# Documento de Arquitetura WebDep

Sumário

[Documento de Arquitetura WebDep 1](#_Toc480464426)

[Versionamento 2](#_Toc480464427)

[Introdução 2](#_Toc480464428)

[Arquitetura Básica 2](#_Toc480464429)

[Diagrama de Pacotes 2](#_Toc480464430)

[Responsabilidades das Classes dos Pacotes 2](#_Toc480464431)

[Fluxo típico de trabalho 3](#_Toc480464432)

[Padronização de nomes no código-fonte 4](#_Toc480464433)

[Funções Especiais 4](#_Toc480464434)

[Gráficos 4](#_Toc480464435)

[Leitura de Logs 5](#_Toc480464436)

[Execuções Periódicas 5](#_Toc480464437)

[Envio de E-mail 5](#_Toc480464438)

[Configuração Inicial do Sistema 5](#_Toc480464439)

[Funções Estatísticas 5](#_Toc480464440)

[Alertas on-line 6](#_Toc480464441)

## Versionamento

|  |  |  |  |
| --- | --- | --- | --- |
| Descrição das Alterações | Versão | Data | Autores |
| Versão inicial do documento. | 1.0 | 19/04/2017 | Diogo Silveira Mendonça |

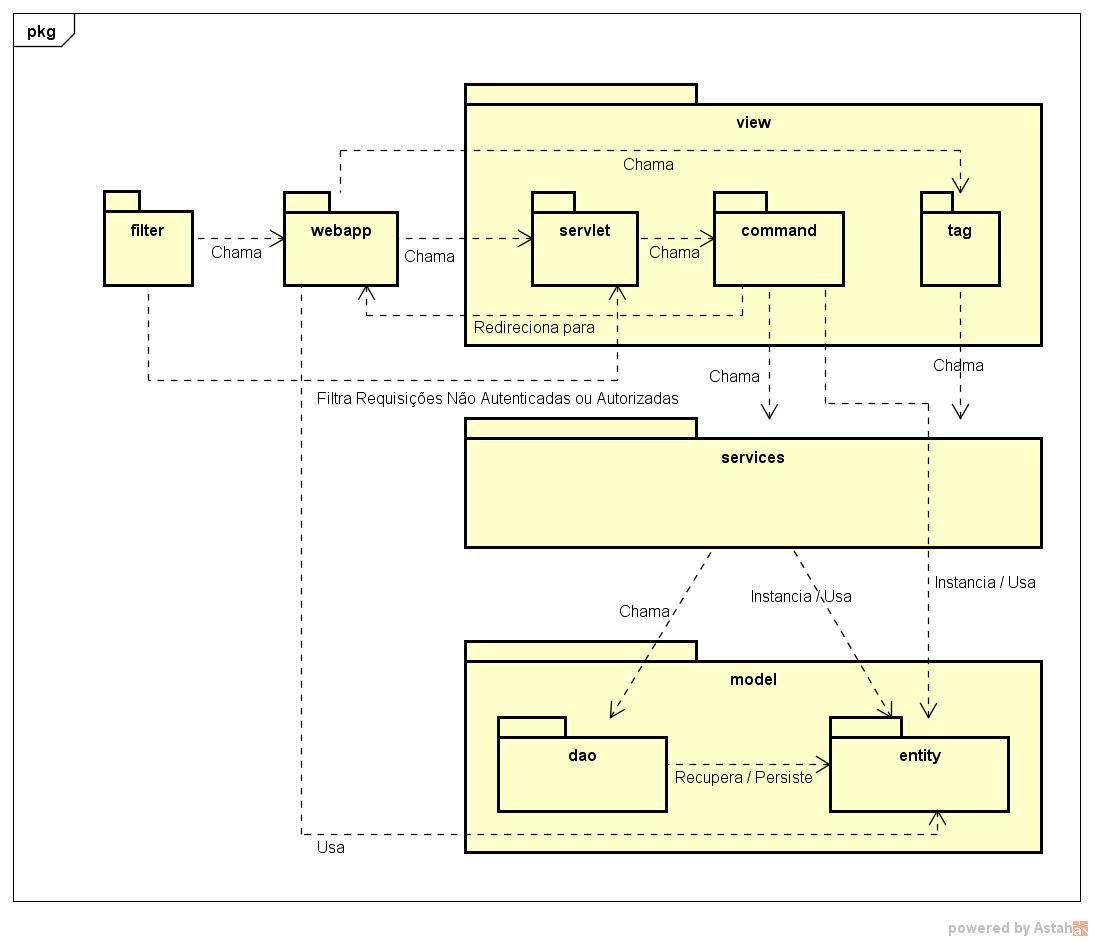
## Introdução

Este documento tem como objetivo estabelecer a arquitetura do sistema WebDep. Nele estão contidos os diversos padrões e estruturas que são adotadas em seu desenvolvimento.

## Arquitetura Básica

Esta seção descreve a estrutura básica dos pacotes, seus relacionamentos e responsabilidades, assim como os cenários típicos de utilização da aplicação.

### Diagrama de Pacotes

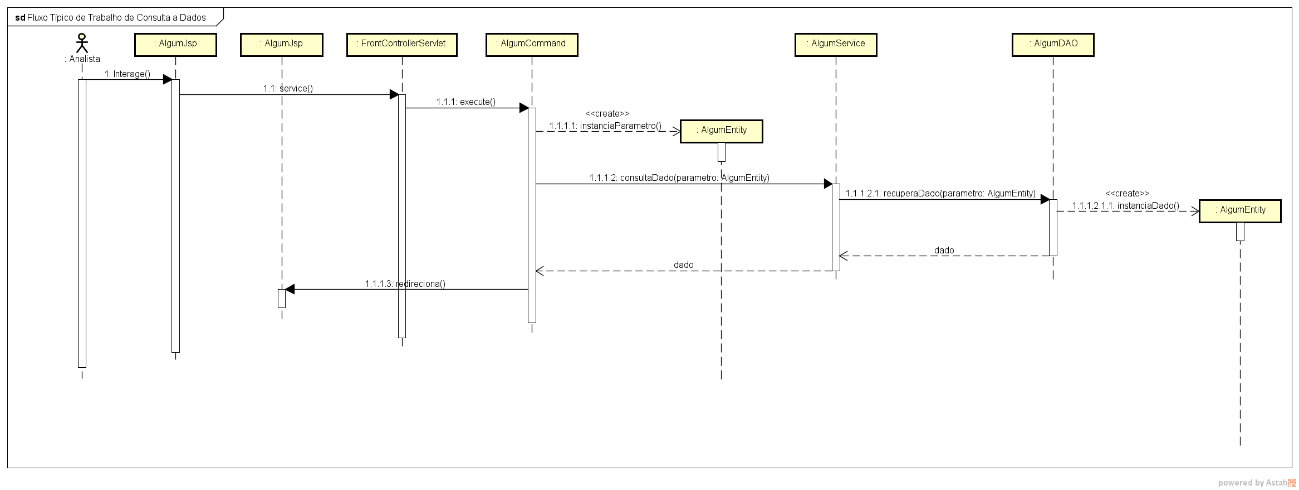


### Responsabilidades das Classes dos Pacotes

|  |  |  |
| --- | --- | --- |
| Pacote | Classes Pertencentes | Responsabilidades |
