ManSAD

Arquitetura de Software

Versão 1.0

Histórico da Revisão

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 03/09/2013 | 0.1 | Criação do documento;  Definição de tópicos para documentação;  Inclusão de conteúdo breve de explicação; | Bruno César |
| 09/09/2013 | 0.2 | Inclusão de diagrama da visão de negócio;  Inclusão de diagrama da visão lógica;  Inclusão e descrição dos pacotes de implementação ; | Bruno César |
| 12/09/2013 | 1.0 | Incrementação nos textos descritos;  Atualização de diagramas; | Bruno César |

Conteúdo

[1. Introdução 5](#_Toc366791806)

[Finalidade 5](#_Toc366791807)

[Escopo 5](#_Toc366791808)

[Definições, Acrônicos e Abreviações 5](#_Toc366791809)

[2. Metas e Restrições da Arquitetura 5](#_Toc366791810)

[3. Visão Lógica 6](#_Toc366791811)

[Visão Geral 6](#_Toc366791812)

[Pacotes de Design 8](#_Toc366791813)

[4. Visão de Negócio 8](#_Toc366791814)

[5. Visão de Implementação 10](#_Toc366791815)

[Visão Geral 10](#_Toc366791816)

[Camadas 10](#_Toc366791817)

[Figura 1 - representação em pacotes do módulo avadoc-server 7](#_Toc366791734)

[Figura 2 - representação em pacotes do módulo avadoc-web 7](#_Toc366791735)

[Figura 3 - visualização em componentes para entendimento do negócio 9](#_Toc366791736)

[Figura 4 - camadas do software de acordo com classes genéricas 11](#_Toc366791737)

# Introdução

## Finalidade

No presente documento poderão ser encontradas as visões necessários para o fácil entendimento da arquitetura de software adotada para o projeto de manutenção de software em curso.

As visões que foram consideradas desnecessárias para o fácil entendimento do documento não foram descritas.

## Escopo

O presente documento, intitulado “Arquitetura de Software” irá documentar a organização lógica de artefatos de código, mostrando as seguintes visões:

* visão lógica: a divisão em módulo Maven e pacotes (namespaces) de classes podem ser entendidos aqui. Ele descreverá todos os principais pacotes de classe, de forma a se ter a visão de como as classes estão organizadas nos artefatos de código.
* visão de implementação: mesmo as camadas de software serem visíveis na visão lógica, esta seção mostrará com mais detalhes como os artefatos que implementam a arquitetura estão relacionados.
* visão de negócio: nesta seção será possível visualizar em alto nível a organização do projeto, como eles são organizados de forma a atender à necessidade de negócio a qual se propõe.

## Definições, Acrônicos e Abreviações

As definições, acrônimos ou abreviações por ventura utilizados neste documento podem ser encontradas no glossário do projeto.

Não foram utilizadas definições de nicho específico para o projeto que não aquelas encontradas no glossário, conforme descrito acima.

# Metas e Restrições da Arquitetura

Para o projeto do qual esta arquitetura de software faz parte, ou seja, o componente de software executável ao qual tal arquitetura é aplicada, possui algumas restrições de plataformas e até mesmo ferramentas.

Sendo um projeto para manutenção de sistema de software que não obteve sucesso em sua primeira implementação, a arquitetura aqui definida presou pela simplicidade, de forma a não onerar o processo de construção do software, tendo em vista o cronograma apertado. A arquitetura de software originalmente concebida para o projeto pode ser encontrada no repositório de gestão de artefatos do projeto anterior.

O projeto, assim como o original, deve ser construído sobre a plataforma Java, usando também a linguagem Java para sua construção, além de ser necessário o uso do Maven como ferramenta de build e gestão de dependências.

Não já, no entanto, obrigatoriedade no uso de uma IDE específica, ficando a cargo do desenvolvedor a escolha de sua IDE de preferência.

# Visão Lógica

Na **Visão Geral** desta seção será explanado a divisão em subsistemas e em **Pacotes de Design** contém a lista com os pacotes de classes que serão encontrados na implementação do sistema.

## Visão Geral

Admitindo a divisão do projeto de software em dois grandes módulos, chamados de **avadoc-server** e **avadoc-web**, temos a seguinte representação de pacotes:

* avadoc-server: o módulo avadoc-server contém todos os artefatos de código que são comuns ao possíveis clientes existentes. Nele, estão as representações para objetos que fazem acesso a dados, modelo de entidades de domínio, serviços para serem acessados por clientes e objetos utilitários.

