UNIVERSIDADE FEDERAL DE SANTA CATARINA - UFSC
CENTRO TECNOLÓGICO - CTC
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA INE

RAFAEL BESEN

INFORMATIZAÇÃO E MODERNIZAÇÃO DE PROCESSOS DE PREFEITURAS UTILIZANDO SOFTWARE LIVRE

SISTEMA DE CONTROLE DE ALMOXARIFADO DE FARMÁCIAS PÚBLICAS

FLORIANÓPOLIS
2007

UNIVERSIDADE FEDERAL DE SANTA CATARINA - UFSC
CENTRO TECNOLÓGICO - CTC
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA INE

RAFAEL BESEN

INFORMATIZAÇÃO E MODERNIZAÇÃO DE PROCESSOS DE PREFEITURAS UTILIZANDO SOFTWARE LIVRE

SISTEMA DE CONTROLE DE ALMOXARIFADO DE FARMÁCIAS PÚBLICAS

Monografia apresentada à
Universidade Federal de Santa Catarina,
para a obtenção do título de
bacharel em sistemas de informação,
sob a orientação do
Prof. José Eduardo De Lucca.

FLORIANÓPOLIS
2007

UNIVERSIDADE FEDERAL DE SANTA CATARINA - UFSC
CENTRO TECNOLÓGICO - CTC
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA INE

RAFAEL BESEN

INFORMATIZAÇÃO E MODERNIZAÇÃO DE PROCESSOS DE PREFEITURAS UTILIZANDO SOFTWARE LIVRE

SISTEMA DE CONTROLE DE ALMOXARIFADO DE FARMÁCIAS PÚBLICAS

Esta monografia foi julgada adequada para obtenção do título de Bacharel em Sistema de Informação e aprovada pela Universidade Federal de Santa Catarina, Centro Tecnológico, Departamento de Informática e Estatística.

Florianópolis, 09 de Abril de 2007

_____

José Eduardo de Lucca
Universidade Federal de Santa Catarina
Orientador

_____

Rui Roberto de Souza Bergman
Centro de Informática e Automação de Santa Catarina

_____

Delson de Valois Santos
GeNESS

Agradecimentos

Agradeço a meu orientador, pelo auxílio durante o desenvolvimento do projeto.

À todos meus amigos que de alguma forma contribuíram para a conclusão do projeto.

E a minha família pelo apoio dado durante todo esse tempo.

Resumo

Este trabalho faz parte do projeto Via Digital, que visa sanar parte de um problema identificado na grande maioria das pequenas prefeituras do Brasil. Este problema trata-se da falta de investimento em infraestrutura e tecnologia, ocasionando um grande vão entre o governo e a população. Neste caso mais especificamente na área da saúde, farmácias públicas.

Baseado em um levantamento de requisitos, feito em uma pequena cidade do Rio Grande do Sul. Será desenvolvido um sistema para controlar o almoxarifado de farmácias públicas, utilizando tecnologias livres, para que estes sistemas sejam disponibilizados a estas prefeituras sem custo algum.

Com este projeto desejamos otimizar os estoques de medicamentos, evitando desperdícios e a falta destes nos estoques. Assim além de direcionar melhor os recursos, as prefeituras que adotarem o sistema, poderão prover um melhor atendimento ao público. Além disso este projeto também visa disseminar o uso do software livre no governo brasileiro.

Palavra-chave: Software Livre, Prefeituras, farmácias públicas, governo eletrônico, Via Digital

Abstract

This work is a part of Via Digital project, that aims to resolve part of a common problem in lot of brazilian city halls. This problem is about investment lack in infrastructure and technology, causing a gap between de government and population. In this in case that more specifically in the area of the health, public pharmacies.

Based in a survey of requirements, made in a little city of Rio Grande do Sul state. A computer system will be developed to control the public pharmacy warehouse, using free technologies, to get this system free of payment to that city halls.

With this project we desire to optimize medicine stocks, preventing wastefulness and lacks of these in supplies. Thus beyond best pointing the resources, the city halls that adopt this system, will be able to provide a best attendance to the public. Moreover this project aims to spread free software use at brazilian government.

Keywords: Open Source, City Hall, public pharmacies, e-gov, Via Digital

Lista de Figuras

Figura 1.....14
Figura 2.....15
Figura 3.....23
Figura 4.....23
Figura 5.....39
Figura 6.....40
Figura 7.....40
Figura 8.....45
Figura 9.....46
Figura 10.....47
Figura 11.....48
Figura 12.....48
Figura 13.....49
Figura 14.....50
Figura 12.....51
Figura 13.....51
Figura 14.....52

Lista de Abreviaturas

MVC – Model View Controller

JVM – Java Virtual Machine

SGBD – Sistema Gerenciador de Banco de Dados

DTO – Data Transfer Object

DAO – Data Access Object

XML – eXtensible Markup Language

CVS – Concurrent Version System

IDE – Integrated Development Environment

JSP – Java Server Pages

CRUD – Create, Read, Update e Delete

Sumário

1 Introdução..... 10
1.1 Justificativa..... 11
1.2 Objetivos..... 12
1.2.1 Objetivos Gerais..... 12
1.2.2 Objetivos Específicos..... 12
1.3 Materiais e Métodos..... 12
1.4 Organização do texto..... 12
2 Estudo de Ferramentas Existentes..... 13
2.1 Sistema SAUDETECH - OlosTech..... 13
2.2 Sistema HYGIA - Techne..... 14
2.3 Avaliações..... 16
3 Funcionalidades do sistema..... 16
3.1 Cadastros..... 17
3.2 Funcionalidades..... 18
3.3 Listagens/Relatórios..... 19
4 Tecnologias..... 20
4.1 Java..... 21
4.2 PostgreSQL..... 21
4.3 Arquitetura..... 22
4.3.1 MVC..... 22
4.3.2 DTO..... 23
4.3.3 DAO..... 23
4.3.4 Business..... 24
4.3.5 ActionSupport..... 24
4.4 Frameworks..... 24
4.4.1 Struts..... 24
4.4.2 Hibernate..... 25
4.4.3 Spring..... 25
4.5 CVS..... 25
4.6 Eclipse..... 26
4.7 IReport..... 26
4.8 TomCat..... 26
4.9 Ibatis..... 26
5 Arquitetura..... 27
5.1 Camada View..... 27
5.2 Camada Controller..... 29
5.3 Camada Model..... 30
5.3.1 Camada de Negócio..... 30
5.3.2 Camada de Persistência Hibernate..... 32
5.3.3 Camada de Persistência Ibatis..... 33
6 Desenvolvimento..... 36
6.1 Configurações..... 36
6.2 Paginas JSP..... 38
6.3 Actions..... 41
6.4 Classes de negócio..... 43
6.5 Classes de Persistência..... 44

6.6 Arquivos XML de Persistência....44
7 Sistema....45
7.1 Listagens....45
7.2 Cadastros....46
7.3 Visualizações....47
7.4 Funcionalidades....49
Conclusão....53
Bibliografia....54
ANEXOS....56
Anexo I - applicationContext.xml....56
Anexo II - struts.xml....66
ANEXO III - MedicamentoDAO.xml....74

1 Introdução

No ano de 2004 foi realizada, pela Softex, uma pesquisa sobre a aplicação de software livre em prefeituras, onde foi identificada a necessidade de uma biblioteca pública onde estivessem disponíveis informações sobre informatização de prefeituras. Então foi criado o Via Digital, que tem como objetivo a disponibilização de um portal que permita acesso e desenvolvimento compartilhado de software livre. Tendo como foco a gestão de pequenas prefeituras. O projeto visa através desse portal formar um catálogo de ferramentas de software e profissionais capacitados, viabilizando a informatização livre dessas prefeituras. [Via Digital]

Dentro do contexto do Via Digital, identifica-se como um dos pontos mais carentes a área da saúde, sendo necessário o desenvolvimento de sistemas na mesma. Sendo assim, no presente trabalho, optou-se por desenvolver um módulo de controle de almoxarifado de farmácias públicas. Este projeto foi denominado ViaFarmacos.

Todo o projeto é baseado em software livre, tecnologia que dá liberdade total ao usuário de usar, estudar, alterar e redistribuir o software, mas para que isso seja possível é imprescindível que se tenha acesso total ao código do software. A filosofia do software livre é basicamente liberdade de expressão, porém muitas pessoas associam software livre a software grátis, e em nenhum dos requisitos para um software ser livre consta este pré requisito, ou seja, independe de gratuidade[softwarelivre]. Existem diversas licenças de software livre, entre elas copyleft, GPL entre outras[softwarelivre1].

1.1 Justificativa

A maioria das pequenas prefeituras brasileiras apresenta sérios problemas em termos de recursos para investimento em infra-estrutural. Juntamente com a falta de informatização, a qualidade e quantidade de serviços prestados a comunidade tende a estagnar-se. Cada vez mais os serviços procurados pelo cidadão encontram-se mais aquém do esperado.

A implantação de ferramentas de governo eletrônico, baseados em soluções livres,

em âmbito municipal tende a aumentar e melhorar o contato entre cidadãos e governantes com relação às necessidades básicas, porém existe pouca ou quase nenhuma oferta de software adequado para suprir tais necessidades. Baseando-se nessas informações, pode-se concluir que, é necessário atuar para suprir as necessidades de desenvolvimento de software livre voltado para prefeituras, e estimular as mesmas de modo que adotem tal tecnologia.

A área da saúde é uma das mais importantes e ao mesmo tempo mais desorganizada, de posse dessas informações optou-se por desenvolver uma aplicação nesta área, de modo a minimizar gastos com software e desperdício da matéria prima por este setor utilizada, em prefeituras que optem por implantação do sistema proposto.

Muitos medicamentos acabam vencendo seu prazo de validade nos almoxarifados de prefeituras, ao mesmo tempo que outros remédios estão em falta. Otimizar esses estoques é uma boa forma para melhor atender a população e melhor direcionar os investimentos públicos, sendo assim atuar no desenvolvimento de um sistema para o almoxarifado de farmácias públicas é uma contribuição significativa para o setor.

1.2 Objetivos

1.2.1 Objetivos Gerais

- Disponibilizar um módulo de controle de almoxarifado de farmácias públicas de prefeituras que seja de código aberto, dando-lhes liberdade de escolher o sistema que eventualmente venham a usar.

1.2.2 Objetivos Específicos

- Controlar a entrada e saída de medicamentos dos almoxarifados das farmácias públicas, evitando o desperdício e a falta de medicamentos no estoque.

- Obter uma visão mais clara sobre a realidade das pequenas prefeituras do Brasil.

- Aperfeiçoar meus conhecimentos técnicos e sobre software livre.

- Incentivar a adoção de software livre por prefeituras que tiverem interesse no sistema.

1.3 Materiais e Métodos

O sistema será desenvolvido baseado em uma especificação de requisitos feita pela empresa SWFactory[swfactory] na prefeitura de Santa Clara do Sul no Rio Grande do Sul, e disponibilizada pelo projeto Via Digital[viadigital]. Os requisitos e casos de uso estáo disponíveis no site do projeto.[ViaFarmacos].

1.4 Organização do texto

Este documento está organizado da seguinte maneira:

- no capítulo 2 é realizado um breve estudo das tecnologias existentes;
- o capítulo 3 apresenta uma descrição das funcionalidades do módulo a ser desenvolvido;
- o capítulo 4 descreve as tecnologias envolvidas no desenvolvimento do sistema;
- no capitulo 5, será descrita a arquitetura do sistema do sistema proposto;
- no capítulo 6 estão os relatos do desenvolvimento do sistema;
- no capítulo 7, podemos ter uma visão melhor sobre o funcionamento do sistema.

2 Estudo de Ferramentas Existentes

Neste capítulo apresenta-se um breve estudo de algumas ferramentas relacionadas a almoxarifado de farmácias públicas existentes no mercado e suas funcionalidades.

Realizada uma busca por ferramentas que atendam os requisitos propostos no presente trabalho, foram encontradas apenas duas soluções: uma chamada SaudeTech da empresa OlosTech e outra chamada Hygia da empresa Techne. Provavelmente existam outras, mas estas duas representam o que o mercado atualmente oferece para prefeituras.

2.1 Sistema SAUDETECH - OlosTech

Figura 1 – Logo SaudeTech

fonte: http://www.olostech.com.br/Images/logoss1.jpg

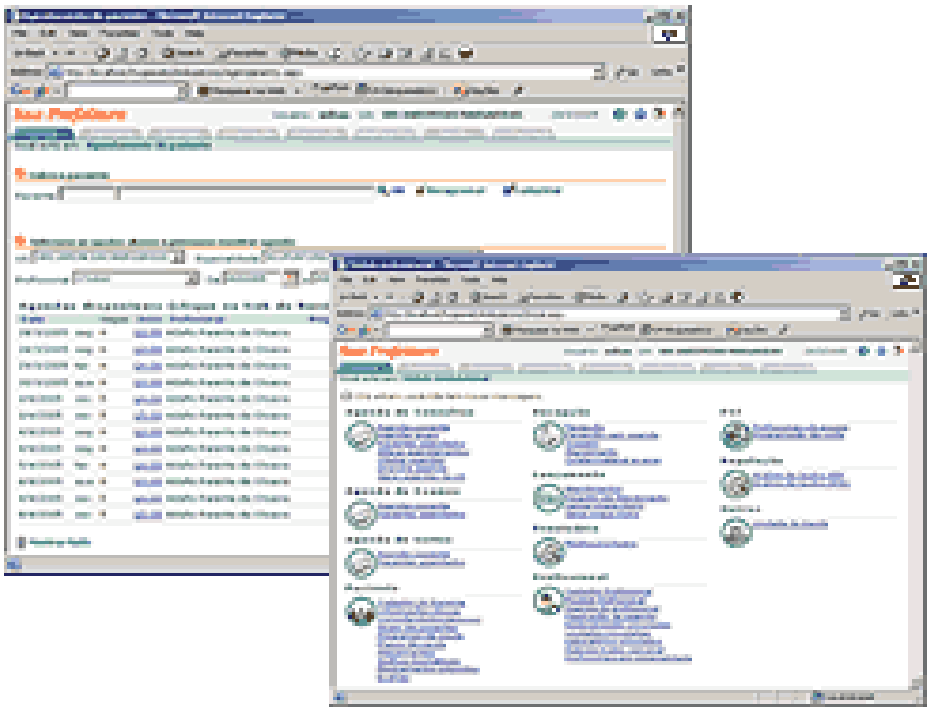
O SaudeTech é um sistema web já implantado em alguns municípios de Santa Catarina. É um sistema de gestão integrada de toda a área da saúde do município, envolvendo diversos temas, muito além das farmácias que são o foco deste trabalho.

