**FACULDADE DE TECNOLOGIA SENAC GOIÁS**

**Alunos:**
Adriel Miranda
Brendon Peixoto
Fábio Dasmacena
Rafael Galdino

**Análise e Desenvolvimento de Sistemas**
Arquitetura de Software

**Projeto Integrador**

Goiânia,
Junho de 2020.

**Arquitetura de Software**

**1. Descrição da arquitetura N-Tiers (N-Camadas)**

**Tier** geralmente se referencia a **camada física**, enquanto **layer** geralmente se referencia a camada lógica. Isso é um conceito mais antigo, hoje basicamente, a maioria dos servidores rodam na núvem, onde essa “separação física” não faz muito sentido. Tratamos Tier e Layer como o mesmo sentido da palavra (camada / nível)

Tier / Layer é camada que representa uma sequência lógica, ponto. Você pode criar uma aplicação de 1..N camadas, da forma que você achar que é o correto. Isso nao significa necessariamente que é algo físico ou em ambientes separados. é somente a separação abstrata de sua aplicação em sí. Por exemplo hoje em dia é bem conhecido aplicações em 3 camadas. (não confundir com MVC).

### Ilustrando o ambiente:

**Camada 1:** Camada de apresentação, onde fica tudo relacionado a view, pode ser telas do Java Swing, Forms do .Net, ou paginas web.

**Camada 2:** Camada de lógica. Todas as regras específicas de sua aplicação, é a parte central de sua app. Onde de fato você está concebendo suas idéias.

**Camada 3:** Acesso ao banco. Bom aqui, você... acessa o banco =). E também pode haver alguma lógica relacionado à transação, etc.

Nada te impede de criar uma app com 1,2,3,4 ou 5+ camadas, por exemplo as 3 acimas citadas, mais outra de integração Rest, outra específica de validação e por ai vai. Tudo depende do requerimento do projeto. Em geral 3 é o suficiente ou até demais.

