FACHADA

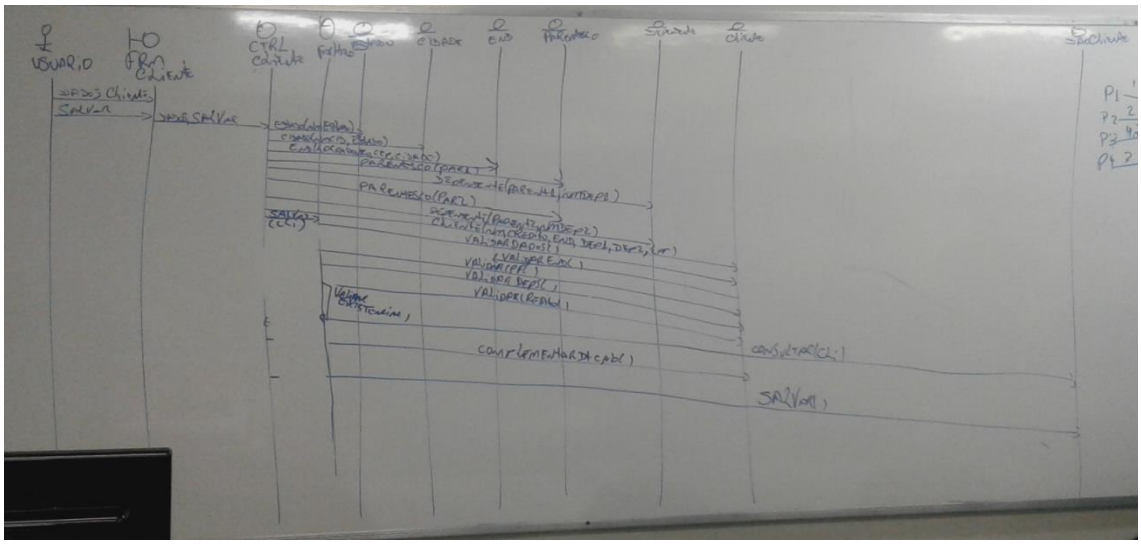


Figura 1

Revisando os conceitos de MVC, este padrão de projeto de arquitetura software serve para alterar a visão, sem a dependência do domínio. Deixar bem claro que o MVC não é a mesma coisa que as 3 camadas (Camada Cliente, aplicação e serviço), que é uma arquitetura de sistemas, no qual se olha a parte física ou lógica do sistema. O MVC é uma implementação que normalmente fica na camada de negócio ali dentro da arquitetura de sistemas.

Na arquitetura de 3 camadas, na camada de aplicação existem vários softwares: Exemplo servidor de aplicação como o apache, tomcat, ERP, e-commerce, etc. Mas nesta camada também pode ter softwares de desktop. Um arquiteto de software é responsável por definir a estrutura para as soluções de sistemas, definir quais tecnologias deverão ser aplicadas e como devem ser organizadas. Um arquiteto deve ter a visão de tudo do que deverá conceber numa arquitetura de sistemas e de software. É ele quem deverá estabelecer custo e prazos da arquitetura, regras de negócio, requisitos e o ciclo de vida de um produto, etc.

Relembrando todas as arquiteturas ensinadas:

1 – Tudo numa mesma classe;

2 – Separação em classes de domínio;

3 – DAO + 2, que isola a camada de persistência do domínio;

4 – MVC, que isola a visão da camada de domínio, ou que estabelece a fácil manutenção da camada de visualização, sem alterar o domínio.

A quinta camada, a fachada é um interpretador de tecnologias, ou seja, esta nova camada permite a alteração e adaptação de novas tecnologias a serem implantadas. O que acontece quando sai uma nova versão de JAVA e que o desempenho dos programas desenvolvidos em JAVA, para web, sejam executadas mesmo assim e com performance ainda melhor. Isso facilita a manutenção. Exemplo: mudança de HTML para HTML5 ou Flash.

O modelo de 3 camadas existiu para possibilitar a centralização das regras de negócio, escalar os usuários mantendo os dados centralizados e poder mudar as regras de negócio sem impactar os clientes. Diferentemente do modelo de 2 camadas, quando ocorresse uma atualização, ela deverá ser efetuada uma por uma máquina. Hoje, do cliente pode ser originados de diversos dispositivos e de diferentes formatos: como tablete, smartphone, etc.

A implementação dos requisitos vai muito além da implementação de validações, pois a sequencia de como chamamos essas validações, impacta na implementação dos requisitos. Então, não é simplesmente mudar a camada de visualização, mas sim fazer com que uma nova camada de controle saiba qual a sequência a ser chamada, independente se for uma aplicação desktop, android, ios, web, a sequência deverá ser conhecida e será perdida. Conclusão, a fachada só é desnecessária se a tecnologia da aplicação for sempre a mesma. Esta camada faz com que as regras de negócio seja reaproveitada, independente da tecnologia, ou seja, a sequência da implementação seja reaproveitada. A fachada pode ser definida também como uma interface que centraliza as chamadas de diversos subsistemas ou serviços, sem impactar os tipos de clientes.

Devemos entender que cliente é qualquer componente de software que acessa e demanda o uso de alguns serviços, e serviço

como qualquer componente que provê um serviço qualquer. Portanto, podemos ter situações, quando programamos orientado a objetos, em que temos um cliente (que pode ser uma classe que esteja acessando outra) que esteja utilizando outros serviços.

