**Arquitetura de Desenvolvimento de Aplicações – ADA**

**SGPTI-HUUFS**

Aracaju (SE)

2018

# Sumário

[Sumário 2](#_Toc522616519)

[Introdução 3](#_Toc522616520)

[Objetivo 3](#_Toc522616521)

[Arquitetura Atual 4](#_Toc522616522)

[Detalhamento das camadas: 5](#_Toc522616523)

[Camada de Apresentação 5](#_Toc522616524)

[Camada de Controle 6](#_Toc522616525)

[Camada de Regra de Negócio 6](#_Toc522616526)

[Camada de Acesso a Dados 7](#_Toc522616527)

[Banco de Dados 8](#_Toc522616528)

[Padrão de Nomenclatura de Banco de Dados 9](#_Toc522616529)

[Nomenclatura dos Objetos 9](#_Toc522616530)

[Tabelas 9](#_Toc522616531)

[Visões Materializadas 9](#_Toc522616532)

[Atributos 10](#_Toc522616533)

[Segurança e Controle de Acesso 11](#_Toc522616534)

# Introdução

O Setor de Gestão de Processos e Tecnologia da Informação do Hospital Universitário da Universidade Federal de Sergipe possui uma equipe de Desenvolvimento de Sistemas que produz soluções para uso interno do Hospital.

# Objetivo

Este documento visa mapear as arquiteturas utilizadas no desenvolvimento de sistemas do setor. Após esse processo, será definida uma arquitetura única a ser continuada, bem como será confeccionada a documentação da arquitetura definida para auxiliar no processo de aprendizado de novos desenvolvedores.

# Arquitetura Atual

O Desenvolvimento de Sistemas no SGPTI segue o padrão de 5 camadas com utilização da Linguagem JAVA e do framework Spring MVC, que possibilita aos desenvolvedores um aumento significativo em produtividade, devido à facilidade de configurações via anotações.

Principais anotações:

**@Component**

Anotação genérica – da qual todas as anotações são derivadas - para qualquer componente gerenciado pelo Spring.

**@Controller**

Anotação que serve para definir que uma classe processa requisições da aplicação web.

**@Service**

Anotação que serve para definir que uma classe pertence à camada de Serviço e contêm o fluxo de negócio da aplicação.

**@Repository**

Anotação que serve para definir que uma classe pertence à camada de persistência. Comumente associada a DAO’s.

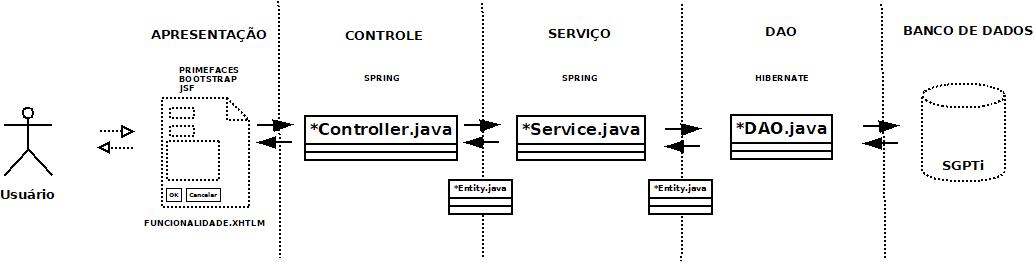
**@Entity**

Anotação utilizada para informar que uma classe também é uma entidade. Utilizada junto com as Anotações @Table e @Column, formando a relação entre a entidade e a tabela no banco de dados, onde os dados de objetos desse tipo poderão ser persistidos.

**@Autowired**

Anotação utilizada para marcar o ponto de injeção na classe. Pode ser colocada sobre atributos ou sobre o construtor.

# Detalhamento das camadas:



## Camada de Apresentação

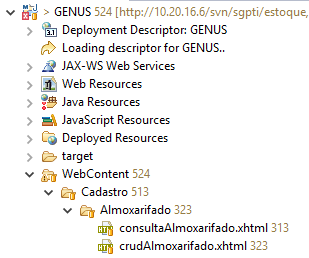
A primeira camada é chamada de Camada de Apresentação que é composta pelas telas das funcionalidades do Sistema. Esta é a camada que interage com o usuário, solicitando as informações necessárias ao processamento da função e exibindo o retorno enviado pelas camadas seguintes.

As telas são constituídas por arquivos XHTML, compostos por componentes JSF com as bibliotecas Primefaces e Bootstrap. Para padronização das telas é utilizada a template adminLTE.