### Porque usamos camadas ?

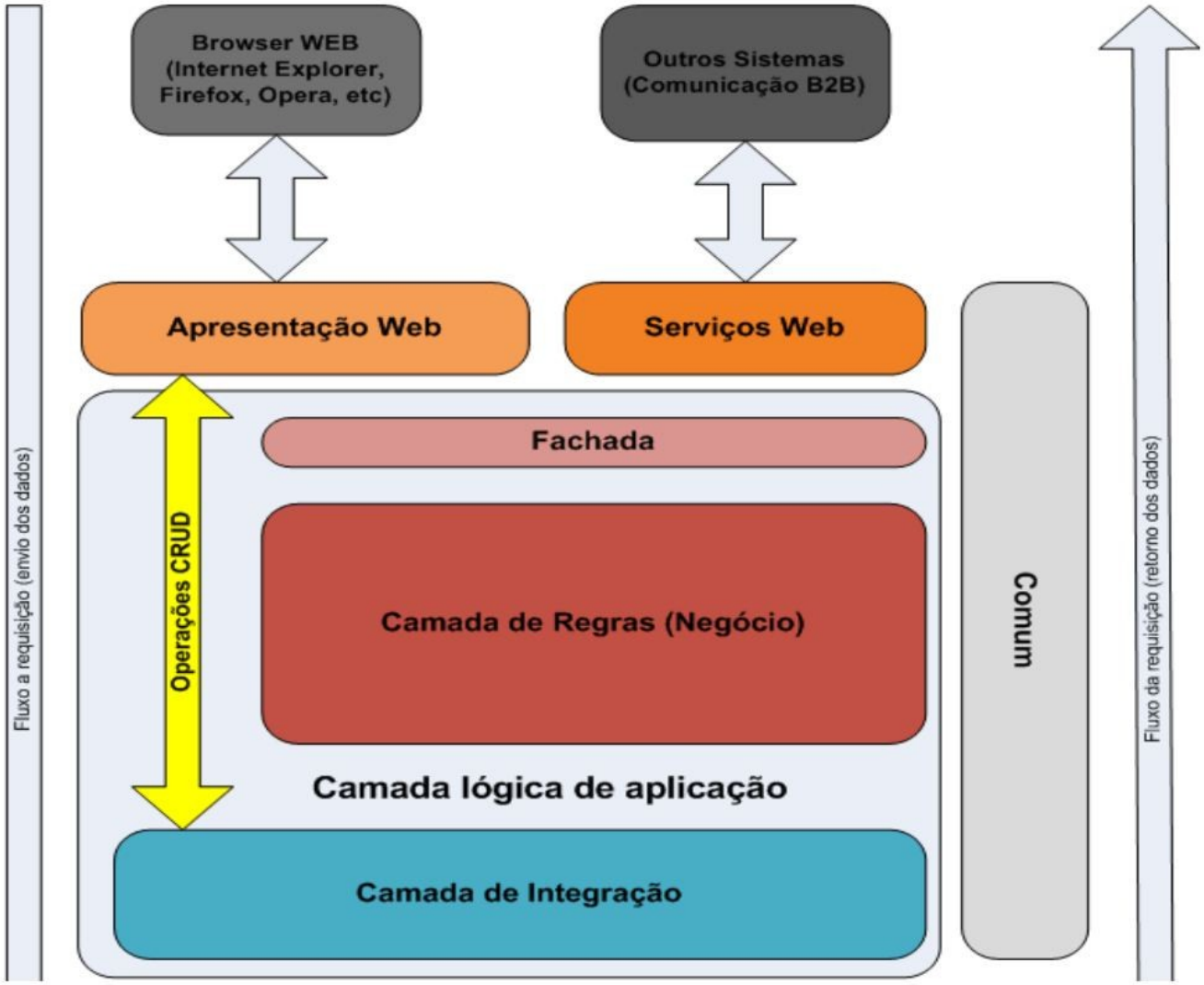
Na prática é para separar os conceitos. **Geralmente** é feito a comunicação através de interfaces, oque facilita muito caso precise adicionar ou substituir uma delas. Outra situação é quando se trabalha em equipe. Onde você tem os carinhas do frontend desenvolvendo a interface gráfica, outros no “core”, etc. Onde cada equipe não altera o código da outra equipe, tudo isso trabalhando em cima do mesmo projeto. E ao mesmo tempo eles sabem oque passar e oque receber da camada superior/ inferior por utilizarem os contratos das interfaces. Tudo isso no mundo de POO, outros paradgmas não sei bem ao certo como deve funcionar.

Frisando que "camada" é a separação abstrata da lógica de seu projeto. Porém quando se tratando de grandes arquiteturas, podemos também ter N camadas, a nivel de aplicação, onde cada camada pode corresponder a uma aplicação rodando em um servidor, que se comunica com outro rodando em outro servidor, um bom exemplo disso são aplicações comercias, onde as coisas começam a ficar bem grandes.

Em geral mais camadas não significa ser melhor, quanto mais camadas, mais desacoplamento você tem, oque é bom mas muitas vezes desnecessário, porém gera uma maior complexidade e trabalho na manutenção.

Note que camada possui o esquema de pilha, onde para chegar a última você deve passar por todas as intermediárias, ex: **camada 1 <-> camada 2 <-> camada 3**

**2. Camadas Layers da Aplicação:**




As camadas que compõem o diagrama da solução proposta estão detalhadas a seguir:

**2.1 - Camada de Apresentação**

O sistema possuirá duas camadas de apresentação:

**2.1.1 - Camada Web / GUI**

A camada de apresentação web/Desktop/Mobile será responsável por prover as funcionalidades para os atores ou usuários diretos do sistema (GUI – Graphical User Interface).

Todas as mensagens do sistema devem estar encapsuladas em arquivo de mensageria, estando o sistema preparado para suporte a multi-linguagem (diversos idiomas como PT_BR, FR, EN, ES, etc).

