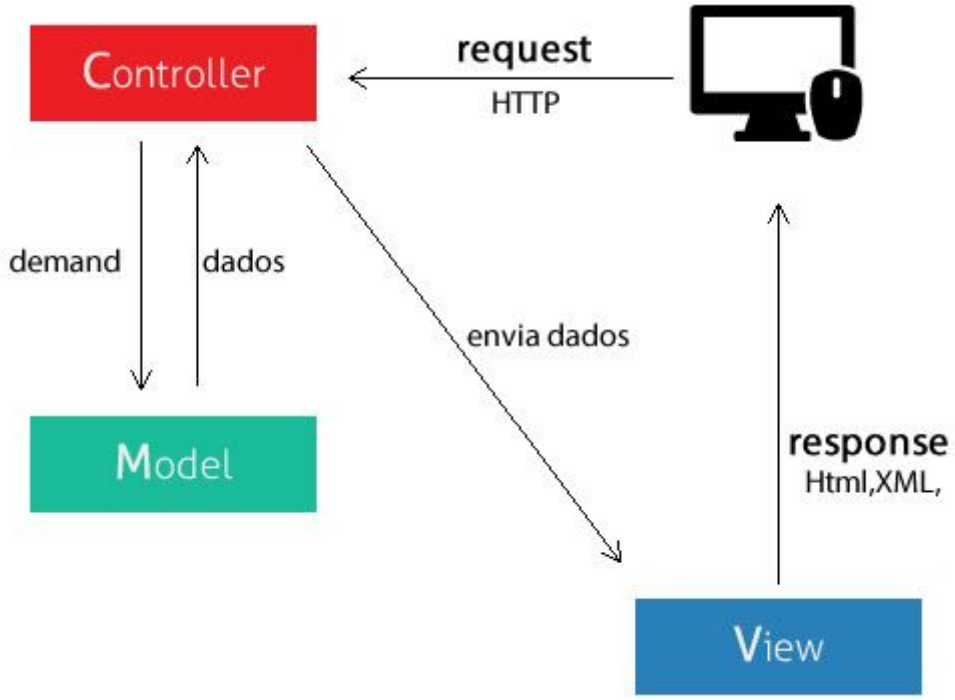
Sugestão de interação - CRIATIVIDADE

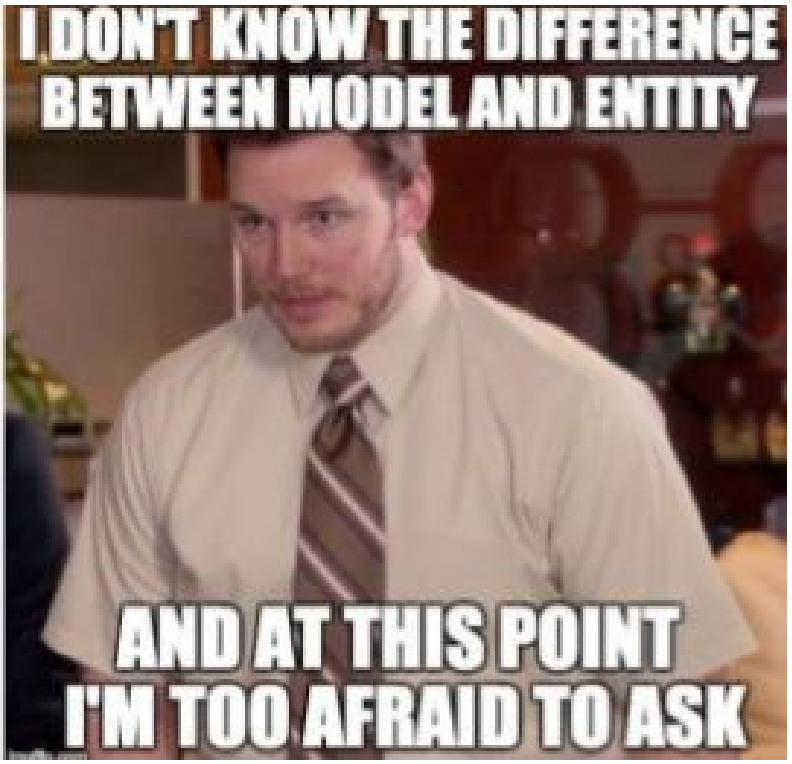
O diálogo das camadas

View: Fala Controller ! O usuário acabou de pedir para acessar o Facebook! Pega os dados de login dele ai.

Controller: Blz. Já te mando a resposta. Ai model, meu parceiro, toma esses dados de login e verifica se ele loga.
Model: Os dados são válidos. Mandando a resposta de login.Controller: Blz. View, o usuário informou os dados corretos. Vou mandar pra vc os dados dele e você carrega a página de perfil. View: Vlw. Mostrando ao usuário...