Especificamente em relação ao módulo que trata de almoxarifado de farmácias a empresa apresenta as seguintes informações: “Manter níveis de estoque economicamente adequados e, principalmente, controlar a demanda de materiais por unidade e ainda, por centros de custo, é tarefa extremamente complexa quando sem assistência adequada. O Saudetech permite criar uma estrutura de controle extremamente eficaz em torno dessa questão. Disponibiliza requisição 'on line', com controle de acesso a pedido de reposição e autorização; estabelece controle de pericividade; permite incluir contas contábeis e exportar dados para contabilidade; disponibiliza um relatórios que permitem obter visão completa dos níveis de estoque, movimentação e demanda de materiais por centros de custo e unidades.” (fonte: http://www.olostech.com.br/saudetech/)

2.2 Sistema HYGIA - Techne

O Hygia, que apresenta a mesma idéia sistema anterior, funciona em ambiente web e integra a gestão de todos os pontos relacionados à saúde. É dividido em módulos, tendo um especificamente para farmácias públicas e almoxarifados. Aparentemente o sistema é extremamente robusto e mais voltado para grandes cidades com grandes números de registros.

Figura 2 – Sistema Hygia



retirado de : http://www.techne.com.br/images/hygia.gif

Sobre o ambiente de almoxarifado e farmácias públicas a empresa afirma o seguinte: "Gerencia os estoques locais e central, controlando os
medicamentos prescritos, ministrados e entregues ao paciente.

e indicação de similares.
O foco é no paciente, com lista única de medicamentos a
entregar disponível em todas as farmácias, facilitando a
entrega e minimizando o desperdício e a fraude.

Controle por código de barras e lote, facilitando a entrada e

Executa transferências entre farmácias. O aviso de final de estoque organiza a lista para a licitação.

”

(fonte:http://www.techne.com.br/produtos/folderhygia.pdf)

2.3 Avaliações

Ambos os sistemas podem atender diversas necessidades das prefeituras na área da saúde. Entretanto, é provável que muitas prefeituras não dispõem de recursos financeiros e humanos suficientes para a implantação de tais sistemas, em função da complexidade destes. Além disso nenhum dos dois é software livre, tornando o cliente dependente da empresa proprietária enquanto usar o sistema.

O ViaFarmacos visa não somente disponibilizar uma ferramenta para gestão integrada de almoxarifado de farmácias públicas, mas também permitir que as prefeituras tenham a opção de escolha, customização e integração com outros sistemas que venham a adotar, independentemente da empresa que contratarem para realizar alguma dessas tarefas, ou mesmo seus próprios funcionários. Esta situação é garantida pela adoção de software livre, visto que as prefeituras estariam de posse do código podendo alterá-lo da maneira que desejarem.

Este levantamento ainda se encontra em desenvolvimento e até a conclusão do presente trabalho pretende-se encontrar e avaliar outras soluções para a área, inclusive, se disponíveis, sistemas desenvolvidos com base na filosofia do software livre.

3 Funcionalidades do sistema

No presente capítulo estão descritas as funcionalidades do módulo a ser desenvolvido. Estas serão divididas em três grupos: Cadastros, Funcionalidades, e Listagens/Relatórios. Como já citado, os requisitos foram levantados na prefeitura de Santa Clara do Sul, dentro do projeto Via Digital, pela empresa SWFactory. Os itens apresentados abaixo são um resumo destes requisitos.

3.1 Cadastros

Cadastro de Medicamentos: Os usuários administrador, digitador ou responsável pelo almoxarifado terão acesso a inserir, alterar, remover e visualizar medicamentos, e na funcionalidade de visualizar além destes, os usuários responsável pela unidade consumidora e gestor também terão acesso. Com exceção de cadastrar medicamento as outra funções serão acessadas pela listagem de medicamentos.

Cadastro de Grupos de Medicamento: Os usuários administrador, digitador ou responsável pelo almoxarifado terão acesso a inserir, alterar, remover e visualizar grupos de medicamentos, e na funcionalidade de visualizar além destes, os usuários responsável pela unidade consumidora e gestor também terão acesso. Com exceção de cadastrar medicamento as outra funções serão acessadas pela listagem de grupos de medicamento. Os grupos básicos, não-básicos, controlados e contínuos serão cadastrados no momento da instalação do sistema.

Cadastro de Categoria de Entrada/Saída: O usuário administrador ou o próprio sistema farão a inserção de novas categorias. No momento da instalação do sistema são inseridas as categorias: compra, doação, implantação, ajuste de estoque, devolução, deterioração, encaminhados para vigilância sanitária, solicitação de medicamentos e distribuição ao usuário da unidade consumidora. Estas categorias não poderão ser alteradas ou removidas.

Cadastro de Localização: Os usuários administrador, digitador ou responsável pelo almoxarifado poderão inserir, alterar e remover localizações, estas são divididas em três tipos: externa, intermediária e interna. A localização externa é a menos detalhada, para localização

de endereço físico, um exemplo seria um corredor. A intermediária tem um nível de detalhamento médio, podemos citar uma prateleira como exemplo. Já a localização interna tem alto nível de detalhamento, por exemplo o andar da prateleira. Todas as localizações se relacionam hierarquicamente, de modo que uma localização interna pertence a uma intermediária que por sua vez faz parte de um externa, dessa forma uma localização completa ficaria da seguinte maneira: corredor x, prateleira y, andar z.

Cadastro de Unidade Consumidora: Os usuários administrador, digitador ou responsável pelo almoxarifado terão acesso a inserir, alterar, remover e visualizar unidades consumidoras, e na funcionalidade de visualizar além destes, os usuários responsável pela unidade consumidora e gestor também terão acesso. Com exceção de cadastrar unidade consumidora as outra funções serão acessadas pela listagem de unidades consumidoras. Para uma unidade consumidora (farmácia pública) ser cadastrada é necessário que seja cadastrado previamente no sistema um usuário com o perfil "Responsável pela Unidade Consumidora".

Cadastro de Fornecedores: Os usuários administrador, digitador ou responsável pelo almoxarifado terão acesso a inserir, alterar, remover e visualizar fornecedores, e na funcionalidade de visualizar além destes, os usuários responsável pela unidade consumidora e gestor também terão acesso. Com exceção de cadastrar fornecedor as outra funções serão acessadas pela listagem de fornecedores.

3.2 Funcionalidades

Informar entrada de medicamentos no almoxarifado: Os usuários administrador e digitador tem acesso a esta funcionalidade. Essa funcionalidade deve ser acessada quando o usuário desejar informar uma entrada de medicamentos no almoxarifado, todas as entradas informadas são reconhecidas pelo sistema como pendentes.

Confirmar entrada de medicamentos no almoxarifado: Os usuários administrador e responsável pelo almoxarifado tem acesso a essa funcionalidade. Essa funcionalidade será acessada quando o usuário desejar confirmar ou rejeitar uma entrada de medicamentos, e será acessada pela listagem de entradas pendentes de medicamentos. Caso a entrada seja rejeitada ela permanece no banco de dados apenas para constar no histórico do sistema, já as entradas confirmadas o estoque é automaticamente atualizado.

Alocar a entrada de medicamentos no almoxarifado: Os usuários administrador e responsável pelo almoxarifado tem acesso a essa funcionalidade. Essa funcionalidade será acessada quando o usuário desejar alocar uma entrada de medicamentos, e será acessada pela listagem de entradas confirmadas de medicamentos.

Solicitar a saída de medicamentos no almoxarifado: Os usuários administrador, responsável pela unidade consumidora e responsável pelo almoxarifado tem acesso a essa funcionalidade. Essa funcionalidade pode ser acessada pelo usuário responsável pela unidade consumidora com a finalidade de abastecer o estoque da unidade, esta solicitação deve ser da categoria "Solicitação de Medicamentos", ou pode ser acessada pelo usuário administrador do almoxarifado por qualquer outro motivo, como deterioração, devolução, etc.

Autorizar saída de medicamentos do almoxarifado: Os usuários administrador e gestor tem acesso a essa funcionalidade. Quando o usuário deseja autorizar uma saída de medicamentos ele acessa a listagem de saídas de medicamentos pendentes, ou ele pode rejeitar e informar uma justificativa. Quando uma saída é autorizada o sistema deve classificar esses medicamentos como medicamentos com saída autorizada.

Confirmar saída de medicamentos no almoxarifado: Os usuários administrador e responsável pelo almoxarifado tem acesso a essa funcionalidade. Essa funcionalidade começa quando o usuário desejar confirmar uma saída de medicamentos, para uma unidade consumidora ou algum outro destino qualquer, e será acessada pela listagem de saídas autorizadas de medicamentos. Caso a saída seja rejeitada o usuário pode colocar uma justificativa, já as saídas confirmadas o estoque é automaticamente atualizado diminuindo a quantidade retirada.

Verificar automaticamente os medicamentos que atingiram estoque mínimo no almoxarifado: Quando um medicamento atinge a quantidade mínima em estoque o sistema deve avisar ao usuário gestor ou responsável pelo almoxarifado assim que este efetuar o login no sistema, o usuário por sua vez deverá confirmar que recebeu a informação, para que seja mantido em histórico.

3.3 Listagens/Relatórios

Abaixo temos a lista das listagens necessárias para o sistema, que poderão ser

impressas em forma de relatório ou exportadas em arquivos de tipo “pdf”.

Relatório/Listagens de Fornecedores;

Relatório/Listagens de Unidades Consumidoras;

Relatório/Listagens das Entradas Pendentes de Medicamentos;

Relatório/Listagens das Entradas Confirmadas de Medicamentos;

Relatório/Listagens das Entradas Alocadas de Medicamentos;

Relatório/Listagens das Saídas Pendentes de Medicamentos (Por Unidades Consumidoras e Grupos de Medicamentos);

Relatório/Listagens das Saídas Autorizadas de Medicamentos (Por Unidades Consumidoras e Grupos de Medicamentos);

Relatório/Listagens das Saídas Confirmadas de Medicamentos (Por Unidades Consumidoras e Grupos de Medicamentos);

Relatório/Listagens dos Medicamentos do Estoque do Almoxarifado (Por Grupos de Medicamentos);

Relatório/Listagens dos Medicamento do Almoxarifado que Atingiram Estoque Mínimo;

Relatório/Listagens dos Medicamentos do Almoxarifado que estão para Expirar o Prazo de Validade (Especificando o Número de Dias);

Relatório/Listagens dos Medicamentos do Almoxarifado que Ultrapassam o Prazo de Validade;

Relatório/Listagens dos Medicamentos Encaminhados para a Vigilância Sanitária Municipal pelo Almoxarifado;

4 Tecnologias

As tecnologias a serem utilizadas são todas livres e seguem alguns requisitos definidos pelo projeto Via Digital. Como linguagem de programação decidimos por Java[java], tornando assim o sistema independente de plataforma. Struts[struts] que é um framework Java para a camada de apresentação do sistema, na camada de negócio será utilizado o Spring framework[spring] para instanciação de objetos e controle de transações, e para persistência o Hibernate[hibernate], fazendo o mapeamento objeto relacional. O banco de dados será o PostgreSQL[postgresql] por ser livre e bastante robusto.

A utilização destas tecnologias sugere a utilização da arquitetura MVC [sun](Model-View-Controller), que separa a camada de apresentação, a camada de negócios e a camada de persistência, facilitando a manutenção e customização do sistema.

O sistema utilizará o ambiente web, assim qualquer computador com um browser e que esteja conectado à rede poderá acessá-lo, evitando que seja necessário que as prefeituras atualizem o hardware que já possuem, a única exigência seria um servidor para rodar o servidor de aplicação e o servidor de banco de dados.

4.1 Java

Java é atualmente a linguagem mais utilizada para desenvolvimento de sistemas. Uma linguagem que utiliza o paradigma de programação orientado à objetos, desenvolvida pela *Sun Microsystems* na década de 90. Uma das suas principais características é que, ao invés de ser compilada para código nativo da máquina, ela é traduzida para um código intermediário, chamado *bytecode* e interpretado por uma máquina virtual. Esta é conhecida como *JVM* e torna código criado independente de plataforma. Uma vez escrito um programa, não é necessário que se faça nenhuma alteração para que ele possa ser executado em qualquer plataforma, desde que esta possua uma versão da *JVM*. A versão utilizada será a 6.0, atualmente a ultima versão liberada pela *Sun*.[sun]

4.2 PostgreSQL

PostgreSQL é um sistema gerenciador de banco de dados(SGBD) relacional de código aberto, que conta com os mais avançados recursos. Atualmente um dos mais utilizados bancos de dados para aplicações de pequeno e médio porte. O PostgreSQL foi desenvolvido pela Universidade de Berkeley na Califórnia, a qual abandonou o projeto na sua quarta versão, foi quando a comunidade de software livre assumiu a continuação de seu desenvolvimento. Atualmente o desenvolvimento do sistema é coordenado pelo *PostgreSQL Global Development Group,* e patrocinado por diversas organizações de todo o mundo. [postgresql]

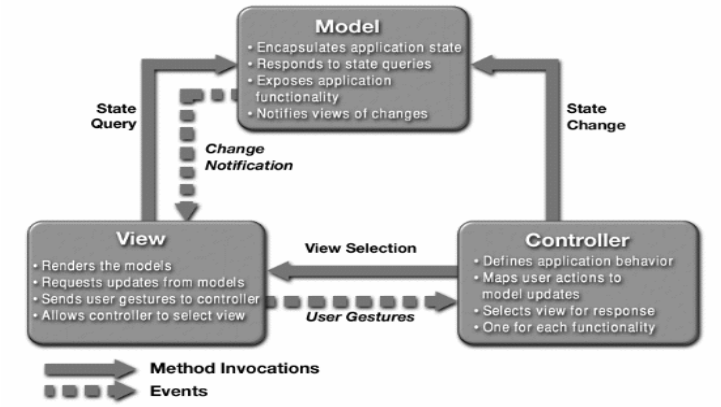
4.3 Arquitetura

A arquitetura do sistema será baseada em camadas para facilitar a implementação e manutenção do mesmo.