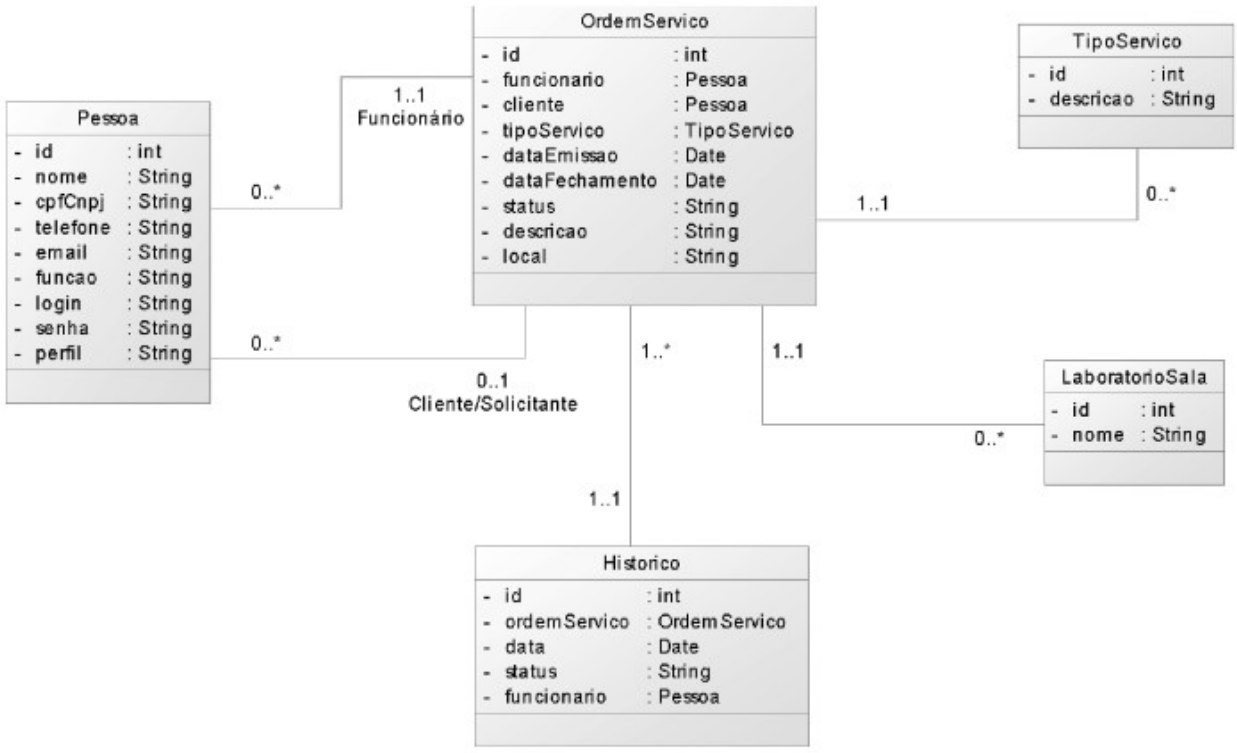
### 2.1.1.2 - Segurança: Autenticação e Autorização

O sistema deverá invocar os serviços da plataforma para recuperar o usuário logado. Somente será habilitado a usar aquele usuário cadastrado e ativo na plataforma (mesmo que simulada, no momento).