Figura 2

O conjunto dos programas que estejam sendo utilizados, façam com que a implementação de um sistema seja estabelecida. A fachada estabelece uma interface única, que pode ser acessada por vários clientes e ainda é responsável por executar todos os serviços na sequência correta para que o requisito ou serviço seja realizado por completo, sem precisar cada cliente acessar de forma independente e sem sequência (bagunçada como na figura acima).

Figura3

Com a fachada, nós centralizamos a chamada dos subsistemas, portanto, a fachada é nada mais que criar a definição ou interface para a chamada de diversos subsistemas. Quando tratamos de orientação a objetos, cada classe pode ser vista como subsistema, ou um programa. Já a fachada cria uma interface de

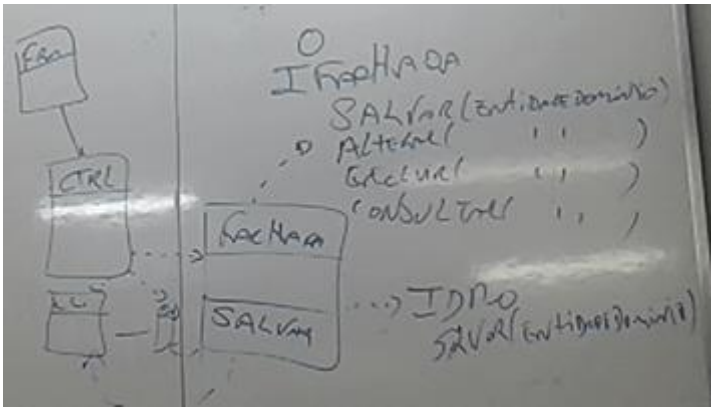
alto nível ou uma interface abstrata que executa a chamada dos diversos subsistemas. Esta interface também possibilita que uma classe realize uma sequência incompleta de serviço, ou seja, o que for apenas necessário para atender a requisição dela. Ela também pode servir como uma API.

Com a fachada, não é possível mudar a camada de visualização, mas todos os eventos das regras de negócio (exceto as instanciações), que eram chamados pela camada de controle no MVC, passam a ser chamados pela fachada (figura 1).

Em uma aplicação web, por exemplo, de cadastrar cliente e o botão de salvar é pressionado, é gerado uma requisição HTTP no servidor, que tem o recurso, que pode ser uma servlet, que saiba interpretar essa requisição. A servlet pega a referida requisição e extrai desta os dados, instancia as classes de domínio, pega essas instancias e manda a fachada realizar os serviços de validações e salvamento. A fachada então é quem deverá saber o que validar, chamar a validação, saber qual DAO deverá chamar, deverá ter o conhecimento quais subsistemas que fará todo os serviço desejado numa aplicação.

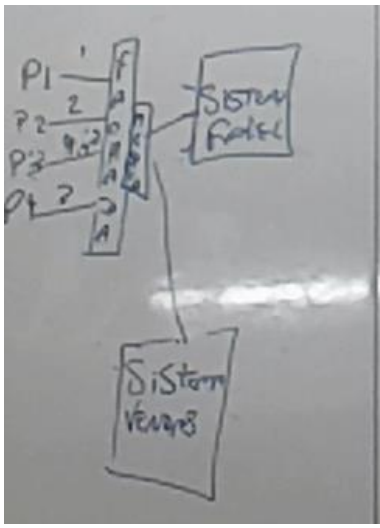
Com a fachada agora, a partir da classe estado da figura 1 serão as classes que representam o MODEL, a fachada e unicamente ela passou a ser a CONTROLLER e a classe de controle deixou de ser a camada de CONTROLLER e passou a ser a camada de **controle da visualização**, ou seja, estará junto com a camada de visualização, ou seja, passou a ser VIEWER (controle de visualização). Na camada de visualização posso ter outra estrutura de MVC e que temos uma controller específica para a VIEWER. Quando houver uma alteração de tecnologia, iremos implementar uma controller específica da tecnologia e que saiba interpretar requisições e chamadas da devida tecnologia, transformar isso num domínio e passar para a controller de uma aplicação e esta controller deverá saber como resolver as chamadas dos subsistemas. As chamadas dos métodos que atendem as regras de negócio serão feitas pela fachada.

Assim como a DAO, a fachada também possui uma interface, a diferença é que a DAO tem a responsabilidade de fazer a persistência de dados, já a fachada se responsabiliza de chamar os subsistemas. Numa aplicação web, a IFachada substitui o IDAO e com as respectivas classes que as implementam também acontece essa substituição. No IDAO só permanece o método de Salvar, pois esta, como dito anteriormente, é responsável pela persistência de domínio.



Resumindo: A fachada resolve o problema da sequência de chamadas de métodos ou serviços, que atende uma dada requisição de um ou mais clientes independente da tecnologia que está sendo usada. Quando uma sequência original muda, sem a fachada todos os clientes serão impactados. A fachada serve de interface que centraliza (abstrai) as chamadas do serviço e que, quando a sequência ou tecnologia mudar, os clientes não serão mais impactados, pois continuarão chamando uma única interface.

Portanto a fachada não vai resolver o problema de MVC, mas sim todo problema em que temos um conjunto de serviços a serem executados, numa sequência estabelecida, que podem ser chamadas por diversos clientes. Uma vez que essa sequência ou a forma de chamada desses serviços mudarem, os clientes não serão mais impactados, pois a fachada cria uma abstração dessas chamadas de serviços, implementações, etc.



- RECEBE OS NÚMEROS
- SOMA OS NÚMEROS P2
- DIVIDE PELA QTDE DE NUMEROS P3
- VERIFICA DIVISOR P4

2235 = 12/4

25