| filter | FiltroLogin | Interceptar requisições a todos os servlets e bloquear as requisições em que o usuário não esteja autenticado ou autorizado. |
| webapp | JSPs | Apresentação de dados e interfaces com o usuário. |
| view.servlet | FrontControllerServlet | Receber requisições de usuários e repassar para os comandos, realizar tratamentos comuns para todas as requisições como tratamento de exceções e geração de logs. |
| view.comand | Comandos (design pattern Command) | Realizar tratamento de requisições específicas do usuário. Cada comando trata exatamente um tipo de requisição, realizando validações, limpeza de dados e repassando as informações necessárias para os serviços e por fim redireciona as requisições para JSPs. |
| view.tag | Tags customizadas. Combos. | Implementação de componentes de interface de usuários que precisam de dados do banco para sua apresentação. |
| services | Serviços | Implementação de regras de negócio da aplicação. Podem fazer uso do controle de transações para tal. |
| model.entity | Entidades | Representam dados do domínio da aplicação. |
| model.dao | Daos Específicos, DAO Genérico, PersistenceManager | DAOs são responsáveis por abstrair o acesso a dados para outras camadas, portanto todo o acesso seja de recuperação ou salvamento de dados deve passar por um DAO. O DAO Genérico provê operações básicas para todas as entidades sem customização. Caso sejam necessárias customizações nas operações devem ser construídos DAOs específicos. O PersistenceManager provê métodos para controle das transações. |