Para cada Funcionalidade existem 2 arquivos XHTML: consulta e crud.

Exemplo: **consultaAlmoxarifado.xhtml e crudAlmoxarifado.xhtml**

As funcionalidades estão organizadas em diretórios seguindo uma hierarquia de acordo com o negócio, como pode ser visto na imagem abaixo:



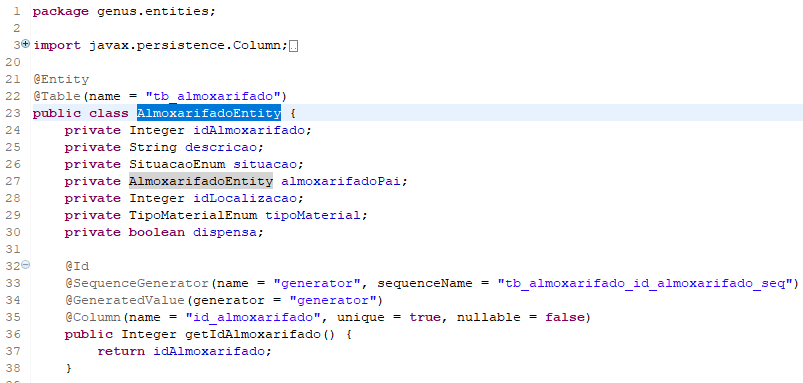
## Camada de Controle

A Camada de Controle é formada por Classes que interagem com a Camada de Apresentação, fazendo validações e consumindo serviços da Camada de Regra de Negócio.

Exemplo: genus.controller.AlmoxarifadoController.java

A partir desta camada são utilizadas também Classes de Entidades para otimizar a passagem de parâmetros entre as camadas. Estas Classes encapsulam conjuntos de dados afins e são implementadas através de uma classe convencional Java, segundo os padrões PO – Persistent Object e VO – Value Object. As Classes de Entidade não contêm regra de Negócio ou métodos adicionais.

Exemplo: **genus.entities.AlmoxarifadoEntity.java**

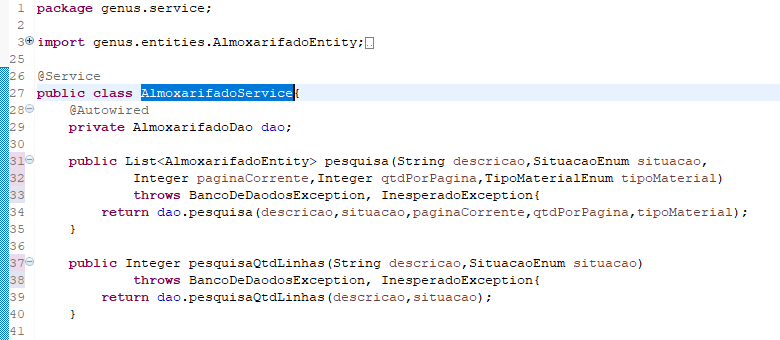


## Camada de Regra de Negócio

A camada de Regra de Negócio é formada por Classes de Serviço que possuem os métodos disponíveis para cada Aplicação. Estes métodos possuem as regras de Negócio e interagem com a camada de Acesso a Dados para obter ou manter Informações no Banco. As classes de Serviço podem interagir também com outras classes de Serviço.

.