4.3.1 MVC

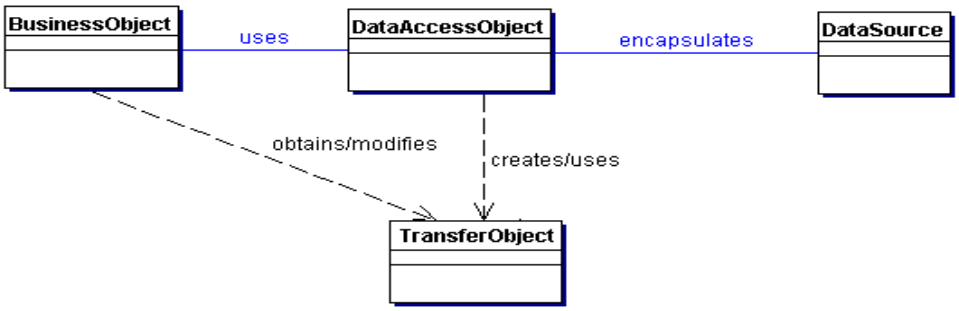
MVC(Model-View-Controller) é um padrão que visa separar as camadas do software em camada de apresentação, camada de controle e camada de lógica da aplicação. A camada lógica ainda pode ser dividida em regras de negócio e persistência. Para implementação desse modelo serão utilizados alguns padrões, sugeridos pela própria *Sun*, que abaixo estão descritos. [sun]

Figura 3 – Arquitetura MVC



retirado de http://java.sun.com/blueprints/patterns/images/mvc-structure-generic.gif

Figura 4 – Padrões Utilizados Para Arquitetura MVC



retirado de http://java.sun.com/blueprints/patterns/images/mvc-structure-generic.gif

4.3.2 DTO

DTO(Data Transfer Object) tem a função de encapsular dados de negócio. Criado o objeto ele é preenchido com os respectivos dados referentes a sua função, e transferido como parâmetro na chamada de métodos. Dessa maneira, além de encapsular os dados, diminuímos o número de parâmetros, conseqüentemente diminuindo também o acoplamento.[sun]

4.3.3 DAO

DAO(Data Access Object) este padrão prove a estrutura necessária para que a aplicação se comunique com a fonte de dados, encapsulando esta, de forma que se torne transparente para a aplicação a maneira com que os dados estão sendo persistidos ou consultados. A camada DAO comunica-se com a camada Business, descrita logo abaixo, permitindo assim que os dados sejam processados e apresentados para o usuário, ou persistidos.[sun]

4.3.4 Business

Business é utilizado para organizar a camada intermediária da aplicação, ele recebe os dados da camada de controle da aplicação e repassa para seus respectivos DAO, além disso esse padrão ajuda a diminuir o acoplamento, permitindo que uma camada seja alterada sem afetar outra diretamente.[sun]

4.3.5 ActionSupport

ActionSupport é o controlador da aplicação, responsável por tratar os dados da camada de apresentação, quando o usuário envia dados para a aplicação ou quanto quando a aplicação é requisitada para disponibilizar dados ao usuário. A validação dos dados também fica sob responsabilidade desta camada.[action]

4.4 Frameworks

Para garantir a utilização correta da arquitetura e dos padrões acima descritos, vamos nos apoiar em alguns *frameworks* de desenvolvimento.

4.4.1 Struts

Para a camada de controle de fluxo e de apresentação será utilizado o *framework* Struts 2.0. Este implementa o padrão *ActionSupport* que é responsável pelo controle do fluxo de aplicações web. O Struts possui também algumas *taglibs* para serem utilizadas na parte de apresentação. Estas por sua vez são responsáveis por montar as telas, preencher e capturar o conteúdo da pagina repassando essas informações para as *Actions*. Toda sua configuração é feita em arquivos *XML*(eXtensible Markup Language) excluindo assim as cofiguração do código.[struts]

4.4.2 Hibernate

Na camada de persistência o *framework* Hibernate tratará o mapeamento objeto-relacional, de forma que a conexão com o banco de dados e o mapeamento das tabelas para objetos se dará de forma automatizada, facilitando e agilizando o desenvolvimento do sistema, tendo em vista que uma parte considerável do tempo de desenvolvimento de um projeto é gasto em consultas e tratamento de transações. toda a configuração do *framework* e o mapeamento objeto-relacional é feita em arquivos *XML* tornando-os mais legíveis ao desenvolvedor.[hibernate]

4.4.3 Spring

O Spring trata a instanciação de objetos, de forma que os objetos da aplicação sejam instanciados apenas uma vez, evitando o uso desnecessário de memória, cuida da passagem de parâmetros na instanciação das classes diminuindo assim o acoplamento das camadas do sistema, o gerenciamento de transações também fica sob responsabilidade deste *framework*, além de integrar-se ao Hibernate e ao Struts de forma que os três funcionem de forma sincronizada comunicando-se, e garantindo assim um melhor funcionamento do sistema como um todo.[spring]

4.5 CVS

Será utilizada a ferramenta *CVS*(Concurrent Version System) para o controle de versões, é uma ferramenta de código aberto que facilita muito o desenvolvimento de aplicações. Ele utiliza uma arquitetura cliente-servidor podendo ser acessado remotamente e mantém um log de todas as alterações feitas em cada arquivo.[cvs]

4.6 Eclipse

A ferramenta para desenvolvimento que será utilizada será o eclipse, uma IDE desenvolvida em Java, de código aberto, e bastante flexível, possibilitando a instalação de vários plugins, dando então a ferramenta possibilidade de suporte automatizado a grande parte da estrutura que será utilizada no projeto, inclusive a ferramenta para controle de versão. [eclipse]

4.7 IReport

A ferramenta iReport é um editor gráfico de código aberto, para relatórios gerados em Java a partir da biblioteca *Jasper Report*, um modelo é criado utilizando a ferramenta. a qual gera um arquivo *XML* com o modelo do relatório, a *Jasper Report* provê a ligação entre a aplicação e o modelo, inicializando o relatório no formato desejado, entre os formatos possíveis estão arquivos *pdf* e *html*, e envia os parâmetros necessários ao modelo para a geração do documento.[ireport]

4.8 TomCat

TomCat trata-se de um container para Servlets, tecnologia necessária para implementação de Servlets e JSPs. Desenvolvido em Java, disponibilizado pela fundação Apache e adotado pela SUN como referência para desenvolvimento java-web. Possui suporte

a API J2EE, sendo possível fazer a conexão com o datasource através dele e até mesmo tratar requisitos de segurança. É recomendado utilizá-lo em conjunto com o servidor http dedicado, como o apache, por questões técnicas.[tomcat]

4.9 Ibatis

Ibatis trata-se de um framework para persistência de dados, assim como o hibernate, porém o mapeamento do bando relacional não precisa ser feito, ao invés disso são feitas as consultas SQL normalmente, e colocadas em um arquivo xml. Este framework tem versões para Java, .NET e Ruby. Esse projeto foi iniciado em 2001 e seu objetivo inicial era o desenvolvimento de aplicações criptográficas, mas acabou mudando seu rumo.

5 Arquitetura

Neste capítulo podemos observar como a aplicação foi estruturada e a função exata de cada camada.

5.1 Camada View

A camada view utiliza páginas jsp para interação com o usuário e interage com a camada controller através do framework Struts.

Estas páginas são montadas dinamicamente pela aplicação. Para construção dessas páginas são utilizadas *tags* do Struts, evitando ao máximo a existência de linhas de código na página, tornando-as mais compreensíveis e facilitando sua manutenção.

Abaixo podemos observar um exemplo de página utilizando html e linhas de código:

```
<% Usuario usuario = ActionContext.getContext() %>
<form action="alterarSenha.action" method="post">
<table>
      <tr>
            <td> align="right"<label>Senha:</label></td>
            <td><input type="password" name="usuario.senha"/></td>
      </tr>
      <tr>
            <td> align="right"<label>Confirmação:</label></td>
            <td><input type="password" name="usuario.senha2"/></td>
      </tr>
      <tr>
            <td><input type="submit" value="Ok"/></td>
      </tr>
</table>
</form>
```

Abaixo temos a mesma página feita utilizando tags Struts:

```
<s:actionerror/>
<s:form action="alterarSenha”  validate="true">
<s:password label="Senha" name="senha"/>
<s:password label="Confirmação" name="senha2"/>
<s:submit value="Ok" name="Ok"/>
</s:form>
```

Comparando os dois exemplos acima podemos observar que a segunda opção é

consideravelmente mais simples, tornando assim o código mais compreensível e o desenvolvimento mais rápido.

A comunicação dessas páginas com a camada controller se dá através do mapeamento destas para as *Actionsuppots*, este mapeamento é feito por um arquivo *xml* de configuração do Struts, o nome desse arquivo é "*viafarmacos-support.xml"*. Em cada operação realizada por uma action temos um texto (String) como retorno. Dependendo desse texto o usuário é direcionado para uma página. Abaixo temos um exemplo de configuração.

```
<action name="sample"
      class="br.org.viadigital.viafarmacos.web.action.SampleAction">

      <result name="cadastrar">/jsp/cadastrar.jsp</result>
      <result name="login">/jsp/login.jsp</result>

</action>
```

O texto que a action retorna é relativo ao atributo "name" da tag "result" e o texto da tag é o endereço da página para que o usuário será direcionado.

5.2 Camada Controller

A camada controller é formada basicamente pelas "Actions" classes que estendem a classe ActionSupport do Struts, fazem a comunicação da aplicação com as jsps. Cada módulo da aplicação possui uma Action e cada uma dessas possui um método para cada ação do módulo. Cada método executa suas respectivas funções retornando um texto com o resultado, como foi descrito no capítulo anterior. Além disso essas classes são responsáveis também por enviar as informações para as páginas exibi-las ao usuário, e também por capturar as informações vindas da páginas.

Nessa camada também é realizada a validação dos dados fornecidos pelo usuário, com isso podemos verificar se algum campo não foi preenchido ou foi preenchido com uma informação inválida. No caso de alguma inconsistência o papel da Action é voltar a página em que o erro foi cometido, e avisar ao usuário da inconsistência através de uma mensagem exibida na tela.

```
package br.org.viadigital.viafarmacos.web.action;
public class SampleAction extends AbstractAction {

	public String executeDefault() {

		if (false) {
			return "Erro";
		} else {
			return SUCCESS;
		}
	}

}
```

Todas as classes dessa camada são mapeadas no arquivo de configuração do Struts, assim como todos seus possíveis resultados.

```
<action name="sample"
        class="br.org.viadigital.viafarmacos.web.action.SampleAction">
            <result name="Erro">/jsp/error.jsp</result>
            <result name="Sucesso">/jsp/amplexo</result>
</action>
```

Todas as requisições realizadas são capturadas e verificadas antes de serem executadas, um exemplo de verificação que sempre é feita é se o usuário está conectado. Para realizar esses tipos de verificação são criados os chamados Interceptor, uma espécie de filtro que verifica cada requisição. Estes também são mapeados no arquivo de configuração do Struts. Esse tipo de classe facilita a implementação de um sistema mais seguro possível.

```
<interceptor name="authentication"
                    class="mailreader2.AuthenticationInterceptor"/>

<interceptor-stack name="user" >
      <interceptor-ref name="authentication" />
      <interceptor-ref name="defaultStack"/>
</interceptor-stack>
```

5.3 Camada Model

A camada model foi dividida em duas partes: a camada de negócio e a camada de persistência. Segue abaixo a descrição de cada uma delas.

5.3.1 Camada de Negócio

A camada de negócio trata das regras de negócio da aplicação e também faz a ligação da camada de controle com a camada de persistência, permitindo a implementação de recursividade, e assim contribuindo para o baixo acoplamento da aplicação.

Essa camada é constituída pelas classes do pacote *business*, cada ator do sistema possui uma dessas classes onde suas ações são representadas através de métodos. São instanciadas pelo Spring, para isso são mapeadas no arquivo de configuração "aplicationContext.xml", onde também recebem parâmetros e podemos ainda configurar o controle de transações e até mesmo o cache da aplicação.

Abaixo temos uma classe de negócio:

```
package br.org.viadigital.viafarmacos.model.business;
import br.org.viadigital.viafarmacos.exception.SystemException;
import br.org.viadigital.viafarmacos.model.DAO.UsuarioDAO;
import br.org.viadigital.viafarmacos.model.DTO.ProdutoDTO;
import br.org.viadigital.viafarmacos.model.DTO.UsuarioDTO;


public class Usuario extends AbstractBusiness {
	private UsuarioDAO usuarioDAO;

	public void setUsuarioDAO(UsuarioDAO usuarioDAO) {
		this.usuarioDAO = usuarioDAO;
	}
	public void cadastraProduto(UsuarioDTO usuarioDTO,
			ProdutoDTO produtoDTO) throws SystemException {
		try {
			usuarioDAO.cadastraProduto(usuarioDTO, produtoDTO);
		} catch (Exception e) {
			throw new SystemException(this.emailSupport,
			this.logger, this.mail, e);
		}
	}
}
```

Abaixo podemos observar um exemplo de configuração de uma classe business:

```
<bean id="usuario" parent="abstractBusiness" class=
"org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
	<property name="transactionManager" ref="transactionManager" />
		<property name="target" ref="usuarioTarget" />
		<property name="proxyTargetClass" value="true" />
		<property name="transactionAttributes">
			<props>
```

```
                        <prop key="cadastraProduto">
                              PROPAGATION_REQUIRED,
                              ISOLATION_READ_UNCOMMITTED,-Exception
                        </prop>
                  </props>
            </property>
      </bean>

      <bean id="usuarioTarget" parent="abstractBusiness"
            class="br.org.viadigital.viafarmacos.model.business.Usuario">
            <property name="usuarioDAO">
                  <ref local="usuariodao" />
            </property>
      </bean>
```

5.3.2 Camada de Persistência Hibernate

A camada de persistência basicamente, encapsula o datasource e faz a comunicação entre este e a aplicação, neste sistema esta é feita através do hibernate. Este por sua vez mapeia as tabelas do banco de dados e as transforma em objetos, de modo que as consultas SQL deixem de ser necessárias.