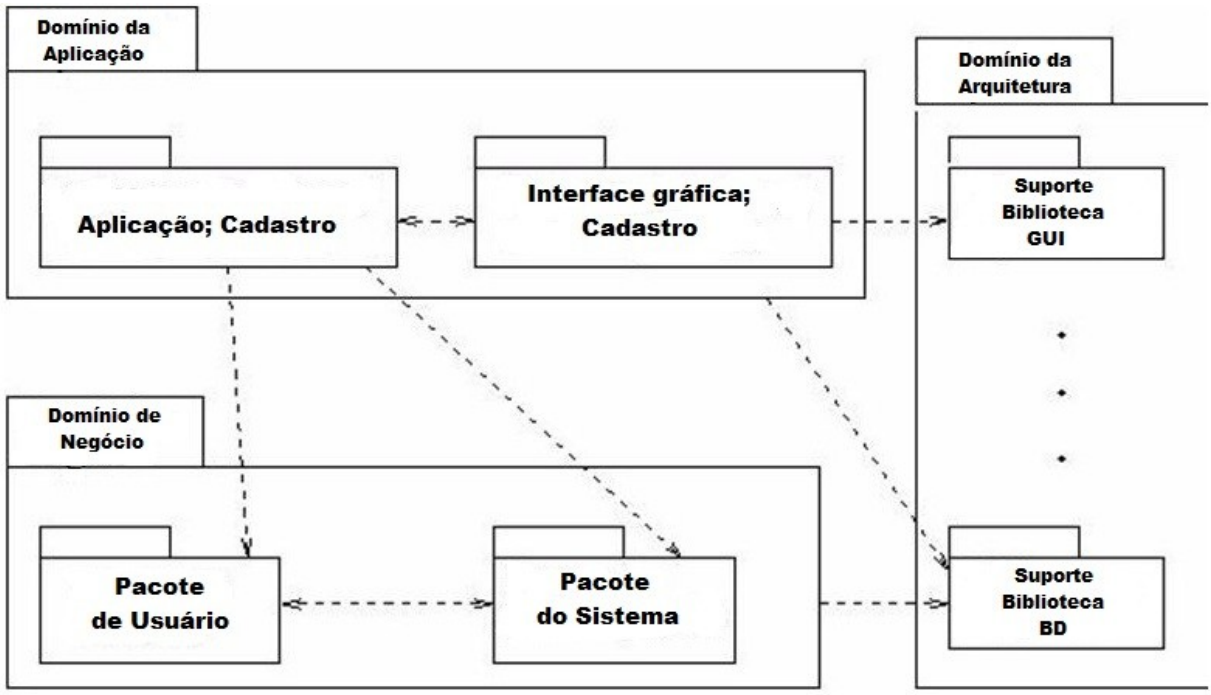
### 2.1.2 - Camada de Serviços WEB

A camada de serviços Web (Web Services) será responsável pelo processo de disponibilização de regras de negócio existentes na plataforma e deverá ser criado pelo desenvolvedor para apenas simular a interação com o sistema. Esta camada deve ser criada em linguagem JAVA.