Exemplo: **genus.service.AlmoxarifadoService.java**



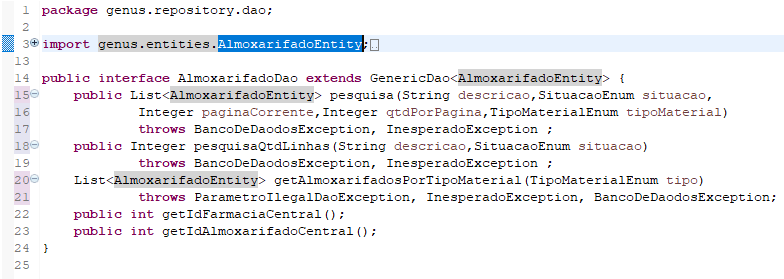
## Camada de Acesso a Dados

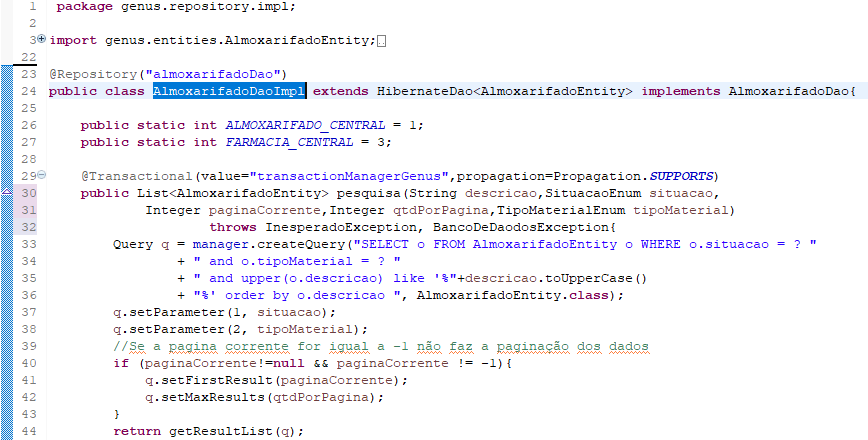
A camada de Acesso a Dados é formada por Classes do tipo DAO – Data Access Object, que permitem o acesso de Leitura e Gravação às informações do Banco de Dados.

Por padrão é utilizada uma interface com a assinatura dos métodos e uma classe com a sua implementação. São chamados respectivamente de <Funcionalidade>Dao e <Funcionalidade>DaoImpl

Exemplos: **genus.repository.dao.AlmoxarifadoDao.java**

**genus.repository.impl.AlmoxarifadoDaoImpl.java**





## Banco de Dados

A camada de Banco de Dados é formada pelos Depósito de dados da Instituição. É o local onde estão depositadas todas as Informações essenciais aos Sistemas, sejam elas de pacientes, atendimentos e exames bem como de materiais utilizados na Instituição. O Banco de Dados utilizado é do tipo relacional e existem dois Modelos:  
MySQL e PostgreSQL.

# Padrão de Nomenclatura de Banco de Dados

Atualmente não existe um padrão para nomenclatura de Banco de Dados. Para um Banco de Dados Pequeno, com poucas tabelas, isso parece não ser importante, mas quando o sistema precisa trabalhar com centenas de tabelas, as coisas mudam. Será muito difícil extrair alguma informação de um Banco que não segue nenhuma regra de nomenclatura de Tabelas, Relacionamentos e Colunas.

O uso de metodologias reduz os custos de desenvolvimento e manutenção do Sistema, facilitando a construção de comandos para Manipulação dos Dados.

## Nomenclatura dos Objetos

Não usar “ç”, acentuação e nem caracteres especiais.

O nome não deve ultrapassar 30 posições e não deve conter preposições.

Todos os nomes serão escritos em letras maiúsculas. Todas as palavras e abreviações devem ser separadas pelo caractere “UNDERLINE” (\_). Exemplo: PESSOA\_FISICA.

Utilizar uma Sigla com 3 letras que identifique o objeto. Caso a sigla já tenha sido usada, encontrar uma solução que torne a identificação intuitiva. Exemplo: PFI\_PESSOA\_FISICA.

## Tabelas

Prefixo **TB** + “\_” + **Sigla** + “\_” + **Nome\_Tabela**.

Exemplos:

TB\_MUN\_MUNICIPIO

TB\_PES\_PESSOA

TB\_PJU\_PESSOA\_JURIDICA

## Visões Materializadas

Prefixo **VW** + “\_” + **Sigla** + “\_” + **Nome\_Visão**.

Exemplos:

VW\_EPE\_ENDERECO\_PESSOA

VW\_PPA\_PRONTUARIO\_PACIENTE

## Atributos

O nome do atributo será formado pela concatenação da Sigla do Objeto a que pertence seguido do domínio do atributo e mais os termos que o formam.

Quanto ao domínio, o atributo pode ser:

|  |  |
| --- | --- |
| CD | CÓDIGO |
| NR | NÚMERO |
| DT | DATA |
| HR | HORA |
| VL | VALOR |
| DS | DESCRIÇÃO |
| NM | NOME |
| TP | TIPO |
| QT | QUANTIDADE |
| ST | STATUS |
| IM | IMAGEM |
| FL | FLAG |
| BY | BYTES |
| SG | SIGLA |

Exemplos:

PES\_CDPESSOA

PES\_NMPESSOA

MUN\_CDMUNICIPIO

MUN\_NMMUNICIPIO

# Segurança e Controle de Acesso

O Sistema de Controle de Acesso (CA) é uma espécie de Portal para as aplicações.