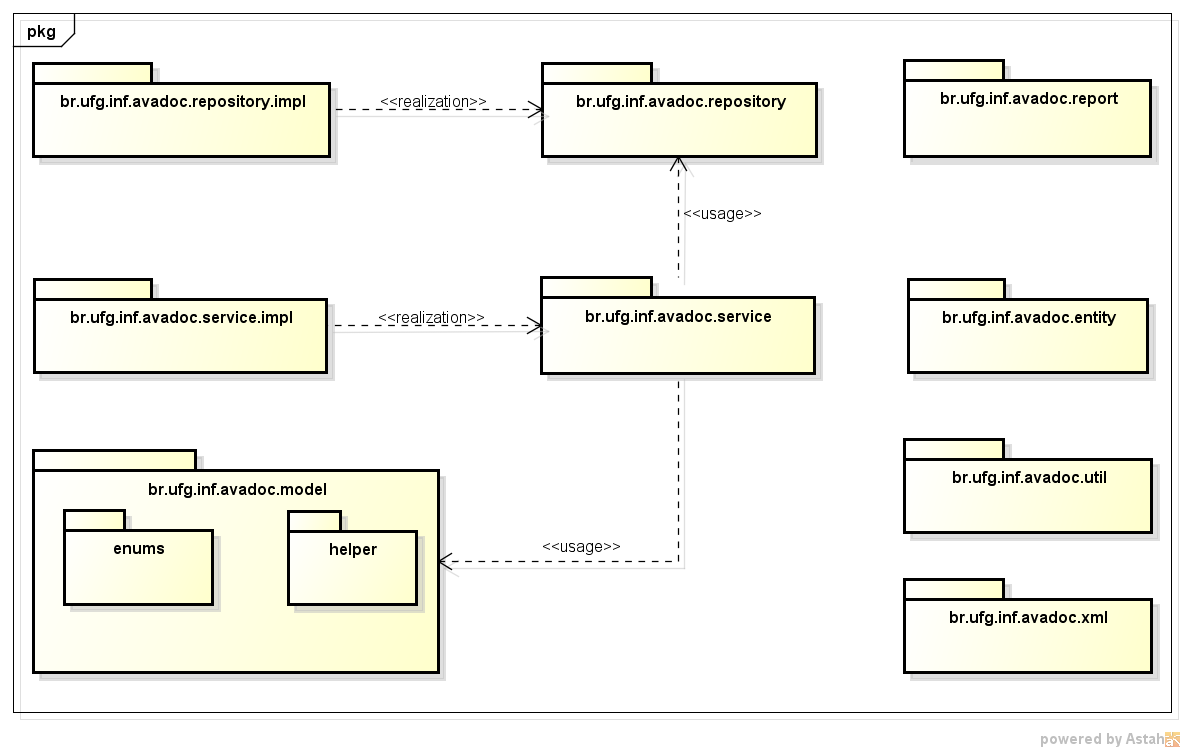


Figura 1 - representação em pacotes do módulo avadoc-server

* avadoc-web: o módulo avadoc-web é um exemplo de cliente. Ele é encarregado de implementar funcionalidade de interação com o usuário, podendo ser acesso em um browser web.

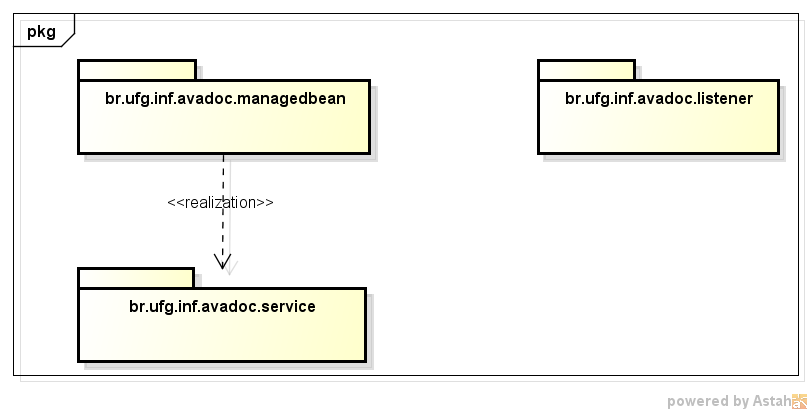


Figura 2 - representação em pacotes do módulo avadoc-web

## Pacotes de Design

***avadoc-server***

Abaixo estão listados os principais pacotes presentes no módulo avadoc-server.

* br.ufg.inf.avadoc.entity: pacotes com a representação em objetos de entidades que necessitam de alguma forma de persistência;
* br.ufg.inf.avadoc.model: classes e enumerados contendo regras do modelo de negócio, assim como objetos de dados criados dinamicamente para cálculos e o que mais for necessário.
* br.ufg.inf.avadoc.repository: classes que encapsulam o acesso aos dados persistidos, independente da forma de assistência.
* br.ufg.inf.avadoc.service: contém interface para acesso a serviços a serem consumidos pelos clientes. É através de serviços que os objetos podem ser persistidos e que cálculo são efetuados

***avadoc-web***

Abaixo estão listados os principais pacotes presentes no módulo avadoc-server.

* br.ufg.inf.avadoc.managedbean: neste pacote estarão classes que permitem o fluxo de navegação na aplicação web cliente, assim como interação com o usuário fornecendo interface gráfica para execução das atividades previstas nos requisitos de software. Sendo o cliente web implementado utilizando JSF, um modelo ***component based***, estado volátil de objeto é gerenciado também aqui.