Esta camada possui classes chamadas DAO, essas classes são acessadas pelas classes da camada de negócio. Existe um DAO para cada Business, abaixo podemos observar um exemplo da classes UsuarioDAO, que é acessada pelo UsuarioBusiness acima descrita.

```
package br.org.viadigital.viafarmacos.model.dao;
public class UsuarioDAO {

      public UsuarioDAO() {
      }

      private String nome;

      private int id;

      public int getId() {
            return id;
      }

      public void setId(int id) {
            this.id = id;
      }

      public String getNome() {
            return nome;
      }

      public void setNome(String nome) {
            this.nome = nome;
```

```
    }
}
```

Essas classes tem acesso a base de dados por meio de arquivos xml que mapeiam as tabelas e suas relações abaixo temos o exemplo do arquivo referente a essa classe:

```
<class   name="br.org.viadigital.viafarmacos.model.dao.UsuarioDAO"
                                                    table="USUARIO">
 <id name="id" column="ID_USUARIO">
     <generator class="native"/>
 </id>
 <property name="nome" column="NM_USUARIO"/>
</class>
```

Essas Classes também são mapeadas no arquivo de configuração "aplicationContext.xml", existindo as mesmas possibilidades de configuração que as classes de negócio. Abaixo podemos observar a classe UsuarioDAO mapeada neste arquivo.

```
<bean id="usuarioDAO" parent="abstractDAO" class=
                  "br.org.viadigital.viafarmacos.model.dao.UsuarioDAO">
      <property name="sessionFactory">
            <ref local="sessionFactory" />
      </property>
</bean>
```

5.3.3 Camada de Persistência Ibatis

A camada de persistência com Ibatis tem o mesmo conceito daquela na qual utiliza-se o hibernate, porém o ibatis é mais simples, e portanto mais adequado para um sistema desse porte, abaixo podemos observar como fica uma classe DAO utilizando Ibatis

```
package br.org.viadigital.viafarmacos.model.dao;

import java.util.List;

import org.springframework.orm.ibatis.support.SqlMapClientDaoSupport;

import br.org.viadigital.viafarmacos.model.dto.Grupo;

public class GrupoDAO extends SqlMapClientDaoSupport {

      public void insertGrupo(Grupo grupo) {

            super.getSqlMapClientTemplate().insert("insertGrupo", grupo);
```

```
	}

	public List selectGrupos() {

		return
super.getSqlMapClientTemplate().queryForList("selectGrupos");

	}

	public Grupo selectGrupo(Integer id) {

		return (Grupo) super.getSqlMapClientTemplate().queryForObject(
				"selectGrupo", id);

	}

	public void updateGrupo(Grupo grupo) {

		super.getSqlMapClientTemplate().update("updateGrupo", grupo);

	}

	public void removeGrupo(Integer id) {

		super.getSqlMapClientTemplate().update("removeGrupo", id);

	}

}
```

Esta classe ao invocar algum de seus métodos, executa um comando SQL localizado em algum dos arquivos xml mapeados, o comando a ser executado é identificado por um id, o que é passado como parâmetro na invocação do método, assim como o objeto com os dados a serem utilizados para a consulta. Abaixo temos o xml referente a classe mostrada acima:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap PUBLIC "-//iBATIS.com//DTD SQL Map 2.0//EN"
"http://www.ibatis.com/dtd/sql-map-2.dtd">
<sqlMap namespace="GrupoDAO">

	<typeAlias alias="grupo"
type="br.org.viadigital.viafarmacos.model.dto.Grupo"/>
	<typeAlias alias="int" type="java.lang.Integer"/>

	<resultMap id="result-grupo" class="grupo">
		<result property="id" column="codigo"/>
		<result property="nome" column="nome"/>
		<result property="descricao" column="descricao"/>
	</resultMap>


	<select id="selectGrupo" resultMap="result-grupo"
parameterClass="int" resultClass="grupo">
		select
```

```
                codigo, nome, descricao
            from
                grupo_medicamento
            where
                codigo=#id#
        </select>

        <select id="selectGrupos" resultMap="result-grupo"
parameterClass="grupo" resultClass="grupo">
            select
                codigo, nome, descricao
            from
                grupo_medicamento
            where
                inativo = false
        </select>

        <select id="selectMaxGrupo" resultClass="int">
            select max(codigo) from grupo_medicamento
        </select>

        <update id="updateGrupo" parameterClass="grupo">
            update
                grupo_medicamento
            set
                descricao = #descricao#,
                nome = #nome#
            where
                codigo = #id#
        </update>

        <update id="removeGrupo" parameterClass="int">
            update
                grupo_medicamento
            set
                inativo = true,
            where
                codigo = #id#
        </update>

        <insert id="insertGrupo" parameterClass="grupo">
            insert into grupo_medicamento (
                codigo,
                nome,
                descricao
            ) values (
                (select 1+max(codigo) from grupo_medicamento),
                #nome#,
                #descricao#
            )
        </insert>

</sqlMap>
```

Assim como o hibernate o Ibatis também é integrado ao Spring, e as classes DAO ficam mapeadas no AplicationContext.xml da seguinte maneira:

```
<bean
id="grupoDAO"class="br.org.viadigital.viafarmacos.model.dao.GrupoDAO">
            <property name="dataSource">
                  <ref local="dataSource" />
            </property>
            <property name="sqlMapClient">
                  <ref local="sqlMapClient" />
            </property>
</bean>
```

6 Desenvolvimento

Neste capítulo encontra-se a descrição completa do desenvolvimento do sistema, divido por camadas. Assim como no capítulo 5, vamos começar com as configurações do sistema, e posteriormente vamos para a camada de visualização do usuário, passando por todas as camadas até chegarmos na camada de persistência.

6.1 Configurações

A aplicação possui diversos arquivos de configuração, vamos descrever cada um iniciando pelos arquivos .properties, os quais possuem propriedades que serão utilizadas pelo sistema. Essas propriedades são utilizadas nesse tipo de arquivo para que se necessário alguma alteração nestas o código não necessite ser recompilado.

A estrutura desses arquivos é a seguinte:

```
# Diretório de Imagens
app.pathImage=/home/img/
# Diretório de Relatórios
app.pathReport=/home/rpt/
# Tempo de Time Out do usuário
app.sessionTimeOut=60
```

O primeiro é o application.properties, o qual possui parâmetros como diretórios utilizados pelo sistema, e algumas informações básicas como tamanho da página para paginação, tempo de sessão do usuário entre outras coisas.

O jdbc.properties contém os parâmetros necessários para a conexão com o bando de dados como driver, url, usuário e senha.

Temos ainda o log4J.properties no qual encontramos as configurações para o log do sistema, e por ultimo o smtp.properties onde ficam as configurações de e-mail como o host, e o e-mail dos administradores do sistema.

As próximas configurações são às do Springframework, as quais encontram-se em um arquivo xml chamado: applicationContext.xml.

Neste arquivo temos mapeadas todas as classes às quais desejamos que fiquem sob o controle do spring. As Actions, Business, DAO, e todas as classes necessárias para a estrutura da aplicação, e também a configuração do Ibatis, este arquivo pode ser visualizado no ANEXO I deste documento.

As configurações do Struts também se encontram em um arquivo xml, o qual é denominado struts.xml, este arquivo possui o mapeamento das actions, e das paginas jsp. Podemos observá-lo no anexo II.

Por fim temos o web.xml, o qual porta as informações que a aplicação precisa para ser inicializada, todas as aplicações web desenvolvidas em java possuem este arquivo.

6.2 Paginas JSP

O leiaute da aplicação foi baseado no ViaInterface[viainterface] disponibilizado no repositório do projeto ViaDigital, precisando acrescentar apenas as opções do menu, e os formulários.

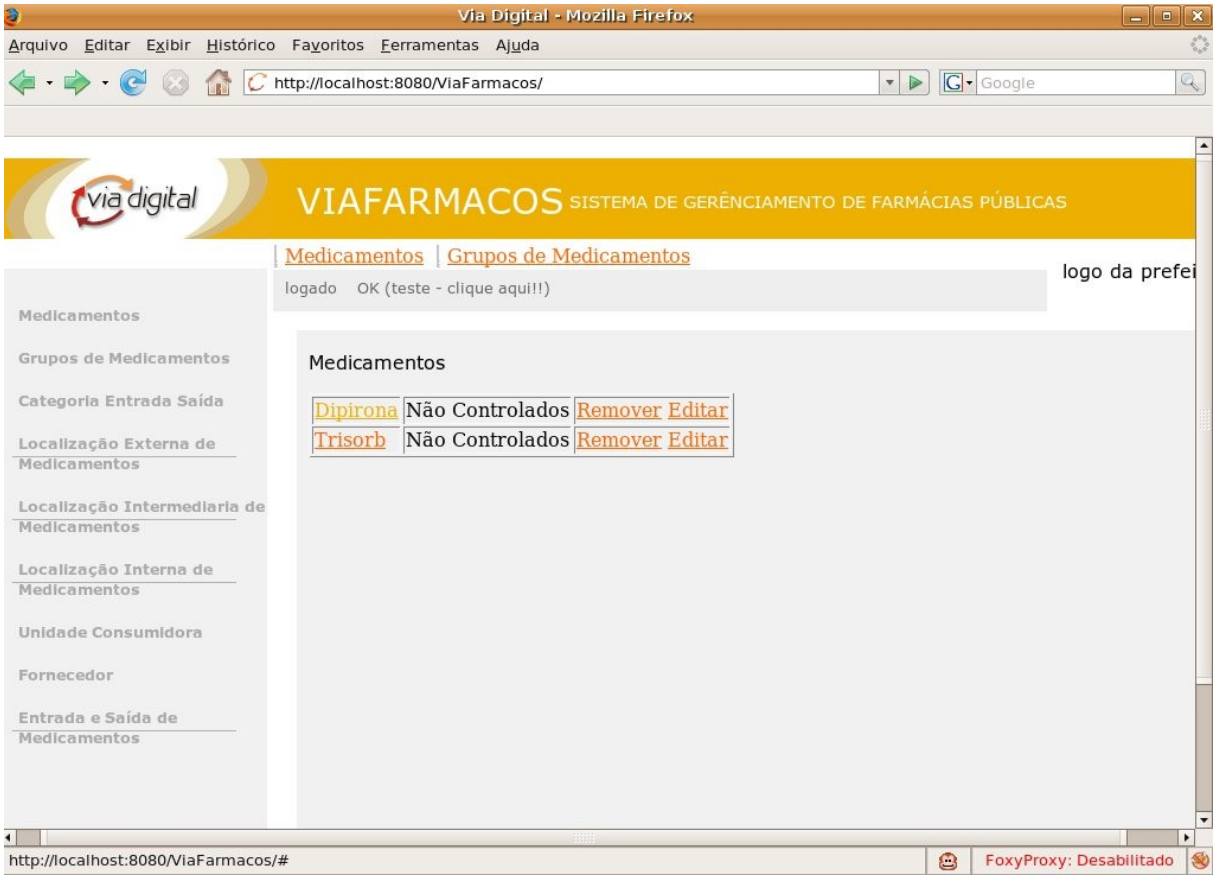
Os formulários foram desenvolvidos utilizando as taglibs do Struts2, de forma que interagissem diretamente com as actions, as quais são invocadas através de endereços pré-definidos, os quais se encontram no arquivo de configuração struts.xml.

Utilizando tais taglibs o desenvolvimento dos formulários foi simples e ágil, encontrando algumas dificuldades inicialmente no combos, implementados utilizando a tag <s:select>, porém foram rapidamente superadas. Foram criados basicamente três tipos de páginas, as de listagem, as de cadastro e as de visualização.

As páginas de listagem apresentam uma lista completa dos itens contidos no módulo selecionado. Nessa lista pode-se clicar sobre o nome do item para ter uma visualização completa, ou do lado direito em editar para editar alguma propriedade do item, ou ainda mais a direita em remover, para remover o item.

Abaixo podemos observar a imagem de uma página de listagem para melhor compreendermos:

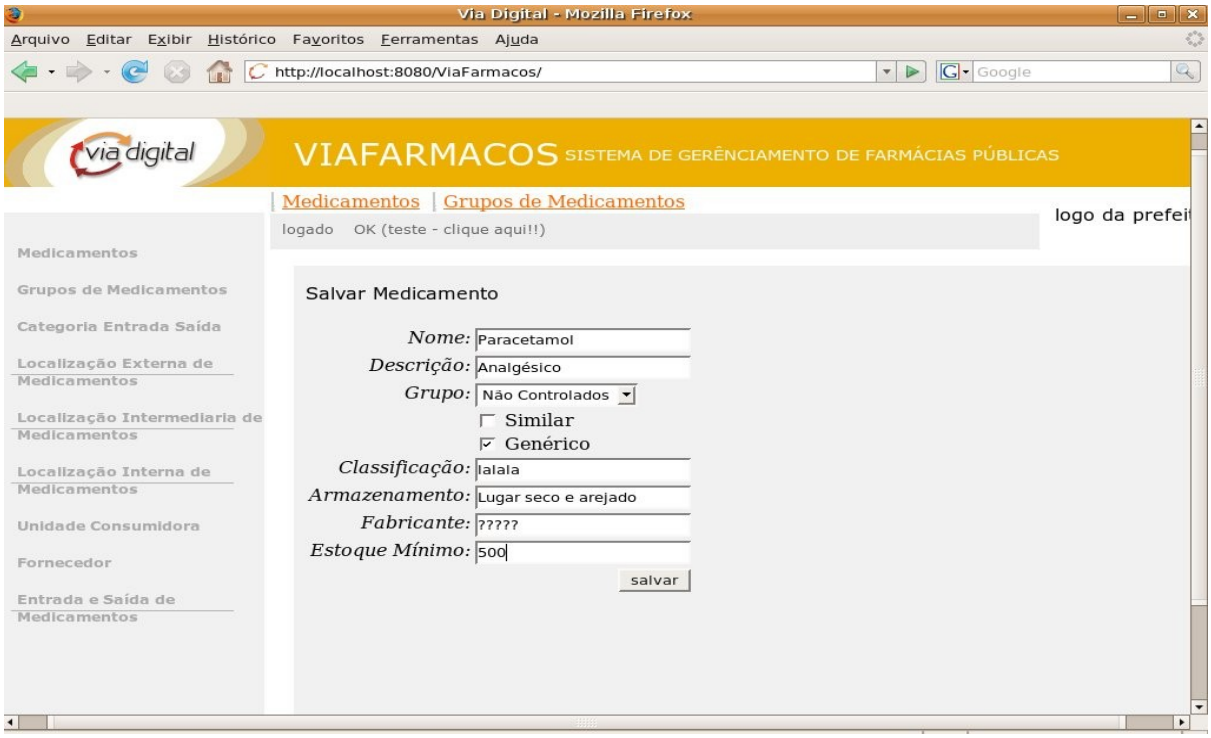
Figura 5 - Listar Medicamentos



O segundo tipo de página é o de cadastro, no qual é disponibilizado ao usuário um formulário com as propriedades do item a ser cadastrado, que depois de preenchido deve ser enviado ao sistema através do botão salvar .

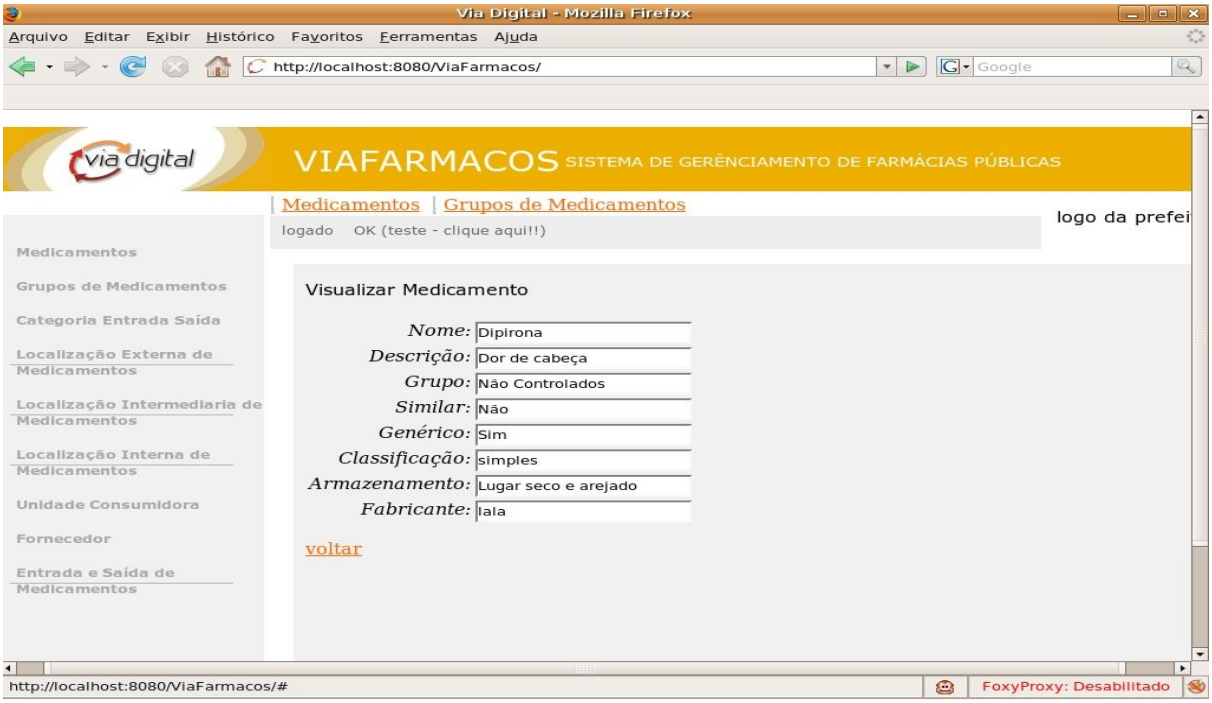
Em seguida podemos observar o formulário para cadastro de medicamentos:

Figura 6 Cadastro de Medicamento



E temos também as páginas de visualização, na qual o usuário pode verificar todas as propriedades do item selecionado. Abaixo podemos observar um exemplo:

Figura 7 - Visualizar Medicamento



6.3 Actions

Para cada módulo do sistema é criada uma action, as quais são basicamente todas iguais, implementando métodos padrões e com atributos padrões, o motivo para cada módulo possuir uma, é que os atributos pertencem a classes diferentes, ou seja, referenciam classes diferentes. Por exemplo a action de medicamentos, chamada MedicamentoAction, possui como atributos um objeto da classe Medicamento, a referência a classes de negócios de medicamento, chamada de MedicamentoBusiness, e a lista de medicamentos. E da mesma maneira as outras actions referenciam os objetos referentes a seu fluxo.

Os atributos básicos das actions são, um Integer que contém o identificador do objeto que está sendo manipulado, um DTO do objeto tratado naquela action, um List, com todos os DTOs, e uma referência ao Business que dá continuidade ao fluxo do módulo.

Os métodos implementados são os métodos de acesso aos atributos, e além deles também temos, o save que trata da inclusão e atualização de dados, execute que atualiza o atributo List, o delete que exclui um atributo selecionado, e por fim o prepare que preenche o objeto que está sendo manipulado.

A seguir temos o exemplo de uma action utilizada na aplicação:

```
package br.org.viadigital.viafarmacos.web.action;

import java.util.List;

import br.org.viadigital.viafarmacos.model.business.GrupoBusiness;
import br.org.viadigital.viafarmacos.model.business.MedicamentoBusiness;
import br.org.viadigital.viafarmacos.model.dto.Grupo;
import br.org.viadigital.viafarmacos.model.dto.Medicamento;

import com.opensymphony.xwork2.ActionSupport;

public class MedicamentoAction extends ActionSupport {

	public MedicamentoBusiness business;

	public GrupoBusiness grupoBusiness;

	public List<Medicamento> medicamentos;

	public List<Grupo> grupos;

	public Medicamento medicamento;

	public Integer id;

	public MedicamentoAction(MedicamentoBusiness business,
```

```
            GrupoBusiness grupoBusiness) {
      this.business = business;
      this.grupoBusiness = grupoBusiness;
}

@SuppressWarnings("unchecked")
public String execute() {
      this.medicamentos = business.selectMedicamentos();
      this.grupos = grupoBusiness.selectGrupos();
      return ActionSupport.INPUT;
}

public String save() {
      try {
            if (medicamento.getId() == 0) {
                  this.business.insertMedicamento(medicamento);
            } else {
                  this.business.updateMedicamento(medicamento);
            }
            this.medicamento = new Medicamento();
      } catch (Exception e) {
            e.printStackTrace();
      }
      return ActionSupport.SUCCESS;
}

public String remove() {

      this.business.removeMedicamento(id);

      return execute();
}

public List<Medicamento> getMedicamentos() {
      return medicamentos;
}

public Integer getId() {
      return id;
}

public void setId(Integer id) {
      this.id = id;
}

public String prepare() throws Exception {
      if (id != null) {
            medicamento = business.find(id);
      }
      return this.execute();
}

public List<Grupo> getGrupos() {
      return grupos;
}

public Medicamento getMedicamento() {
      return medicamento;
}
```

```
	public void setMedicamento(Medicamento medicamento) {
		this.medicamento = medicamento;
	}
}
```

6.4 Classes de negócio

Nessas classes encontram-se as regras de negócio da aplicação, porém utilizando os frameworks escolhidos, grande parte dessas regras, estão nos arquivos de configuração, como controle de transações e cache da aplicação. Abaixo podemos observar a configuração para instanciação e controle de transações de uma classes de negócios.

```
<bean id="medicamentoBusiness"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryB
ean">
	<property name="transactionManager" ref="transactionManager" />
	<property name="target" ref="medicamentoTarget" />
	<property name="proxyTargetClass" value="true" />
	<property name="transactionAttributes">
		<props>
			<prop key="insertMedicamento">
				PROPAGATION_REQUIRED, ISOLATION_READ_UNCOMMITTED,
					-Exception
			</prop>
			<prop key="updateMedicamento">
				PROPAGATION_REQUIRED, ISOLATION_READ_UNCOMMITTED,
					-Exception
			</prop>
		</props>
	</property>
</bean>

<bean id="medicamentoTarget" parent="abstractBusiness"
class="br.org.viadigital.viafarmacos.model.business.MedicamentoBusiness">