### Fluxo típico de trabalho

O fluxo típico ampliado pode ser consultado arquivo de modelagem do sistema.



## Padronização de nomes no código-fonte

O código-fonte da aplicação deve ser padronizado seguindo padrão descrito nesta seção. Todos os nomes definidos no código-fonte devem seguir o padrão Java de programação, ou seja, capitalização Cammel Cased. A menos de alguns nomes que estão padronizados em Inglês todos os outros nomes da aplicação devem ser escritos em português.

* Nomes de Classes – Utilizar um sufixo que represente que tipo de classe ela representa
  + Comandos – Command
  + Servlets – Servlet
  + Serviços – Services
  + Tags – Tag
  + Filtros – Filter
  + DAOs – DAO
* Nomes de métodos e variáveis – Deve seguir padrão Java de capitalização e devem ser em português.

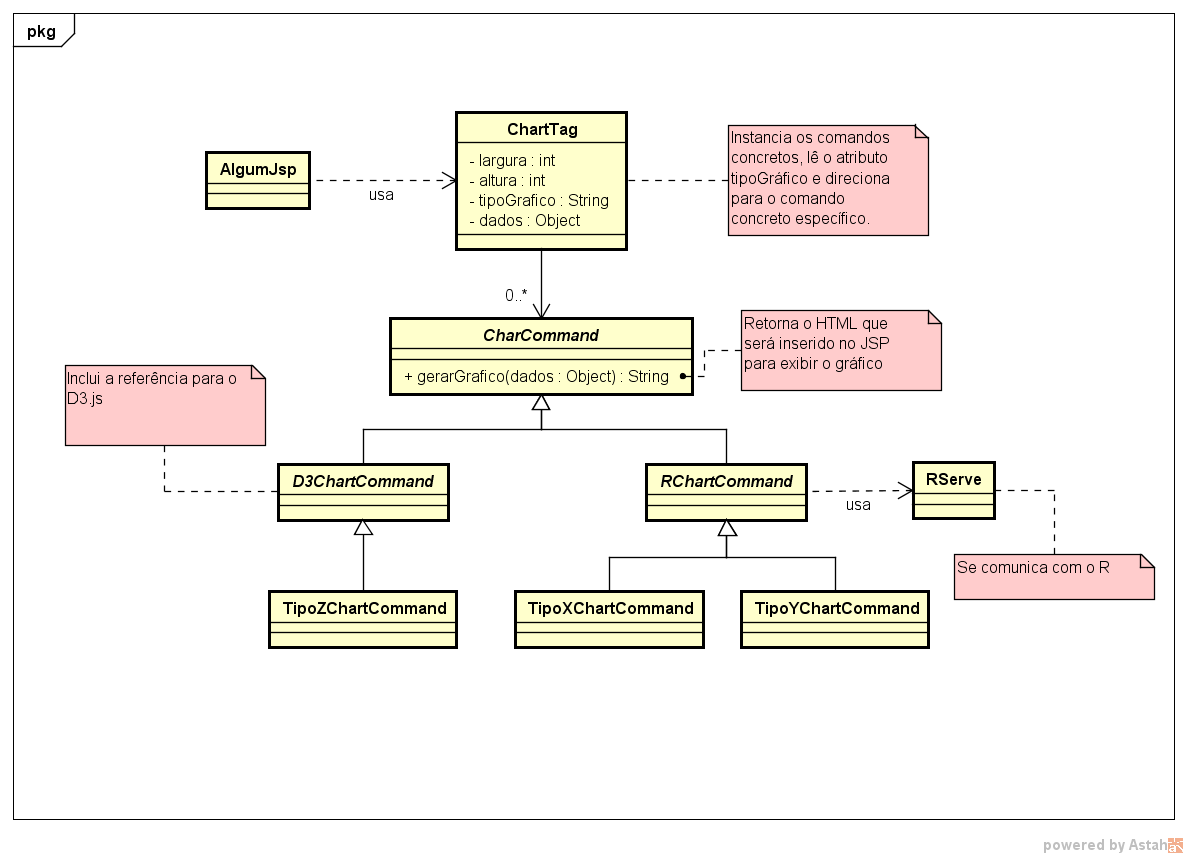
## Funções Especiais

### Gráficos

Para exibição dos gráficos serão utilizadas as bibliotecas D3.js e o R dependendo do gráfico que precisa ser gerado. As opções de bibliotecas gráficas que foram levadas em consideração foram o D3.js, JFreeChart e gráficos gerados na linguagem R. A decisão por não utilizar o JFreeChart teve como principal fator de influência a documentação ser paga e o aspectos visuais dos gráficos gerados não ser bom. Os gráficos serão gerados utilizando D3.js quando for necessária interatividade do usuário com os gráficos. Para gráficos simples sem interatividade será utilizado o R. Esta decisão levou em consideração os aspectos abaixo:

* Aspecto visual. Dentre as bibliotecas analisadas a D3.js foi a que apresentou melhor aspecto visual dos gráficos. Seguido do R e posteriormente o JFreeChart.
* Trafego de dados em rede e carga de processamento servidor. Os gráficos gerados no D3.js são processados no lado cliente da aplicação, diminuindo assim a carga de trabalho no lado servidor e também o tráfego de dados em rede. As outras bibliotecas trabalham com o processamento de imagens no lado servidor e sua transferência via rede.