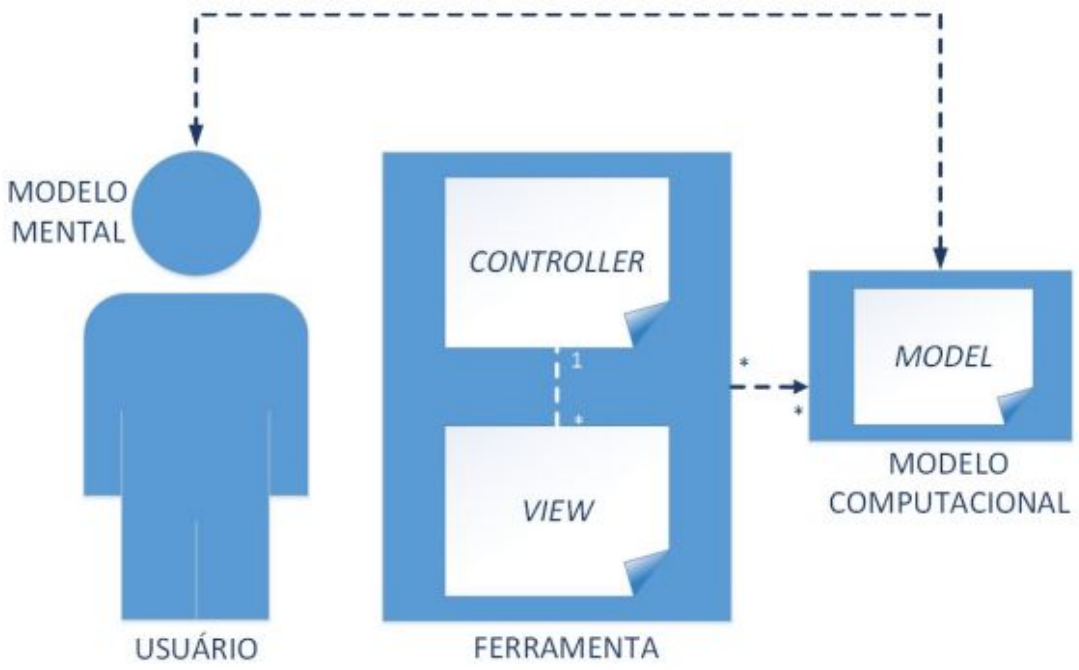
IMAGENS:

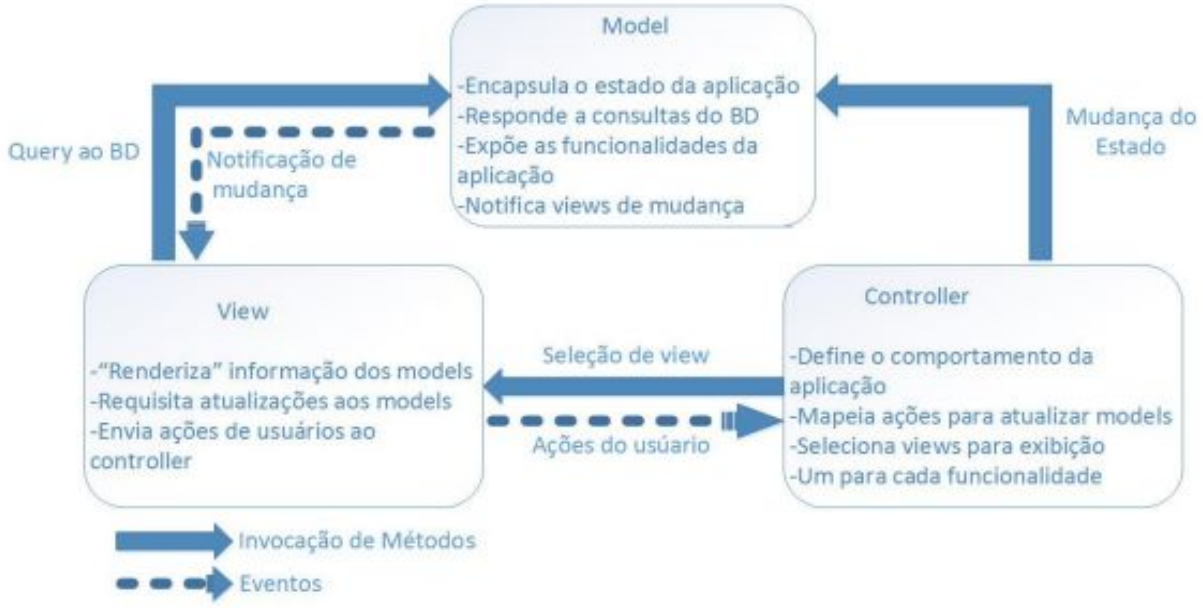


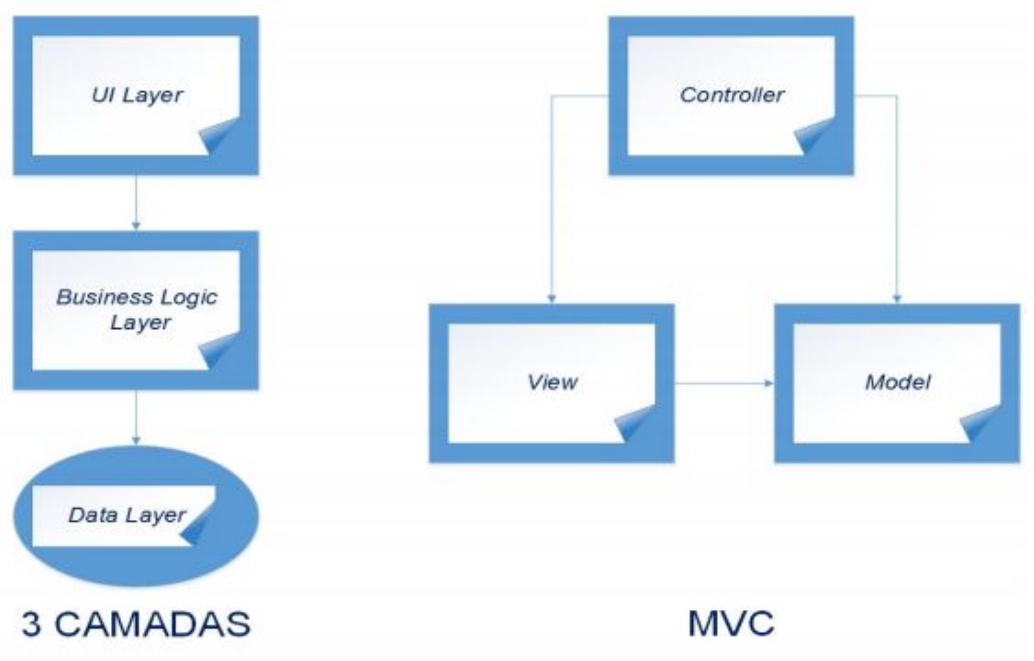


MVC no Android

Imagem que ilustra o conceito original para MVC no Android


MODELO
MENTAL
CONTROLLER
1
*
VIEW
*
MODEL
MODELO
COMPUTACIONAL
USUÁRIO
FERRAMENTA


Model
-Encapsula o estado da aplicação
-Responde a consultas do BD
-Expõe as funcionalidades da aplicação
-Notifica views de mudança
Query ao BD
Notificação de mudança
Mudança do Estado
View
-"Renderiza" informação dos models
-Requisita atualizações aos models
-Envia ações de usuários ao controller
Seleção de view
Ações do usúario
Controller
-Define o comportamento da aplicação
-Mapeia ações para atualizar models
-Seleciona views para exibição
-Um para cada funcionalidade
Invocação de Métodos
Eventos



REFERÊNCIAS BIBLIOGRÁFICAS:

https://www.dropbox.com/pt_BR/business/trust/security/architecture

https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308

http://stephenwalther.com/archive/2008/08/24/the-evolution-of-mvc

https://tableless.com.br/mvc-afinal-e-o-que/

http://heim.ifi.uio.no/~trygver/2007/MVC_Originals.pdf

https://pt.wikipedia.org/wiki/MVC

https://pt.slideshare.net/PauloSato1/historia-do-mvc-mvp-e-mvvm-no-android

https://hangouts.google.com/_/elUi/chat-redirect?dest=http%3A%2F%2Frevistas.unifenas.br%2Findex.php%2FRE3C%2Farticle%2Fdownload%2F54%2F13

2https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture

VIDEOS:

https://www.youtube.com/watch?v=ZW2JLtX4Dag
https://www.youtube.com/watch?v=h9ZIPIdXFuY

1. **Demonstrando**
    - **A descrição do Padrão arquitetural (histórico, caso haja)**
    - **Quando deve ser utilizado**
    - **Quando não deve ser utilizado**
    - **E trazer exemplos do mesmo**
    -
2. **Critérios de avaliação**
    - **Corretude**
    - **Completude**
    - **Didática**
    - **Criatividade (Executar com Exercício ou interação visando a fixação do assunto demonstrado)**

Para dias 3 e 4 de setembro Todo o material que será apresentado deve ser entregue até as 13:30 DO DIA 03 - (25 minutos de apresentação)

DINÂMICA APP AnimalSongs

O grupo escolhe voluntários para representar os papéis de model(três ou mais), view(dois ou mais) e controller(um);

Os models sabem executar serviços específicos de suas atribuições;

O controller recebe as interações da view e mandam para os models requisitados;

Os models executam o serviço requisitado e retornam pro controller, se necessário;

O controller repassa o retorno (se houver) ou uma confirmação pra view;

A view informa para o usuário alguma confirmação caso necessário ou interage com base no retorno do controller;

Papéis da dinâmica:

Models:

GatoService (sabe seu nome, e executa a função miar());

CachorroService(sabe seu nome, e executa a função latir());

BoiService(sabe seu nome, executa a função mugir() e darCoice());

Controller:

ControllerAnimals (apenas controla o fluxo de dados e verifica os dados das entradas, para poupar o estresse dos services)

Views:

ViewAntiga (mostra por escrito no quadro os retornos das funções. Ex.: escreve no quadro “Miau”, se o service for gatoService)

ViewNova (reproduz o “som” do retorno dos services Ex.: O voluntário realmente late ao receber retorno da função latir() do service cachorro)

As entradas são feitas em papel pela equipe, e as mensagens entre os objetos são em papel também.