	<property name="medicamentoDAO">
		<ref local="medicamentoDAO" />
	</property>

</bean>
```

Com esses configurações o controle de transações seria realizado nos métodos updateMedicamento e insertMedicamento da classe MedicamentoBusiness .

6.5 Classes de Persistência

As classes de persistência ou DAOs, implementam os métodos de acesso ao banco de dados, cada módulo possui uma dessas classes, apenas elas podem acessar o banco de dados, assim para executar um *insert* ou um *select* obrigatóriamente a requisição tem que passar por esta classe. O desenvolvimento desta camada da aplicação é bastante simples, visto que, é necessário apenas cria-lo os com os mesmos nomes aos do business, e em seu conteúdo tem apenas uma chamada a um sql. Passando quando necessário algum parâmetro.

Na aplicação o único DAO que foge um pouco ao padrão é o LozalizacaoDAO, visto que existem três tipos de localização e todos acessam essa mesma classe, na verdade todo o fluxo desse módulo funciona dessa maneira, porém a única diferença é que possui um *CRUD* para cada tipo de localização. Esses tipos são: Externa, Intermediária e interna.

6.6 Arquivos XML de Persistência

Os arquivos xml de persistência ou DAO.xml tratam do mapeamento das tabelas do banco de dados para objetos, e possuem as consultas SQL, as quais são acessadas pelas classes DAO. Nesta aplicação temos basicamente seis desses arquivos, porém seria necessário apenas um. Utilizamos um arquivo para cada caso de uso, de maneira que o sistema fique mais organizado e manutenivel, sempre que surge um problema referente a alguma consulta de medicamento, por exemplo, encontraremos a consulta com problema facilmente no arquivo MedicamentoDAO.xml.

A estrutura desses arquivos é pré-definida pelo ibatis, impondo um padrão bastante claro e objetivo, podemos observar o arquivo MedicamentoDAO.xml, no anexo III.

7 Sistema

No presente Capítulo serão apresentadas algumas das interfaces do sistema, com suas respectivas descrições. As interfaces seguem um padrão, cadastros, listagem e visualizações mudam apenas os campos apresentados, por isso julga-se desnecessário apresentar todas as telas do sistema.

7.1 Listagens

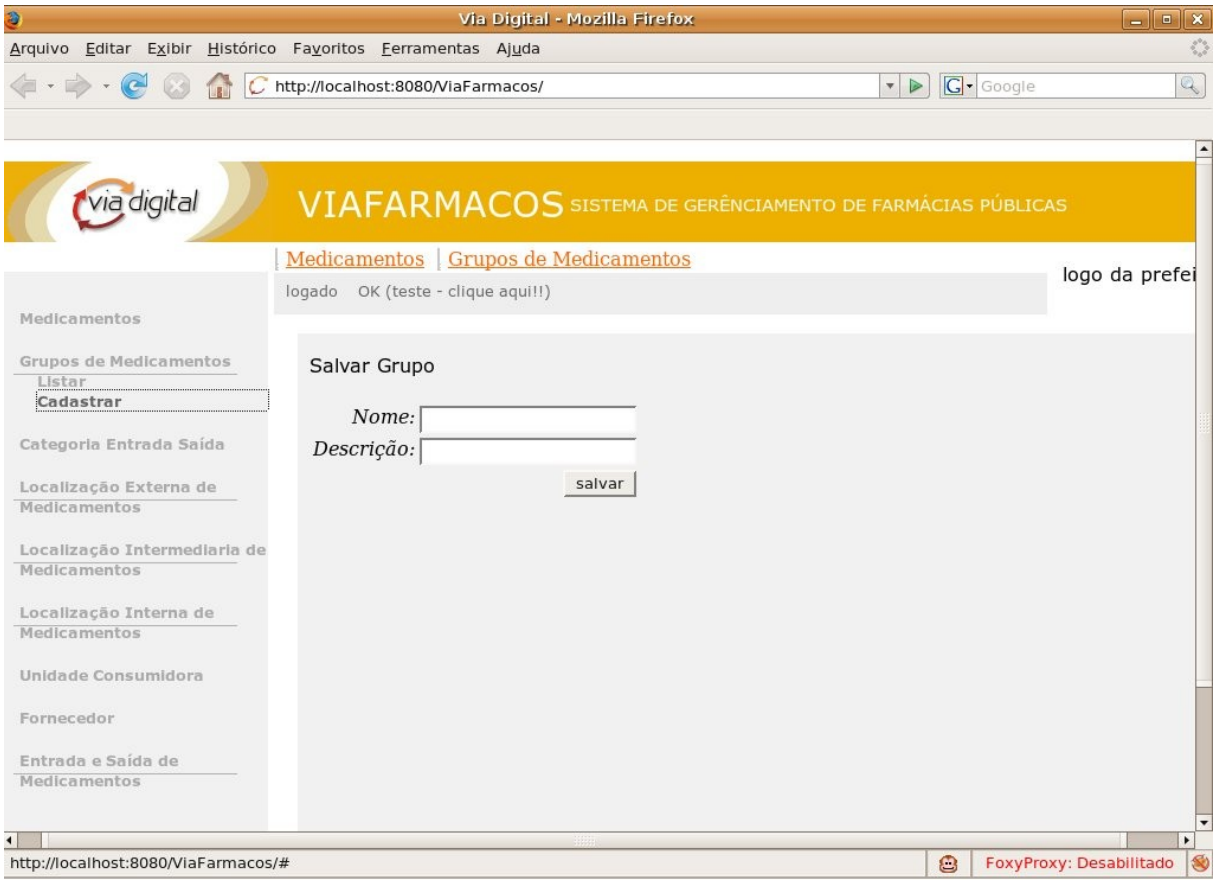
Todas as listagens do sistema seguem um padrão, basicamente é exibida uma tabela com alguns dos dados do item de interesse, e a descrição do item é um link para que possamos melhor visualizar o item desejado. Abaixo podemos visualizar o uma listagem que segue este padrão:

Figura 8 - Listagem de Saídas Autorizadas de Medicamentos

7.2 Cadastros

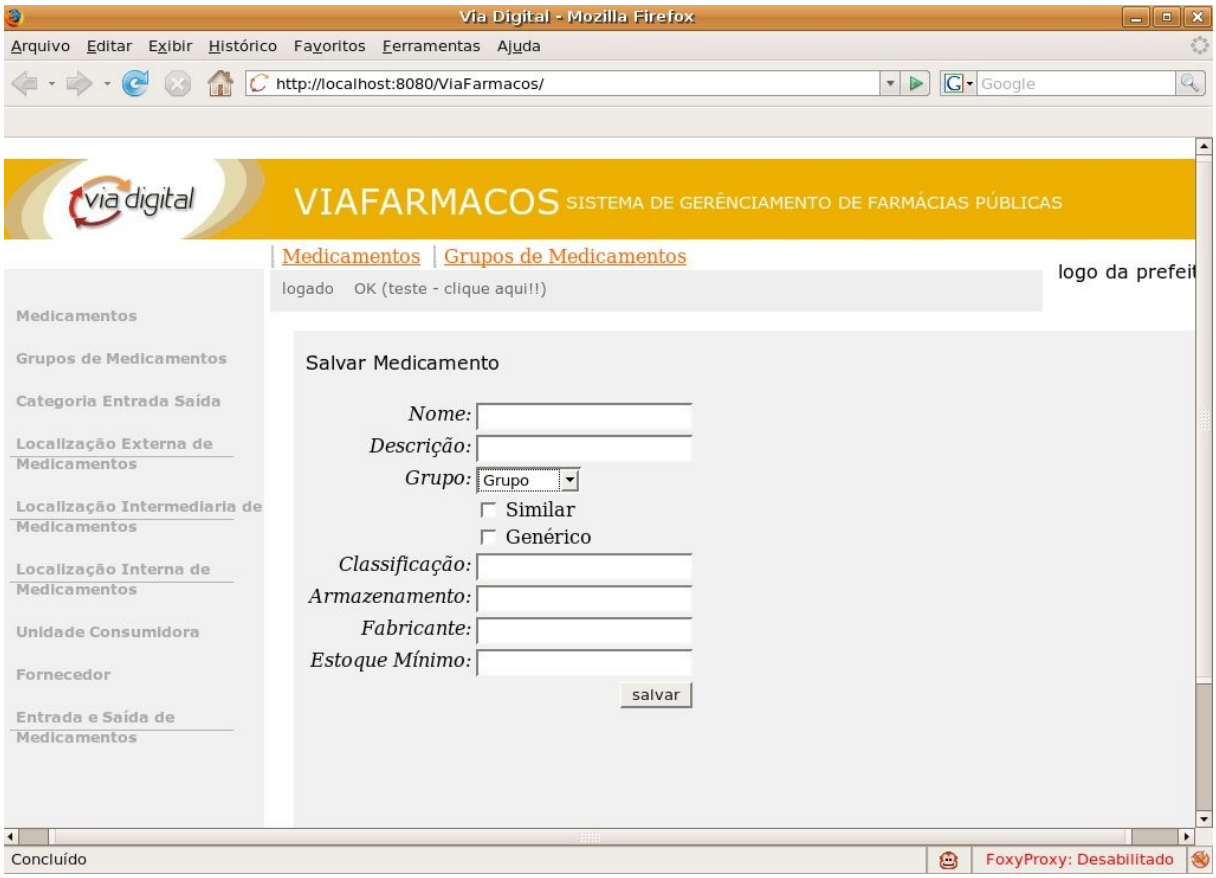
Abaixo Podemos observar algumas das telas de cadastro do sistema, a primeira tela que podemos observar é a de cadastro de grupo de medicamentos:

Figura 9 - Cadastro de grupo de Medicamentos



É um cadastro bastante simples com apenas dois campos, porém para poder cadastrar um medicamento antes precisamos ter o seu grupo cadastrado, a próxima imagem é o cadastro de medicamento:

Figura 10 - Cadastro de medicamentos

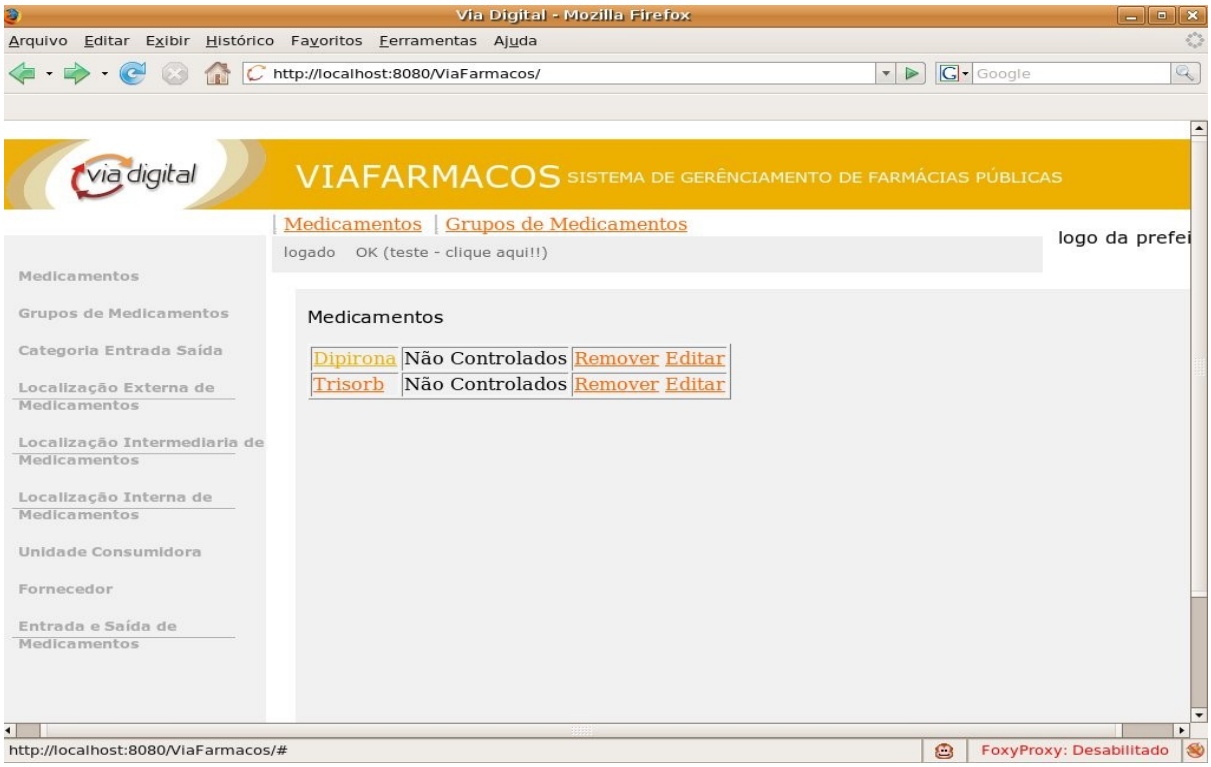


Este é um cadastro uma pouco mais complexo, com alguns campos abertos, um combo, no qual o usuário pode escolher o grupo de medicamentos e alguns *checkbox* para informar ao sistema se o medicamento é similar e/ou genérico.

7.3 Visualizações

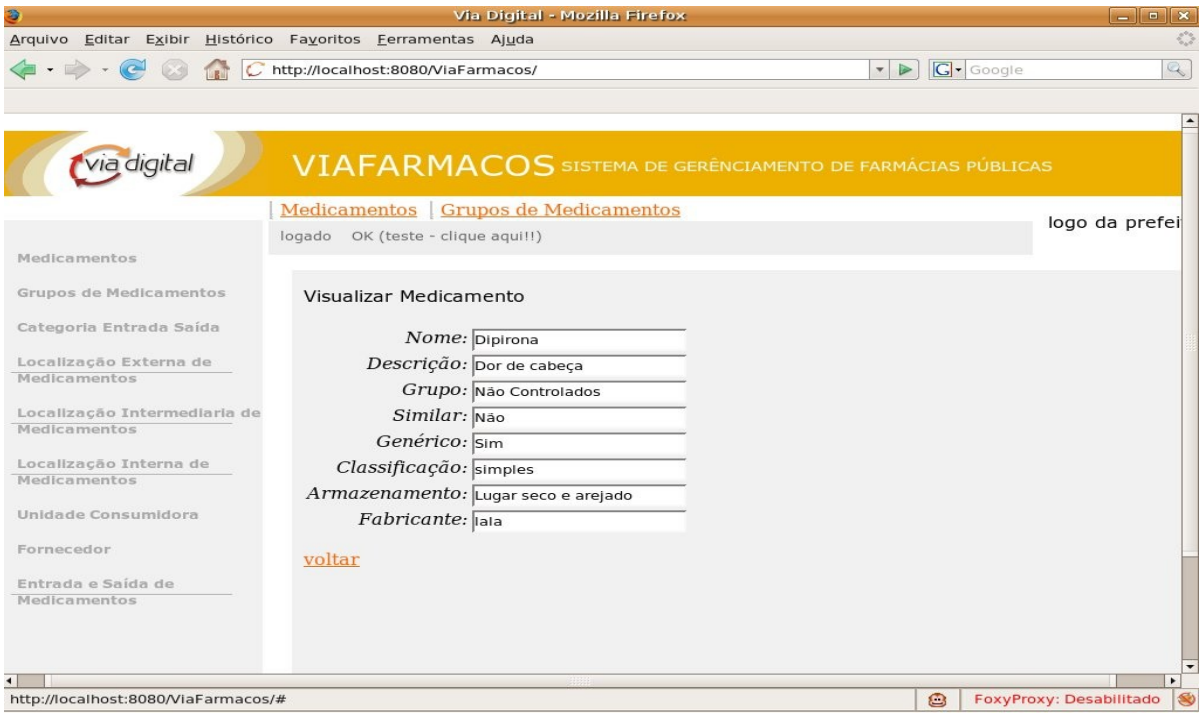
Da mesma maneira que os cadastros a visualização segue um padrão bastante semelhante com o do cadastro, porém os campos não são editáveis. Esta funcionalidade é utilizada para se obter maiores informações sobre o item selecionado, ela pode ser acessada a partir da listagem do item de escolha, abaixo temos um figura da listagem de medicamentos, na qual podemos observar que o nome do medicamento é na verdade um *link*, pelo qual podemos obter mais informações sobre o mesmo.

Figura 11 - Listagem de Medicamentos



Após clicar sobre o medicamento escolhido o usuário visualizará a seguinte tela:

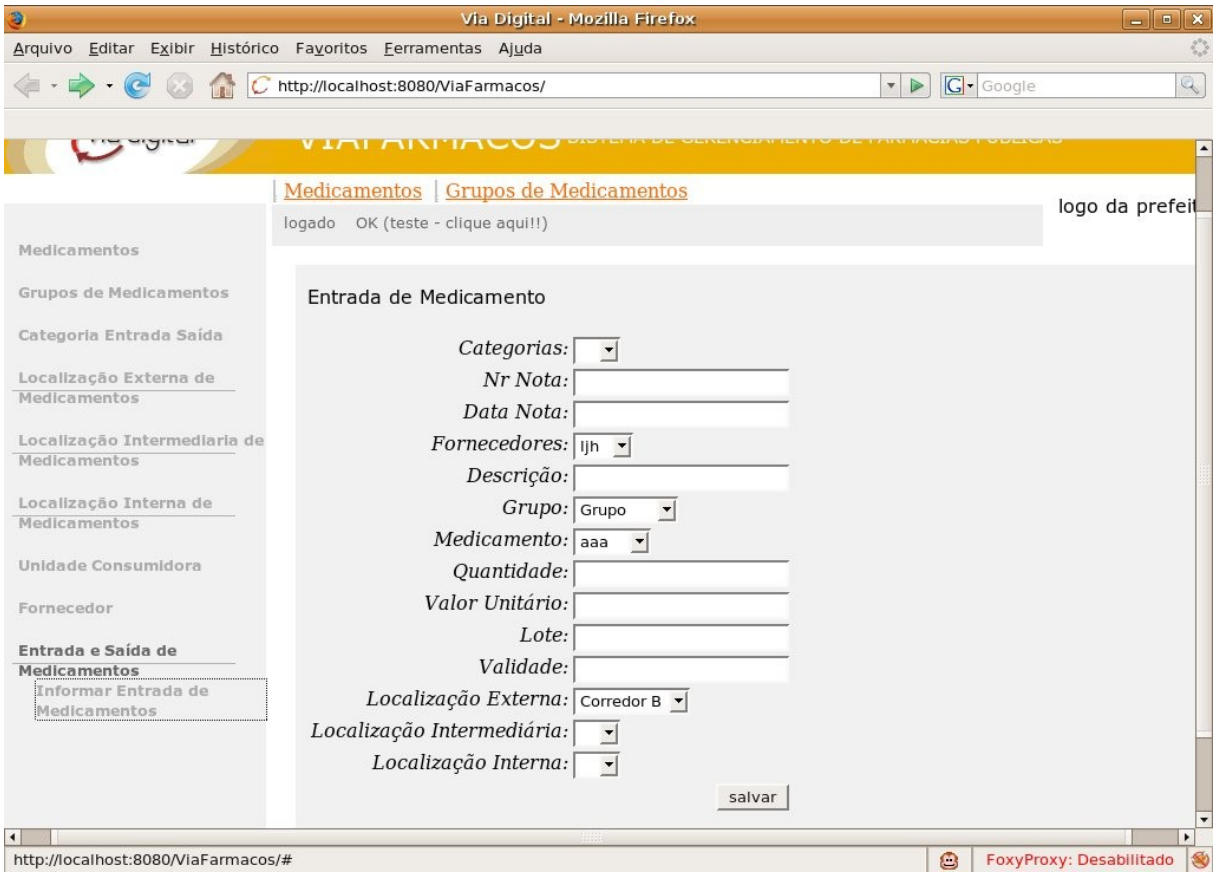
Figura 12 - Visualização de Medicamento



7.4 Funcionalidades

Aqui poderemos observar as telas de cada funcionalidade e entender exatamente o que elas fazem, a primeira funcionalidade a ser observada é a de "Informar Entrada de Medicamentos", na qual o usuário informa ao sistema que uma nova remessa de medicamentos chegou. Abaixo visualizamos a interface da funcionalidade:

Figura 13 - Entrada de Medicamentos



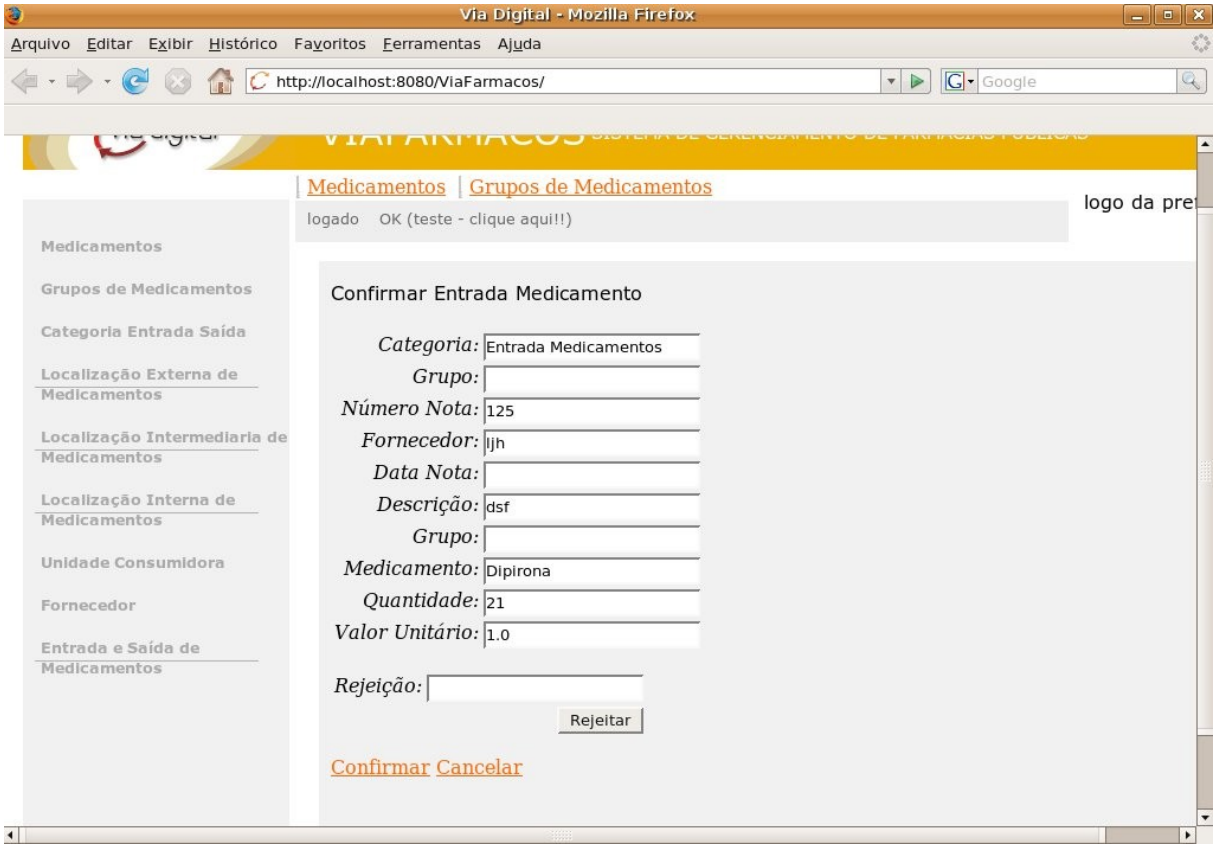
É necessário informar separadamente sobre cada lote do medicamento, na propriedade categorias o usuário poderá escolher dentre as categorias de entradas de medicamento, se a entrada é no almoxarifado central ou em alguma das farmácias, na propriedade medicamento, aparecem apenas os medicamentos do grupo selecionado, da mesma forma acontece nas localizações que funcionam hierarquicamente.

Depois que é informada uma entrada de medicamento, ela deve ser confirmada.

Então o usuário seleciona na funcionalidade “confirmar entrada de medicamentos”, e será apresentada uma tela com a lista das entradas ainda não confirmadas. Para visualizar melhor a entrada cadastrada o usuário clica sobre a entrada, e aparecerá uma tela com todos os dados desta, na qual o usuário poderá confirmar a entrada ou rejeitá-la, dando um justificativa.

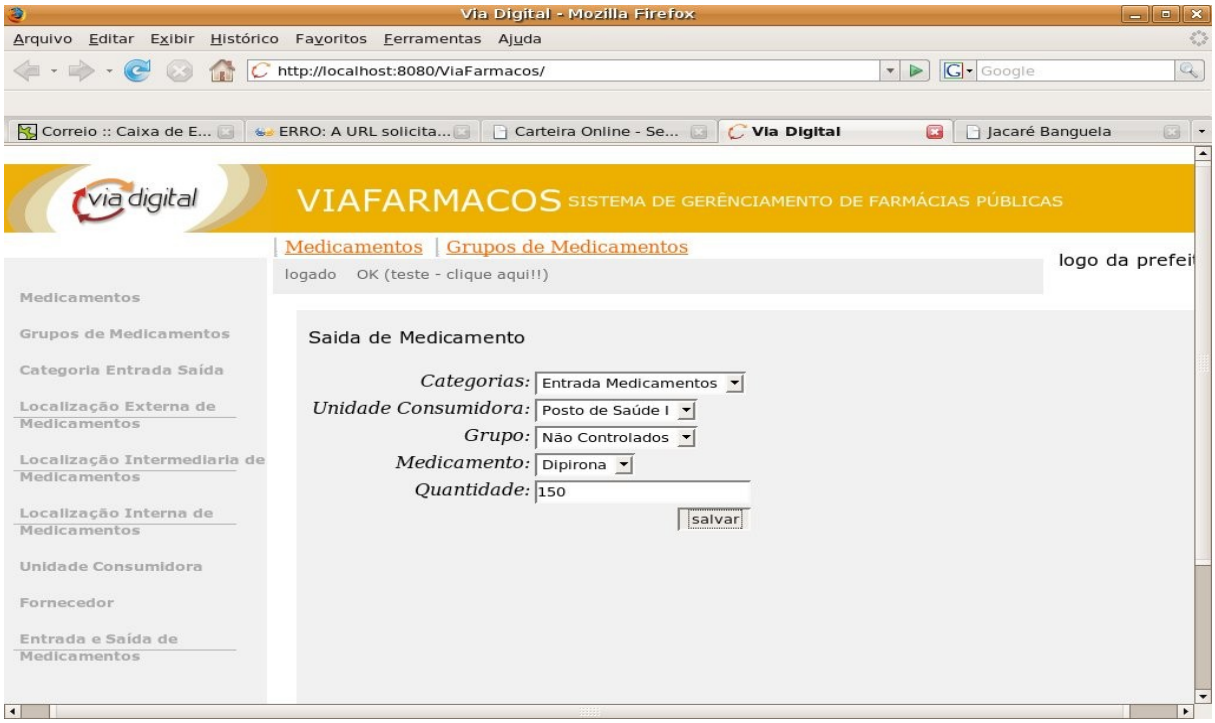
Abaixo podemos observar essa tela:

Figura 14 - Confirmação de Entrada de Medicamentos



Os medicamentos contidos no almoxarifado podem ter sua saída para uma unidade consumidora solicitada através da funcionalidade “Solicitar saída de medicamentos”, a qual pode ser feita pelo responsável de uma unidade consumidora, a fim de abastece-la, ou pelo responsável pelo almoxarifado por qualquer outro motivo. Esta solicitação é realizada a partir da seguinte interface:

Figura 15 - Solicitação de Saída de medicamentos



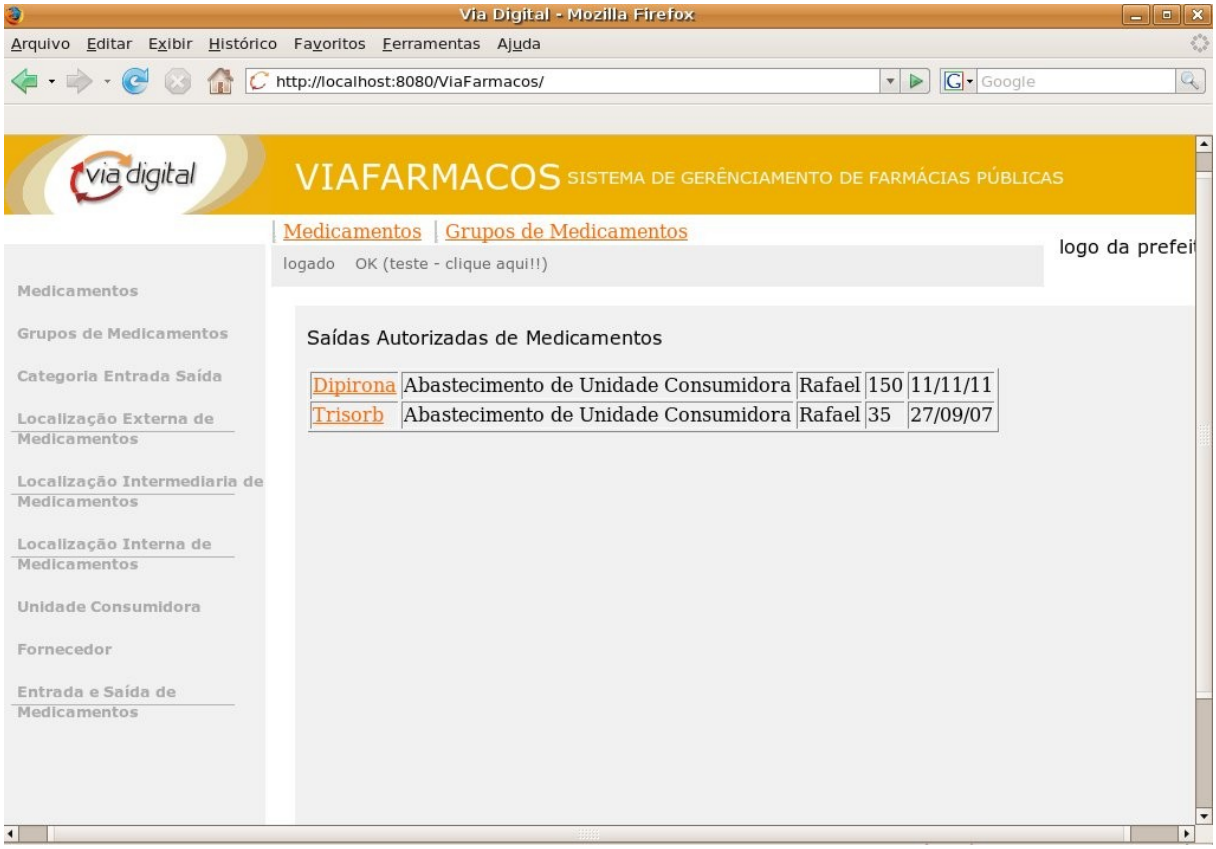
Depois de uma solicitação de saída ser feita, o usuário deve aguardar que o usuário gestor autorize a mesma. Este pode fazer isso acessando a seguinte tela do sistema:

Figura 16 -Autorizar Saída de Medicamentos

Na qual a solicitação pode ser autorizada, ou negada mediante uma justificativa.

Depois que uma saída é autorizada, a unidade consumidora pode ir retirar os medicamentos solicitados do almoxarifado central, no momento em que esta retirada é realizada o usuário deve confirmar a saída dos medicamentos no sistema. Nesta funcionalidade todas as saídas autorizadas serão listadas, e o usuário deverá selecionar a referente àquela saída para ser confirmada.

Figura 17 - Listar Saídas Autorizadas



Sempre que o usuário responsável pelo almoxarifado efetua o login no sistema, este executa uma verificação para checar se algum dos medicamentos atingiu a quantidade mínima no estoque. Quando um ou mais medicamentos se encontram nessa situação, o sistema mostra ao usuário uma lista com algumas informações dos medicamentos, os quais o usuário deve confirmar ao sistema que está ciente de sua escassez.

Conclusão

Com a estrutura montada para o desenvolvimento do aplicativo, seu desenvolvimento se deu de forma bastante simples e rápida, desde os cadastros até os relatórios. Com a utilização dos frameworks escolhidos, o reuso foi bastante significativo, encurtando de forma considerável a curva de desenvolvimento.

Tal sistema atingiu as expectativas, e está apto a auxiliar as farmácias públicas de modo a melhorar a organização de seus almoxarifados, e otimizar os recursos direcionados a compra de medicamentos. Gerenciando de forma centralizada todas as unidades consumidoras, dessa forma, existe a possibilidade equilibrar a distribuição destes medicamentos diminuindo o desperdício.

Durante o desenvolvimento do trabalho, estive em contato com algumas das prefeituras da região. Mais especificamente com a área da saúde de cada uma, e pude constatar que, as que ainda não informatizam seus postos de saúde e farmácias públicas estão providenciando recursos para isso. Possibilitando assim que o sistema realmente seja útil para estas.

Das prefeituras que tive a oportunidade de me comunicar, todas demonstraram interesse no sistema, desde profissionais da saúde, até da informática. Porém mesmo apresentando este interesse, não foram capazes de fazer esforço algum para que isto ocorresse.