* Diversidade e possibilidade de criação de novos gráficos. O D3.js apresenta a maior diversidade de gráficos entre as três opções. Existe também a possibilidade de criação de novos gráficos através de sua API. Nas outras duas opções só existe a possibilidade de utilização dos gráficos existentes.
* Interação do usuário. O D3.js proporciona maiores possibilidades de iteração do usuário com os gráficos através do uso do SVG que permite que sejam incluídos links nos elementos gráficos. O JFreeChart permite interação através de links. O R não permite nenhum tipo de iteração do usuário com os gráficos.
* Dificuldade de programação e know-how. Maior facilidade de programação dos gráficos em R devido a sua geração se dar por simples chamadas de funções. O D3.js necessita de configurações feitas em JS que são mais complicadas que o uso do R. Já o JFreeChart toda a codificação dos gráficos é feita em Java, mas a documentação não está disponível sem custos.
* Portabilidade. Browsers nativamente suportam Javascript, portanto o uso do D3.js não adicionaria nenhuma dependência de infraestrutura no lado cliente ou servidor. A utilização do R como ferramenta de geração dos gráficos geraria uma dependência no lado servidor do R estar instalado. O JFreeChart é desenvolvido em Java e não adicionaria nenhuma dependência extra a menos da sua biblioteca.
* Licenças. O D3.js é distribuído sobre a licença BSD-3 que não impõe nenhuma restrição considerável ao seu uso, contudo cada exemplo de gráfico do D3.js tem sua própria licença de uso. O R é distribuído sobre a licença GPL-2 o que restringe a sua distribuição somente a software que possua a licença GPL (reciproca). O JFreeChart é distribuído sobre a licença LGPL, que não impõe nenhuma restrição considerável ao seu uso, contudo a documentação do JFreeChart é paga e não está disponível sem custos.

#### Arquitetura da Solução



### Leitura de Logs

A preencher.

### Execuções Periódicas

A preencher.

### Envio de E-mail

A preencher.

### Configuração Inicial do Sistema

#### Arquivos de Propriedades

#### Arquivos de XML

A preencher.

### Funções Estatísticas

A preencher.

### Alertas on-line

A preencher.