# Visão de Negócio

Para fácil entendimento, a divisão em alto nível arquitetural adotada no projeto é a representada na imagem abaixo:

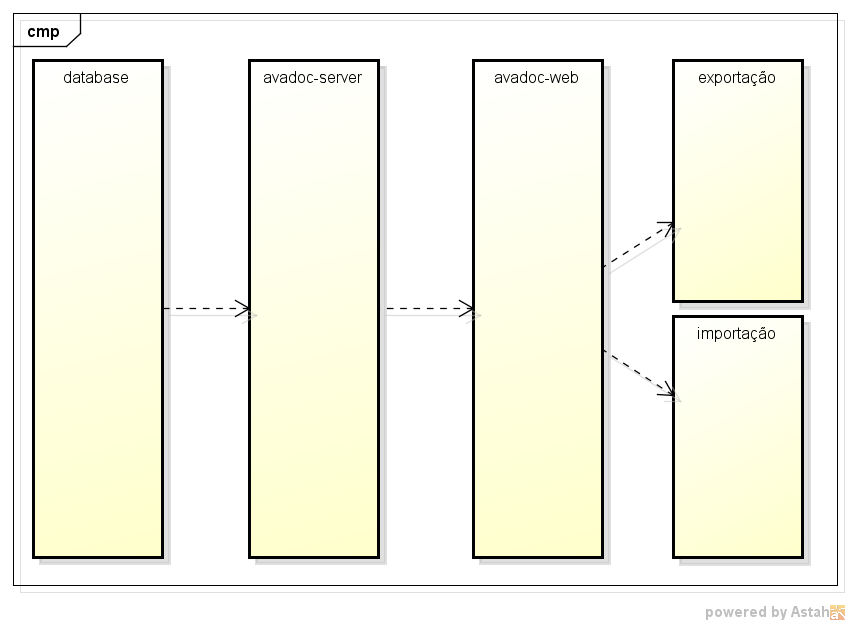


Figura 3 - visualização em componentes para entendimento do negócio

Aqui, tem grandes arcabouços que constituem componentes que, em alguns casos, representam algum valor de negócio. Os componentes podem ser resumidos conforme o que segue:

* database: repositório de dados com estados persistíveis;
* avadoc-server: motor de regras de negócio e de abstração de conceitos, como acesso a dados e regras de negócio;
* avadoc-web: cliente com interface web para interação com o usuário;
* exportação: módulo que, após cálculos do docente, exporta informações em formato padronizado para integração com terceiros;
* importação: realiza a importação de dados padronizados, concebendo integração entre aplicações no âmbito do escopo do projeto.

# Visão de Implementação

## Visão Geral

Por se de fácil domínio e de escopo reduzido, acordou-se que a arquitetura seria simples, facilitando uma possível manutenção e reduzindo o esforço para sua implementação.

Todos os componentes estarão em um container web, este último podendo ser parte de um ***application server***.

Os componentes do software construindo podem ser identificados em quatro grandes grupos:

* GenericManagedBean: interface base para managed beans JSF para interação com o usuário, através de aplicativo web;
* GenericService: interface base para implementação de serviços a serem utilizados por clientes;
* GenericRepository: interface base para manipulação de dados persistíveis;
* GenericModel: interface base para uso de serviços, contendo métodos para cálculos e o que mais for necessário, ou seja, contém as regras de negócio que devem ser aplicadas sobre os dados.

## Camadas

O software foi divido em quatro camadas, tendo duas delas, com conceitos diferentes, ocupando o mesmo nível de hierarquia. A imagem abaixo representa as camadas:

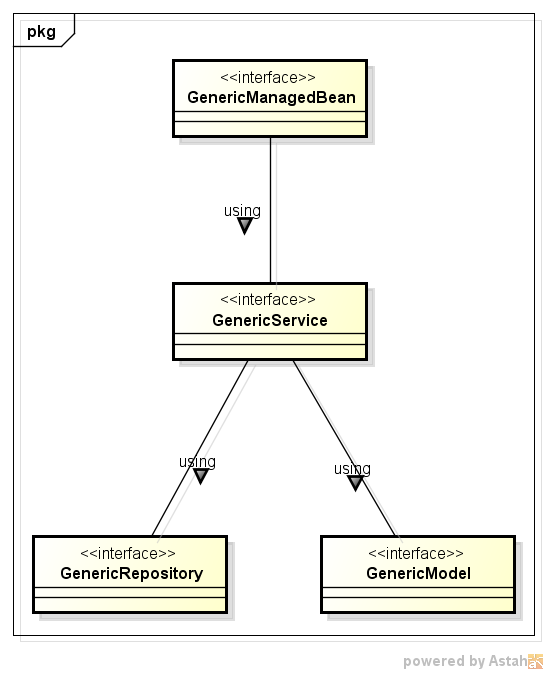


Figura 4 - camadas do software de acordo com classes genéricas