## 3. Diagrama de Classes

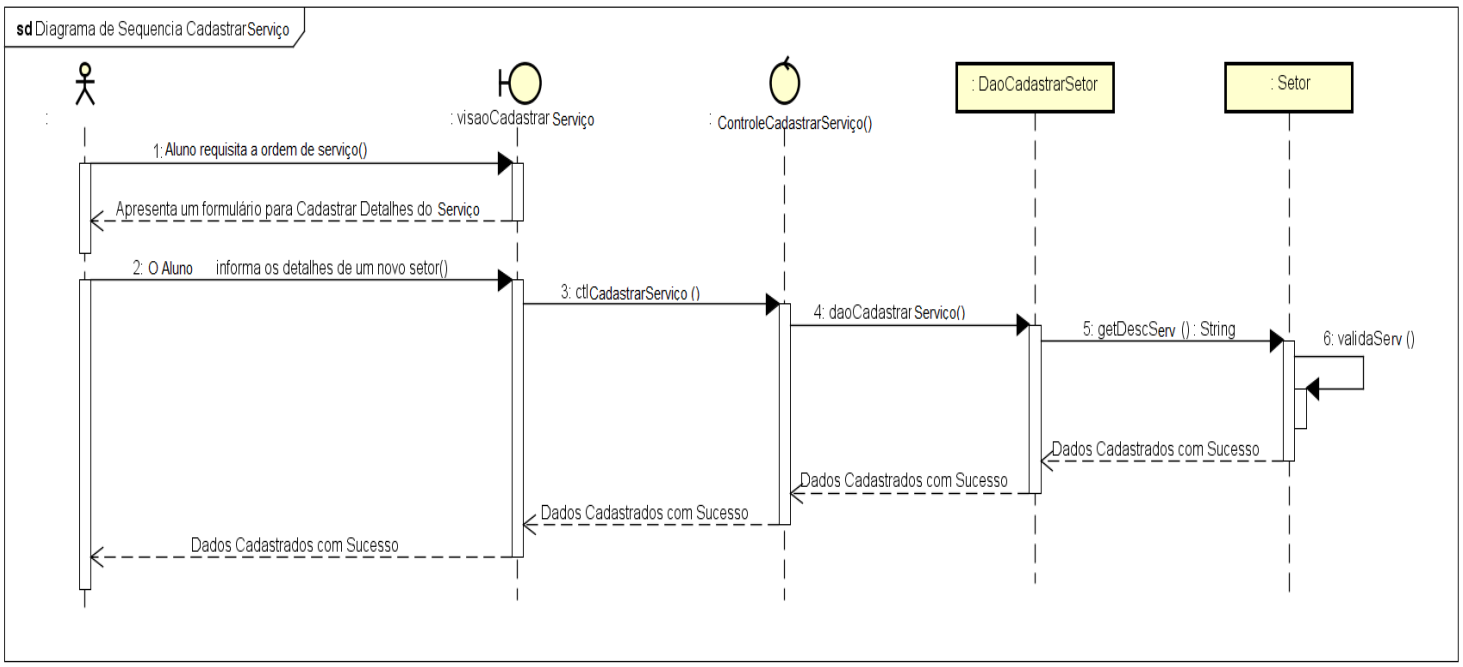



## 4. Diagrama de Pacotes



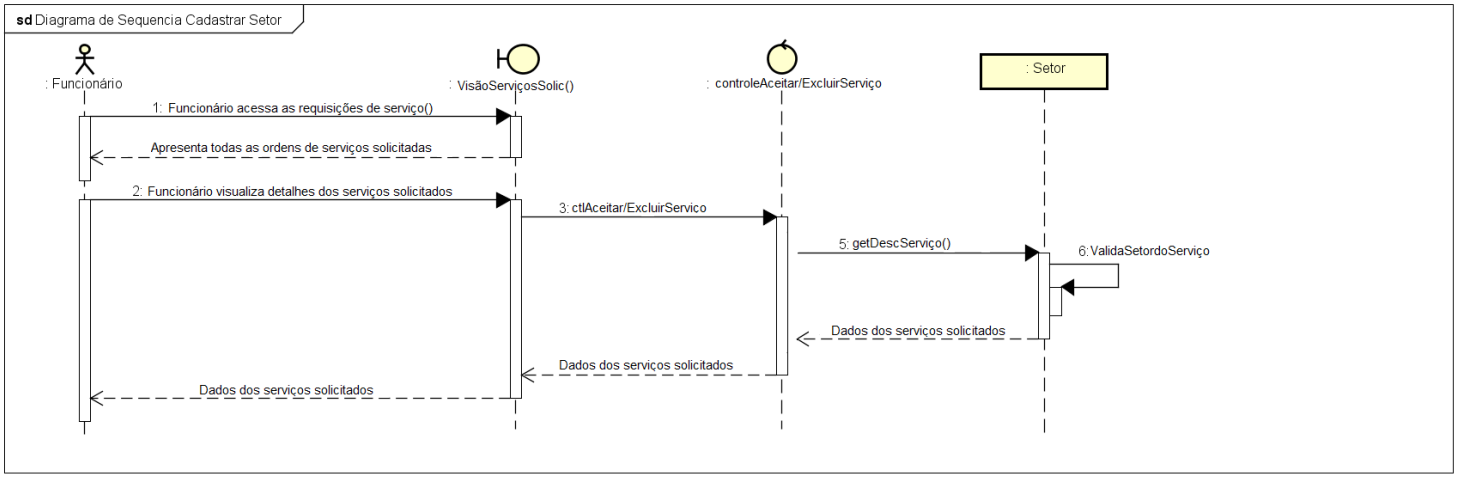

## 5. Diagrama de Sequência

**Aluno:**






**Funcionário:**






## 6. Objeto de Domínio