O que pude observar com tudo isso é que, apesar da infraestrutura das prefeituras estar melhorando, as pessoas envolvidas dificultam o processo, por falta de interesse enfatizando questões não prioritárias em benefício próprio. Talvez se as pessoas responsáveis pela área não tivessem interesse político, e fossem melhor preparadas o facilitaria as coisas. De nada adianta um sistema que pode melhorar, e muito o funcionamento de uma farmácia pública, se ninguém fizer o esforço de utiliza-lo.

Bibliografia

[spring] Spring Framework, disponível em: http://www.springframework.org/

[sun] SUN Microsystems, Informações sobre a linguagem Java, Design Patterns, e arquitetura MVC, disponível em http://java.sun.com/

[postgresql] PostgreSQLBrasil, Informações sobre o sgbd PostgreSQL, disponível em: http://www.postgresql.org.br

[hibernate] Hibernate, informações sobre o framework hibernate, disponivel em: www.hibernate.org

[struts] Struts, disponível em: http://struts.apache.org/

[viadigital] Via Digital, informações sobre projetos concluídos e em andamento, disponível em: http://www.viadigital.ufsc.br/

[softwarelivre] Software livre, informações sobre software livre, retirado de: www.softwarelivre.gov.br/swlivre

[softwarelivre1] Software livre, informações sobre licenças de software livre, retirado de: www.softwarelivre.org/whatisit.php

[action] Apache, informações técnicas sobre ActionSupport, retirado de: http://struts.apache.org/2.0.8/struts2core/apidocs/com/opensymphony/xwork2

[cvs] Guia Foca GNU/Linux, informações sobre cvs, retirado de: http://focalinux.cipsga.org.br/guia/avancado/ch-s-cvs.htm

[ireport] IReport, informações a respeito do IReport e JasperReport, retirado de:

http://jasperforge.org/sf/projects/ireport

[eclipse] Eclipse.org, Informações e download do Eclipse, retirado de:
http://www.eclipse.org

[tomcat] Apache, informações e download do tomcat, retirado de:
http://tomcat.apache.org

[ViaFarmacos] Site do projeto, https://repositorio.viadigital.org.br/projects/farmacias/

[swfactory] Empresa SwFactory consultoria e sistemas, http://www.swfactory.com.br/

ANEXOS

Anexo I - applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

	<!-- === DefiniÃ§Ã£o de arquivos/recursos de mensagens e constantes
=== -->
	<bean id="messageSource"


class="org.springframework.context.support.ResourceBundleMessageSource">
		<property name="basename">

	<value>br/org/viadigital/viafarmacos/common/MessageResources</value>
		</property>
	</bean>

	<bean id="messageSourceAccessor"

	class="org.springframework.context.support.MessageSourceAccessor">
		<constructor-arg>
			<ref bean="messageSource" />
		</constructor-arg>
	</bean>

	<bean id="constantSource"


class="org.springframework.context.support.ResourceBundleMessageSource">
		<property name="basename">

	<value>br/org/viadigital/viafarmacos/common/ConstantResources</value>
		</property>
	</bean>

	<bean id="constantSourceAccessor"

	class="org.springframework.context.support.MessageSourceAccessor">
		<constructor-arg>
			<ref bean="constantSource" />
		</constructor-arg>
	</bean>


	<!-- === Recurso de configuraÃ§Ã£o. Carrega e propriedades
(constantes) da aplicaÃ§Ã£o -->
	<bean id="propertyConfigurer"
```

```
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigur
er">
            <property name="locations">
                  <list>
                        <value>/WEB-INF/jdbc.properties</value>

                        <value>/WEB-INF/application.properties</value>
                        <value>/WEB-INF/smtp.properties</value>
                  </list>
            </property>
      </bean>

      <!-- === Recursos de Cache === -->
      <bean id="cacheManager"


class="org.springmodules.cache.provider.oscache.OsCacheManagerFactoryBean">
      </bean>

      <bean id="cacheProvider"
            class="org.springmodules.cache.provider.oscache.OsCacheFacade">
            <property name="cacheManager" ref="cacheManager" />
      </bean>

      <!-- === Recursos disponibilizados pelo Spring === -->
      <!--
      <bean id="mailSender"
            class="org.springframework.mail.javamail.JavaMailSenderImpl">
            <property name="host" value="${mail.host}" />
      </bean>

      -->

      <!-- === Data Base Objects == -->
      <bean id="transactionManager"


class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
            <property name="dataSource" ref="dataSource" />
      </bean>

      <bean id="dataSource"
            class="org.apache.commons.dbcp.BasicDataSource"
            destroy-method="close">
            <property name="driverClassName"
                  value="${jdbc.driverClassName}" />
            <property name="url" value="${jdbc.url}" />
            <property name="username" value="${jdbc.username}" />
            <property name="password" value="${jdbc.password}" />
      </bean>

      <bean id="sqlMapClient"
            class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
            <property name="configLocation">
                  <value>/WEB-INF/sqlmap-config.xml</value>
            </property>
      </bean>
```

```
<!-- === Data Access Objects (DAO) === -->
<!-- <property> Ã© o nome do atributo na classe <ref> Ã© o nome da
referencia nesse arquivo
<value> Ã© um valor que pode ser definido(vai ser pego do
aplication.properties) -->


      <bean id="medicamentoDAO"
class="br.org.viadigital.viafarmacos.model.dao.MedicamentoDAO">
            <property name="dataSource">
                  <ref local="dataSource" />
            </property>
            <property name="sqlMapClient">
                  <ref local="sqlMapClient" />
            </property>
      </bean>

      <bean id="grupoDAO"
class="br.org.viadigital.viafarmacos.model.dao.GrupoDAO">
            <property name="dataSource">
                  <ref local="dataSource" />
            </property>
            <property name="sqlMapClient">
                  <ref local="sqlMapClient" />
            </property>
      </bean>

      <bean id="categoriaEntradaSaidaDAO"
class="br.org.viadigital.viafarmacos.model.dao.CategoriaEntradaSaidaDAO">
            <property name="dataSource">
                  <ref local="dataSource" />
            </property>
            <property name="sqlMapClient">
                  <ref local="sqlMapClient" />
            </property>
      </bean>

      <bean id="localizacaoDAO"
class="br.org.viadigital.viafarmacos.model.dao.LocalizacaoDAO">
            <property name="dataSource">
                  <ref local="dataSource" />
            </property>
            <property name="sqlMapClient">
                  <ref local="sqlMapClient" />
            </property>
      </bean>

      <bean id="unidadeConsumidoraDAO"
class="br.org.viadigital.viafarmacos.model.dao.UnidadeConsumidoraDAO">
            <property name="dataSource">
                  <ref local="dataSource" />
            </property>
            <property name="sqlMapClient">
                  <ref local="sqlMapClient" />
            </property>
      </bean>

      <bean id="fornecedorDAO"
class="br.org.viadigital.viafarmacos.model.dao.FornecedorDAO">
```

```
            <property name="dataSource">
                  <ref local="dataSource" />
            </property>
            <property name="sqlMapClient">
                  <ref local="sqlMapClient" />
            </property>
      </bean>

      <bean id="entradaSaidaMedicamentoDAO"
class="br.org.viadigital.viafarmacos.model.dao.EntradaSaidaMedicamentoDAO">
            <property name="dataSource">
                  <ref local="dataSource" />
            </property>
            <property name="sqlMapClient">
                  <ref local="sqlMapClient" />
            </property>
      </bean>

      <!-- === Commom Classes === -->

      <bean id="session"
class="br.org.viadigital.viafarmacos.common.Session">
      </bean>

      <bean id="mail" class="br.org.viadigital.viafarmacos.common.Mail">
            <property name="from" value="${mail.from}" />
            <!--
            <property name="mailSender">
                  <ref local="mailSender" />
            </property>
             -->
      </bean>


      <!-- === Classes de negÃ³cio - Business Rules === -->

      <bean id="abstractBusiness" abstract="true"


class="br.org.viadigital.viafarmacos.model.business.AbstractBusiness">
            <property name="constant">
                  <ref local="constantSourceAccessor" />
            </property>
            <property name="message">
                  <ref local="messageSourceAccessor" />
            </property>
            <property name="mail">
                  <ref local="mail" />
            </property>
            <property name="emailSupport" value="${mail.to.support}" />
      </bean>

      <bean id="medicamentoBusiness"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryB
ean">
            <property name="transactionManager" ref="transactionManager" />
            <property name="target" ref="medicamentoTarget" />
            <property name="proxyTargetClass" value="true" />
            <property name="transactionAttributes">
```

```
                  <props>
                        <prop key="insertMedicamento">
                              PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
                              -Exception
                        </prop>
                        <prop key="updateMedicamento">
                              PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
                              -Exception
                        </prop>
                  </props>
            </property>
      </bean>

      <bean id="medicamentoTarget" parent="abstractBusiness"


class="br.org.viadigital.viafarmacos.model.business.MedicamentoBusiness">
            <property name="medicamentoDAO">
                  <ref local="medicamentoDAO" />
            </property>

      </bean>

      <bean id="grupoBusiness"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryB
ean">
            <property name="transactionManager" ref="transactionManager" />
            <property name="target" ref="grupoTarget" />
            <property name="proxyTargetClass" value="true" />
            <property name="transactionAttributes">
                  <props>
                        <prop key="insertGrupo">
                              PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
                              -Exception
                        </prop>
                  </props>
            </property>
      </bean>

      <bean id="grupoTarget" parent="abstractBusiness"

      class="br.org.viadigital.viafarmacos.model.business.GrupoBusiness">
            <property name="grupoDAO">
                  <ref local="grupoDAO" />
            </property>

      </bean>

      <bean id="categoriaEntradaSaidaBusiness"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryB
ean">
            <property name="transactionManager" ref="transactionManager" />
            <property name="target" ref="categoriaEntradaSaidaTarget" />
            <property name="proxyTargetClass" value="true" />
            <property name="transactionAttributes">
                  <props>
```

```
                        <prop key="insertCategoriaEntradaSaida">
                              PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
                              -Exception
                        </prop>
                  </props>
            </property>
      </bean>

      <bean id="categoriaEntradaSaidaTarget" parent="abstractBusiness"


class="br.org.viadigital.viafarmacos.model.business.CategoriaEntradaSaidaBu
siness">
            <property name="categoriaEntradaSaidaDAO">
                  <ref local="categoriaEntradaSaidaDAO" />
            </property>
      </bean>

      <bean id="localizacaoBusiness"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryB
ean">
            <property name="transactionManager" ref="transactionManager" />
            <property name="target" ref="localizacaoTarget" />
            <property name="proxyTargetClass" value="true" />
            <property name="transactionAttributes">
                  <props>
                        <prop key="inserirLocalizacaoExt">
                              PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
                              -Exception
                        </prop>
                  </props>
            </property>
      </bean>

      <bean id="localizacaoTarget" parent="abstractBusiness"


class="br.org.viadigital.viafarmacos.model.business.LocalizacaoBusiness">
            <property name="localizacaoDAO">
                  <ref local="localizacaoDAO" />
            </property>
      </bean>

      <bean id="fornecedorBusiness"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryB
ean">
            <property name="transactionManager" ref="transactionManager" />
            <property name="target" ref="fornecedorTarget" />
            <property name="proxyTargetClass" value="true" />
            <property name="transactionAttributes">
                  <props>
                        <prop key="inserirFornecedor">
                              PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
                              -Exception
                        </prop>
                  </props>
```

```
            </property>
      </bean>

      <bean id="fornecedorTarget" parent="abstractBusiness"


class="br.org.viadigital.viafarmacos.model.business.FornecedorBusiness">
            <property name="fornecedorDAO">
                  <ref local="fornecedorDAO" />
            </property>
      </bean>

      <bean id="unidadeConsumidoraBusiness"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryB
ean">
            <property name="transactionManager" ref="transactionManager" />
            <property name="target" ref="unidadeConsumidoraTarget" />
            <property name="proxyTargetClass" value="true" />
            <property name="transactionAttributes">
                  <props>
                        <prop key="inserirUnidadeConsumidora">
                              PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
                              -Exception
                        </prop>
                  </props>
            </property>
      </bean>

      <bean id="unidadeConsumidoraTarget" parent="abstractBusiness"


class="br.org.viadigital.viafarmacos.model.business.UnidadeConsumidoraBusin
ess">
            <property name="unidadeConsumidoraDAO">
                  <ref local="unidadeConsumidoraDAO" />
            </property>
      </bean>

      <bean id="entradaSaidaMedicamentoBusiness"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryB
ean">
            <property name="transactionManager" ref="transactionManager" />
            <property name="target" ref="entradaSaidaMedicamentoTarget" />
            <property name="proxyTargetClass" value="true" />
            <property name="transactionAttributes">
                  <props>
                        <prop key="inserirEntradaMedicamento">
                              PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
                              -Exception
                        </prop>
                  </props>
            </property>
      </bean>

      <bean id="entradaSaidaMedicamentoTarget" parent="abstractBusiness"
```

```
class="br.org.viadigital.viafarmacos.model.business.EntradaSaidaMedicamento
Business">
            <property name="entradaSaidaMedicamentoDAO">
                  <ref local="entradaSaidaMedicamentoDAO" />
            </property>
      </bean>

      <bean id="usuarioBusiness"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryB
ean">
            <property name="transactionManager" ref="transactionManager" />
            <property name="target" ref="usuarioTarget" />
            <property name="proxyTargetClass" value="true" />
            <property name="transactionAttributes">
                  <props>
                        <prop key="inserirLocalizacaoExt">
                              PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
                              -Exception
                        </prop>
                  </props>
            </property>
      </bean>

      <bean id="usuarioTarget" parent="abstractBusiness"

      class="br.org.viadigital.viafarmacos.model.business.UsuarioBusiness">

      </bean>


<!-- === Helper Classes === -->
      <bean id="paginacaoHelper"

      class="br.org.viadigital.viafarmacos.web.helper.PaginacaoHelper">
            <property name="pageSize" value="${app.pageSize}" />
      </bean>

      <!-- === Struts Actions === -->
<!--  <bean id="abstractAction" abstract="true"

      class="br.org.viadifital.viafarmacos.web.action.AbstractAction">
            <property name="constant">
                  <ref local="constantSourceAccessor" />
            </property>
            <property name="mail">
                  <ref local="mail" />
            </property>
            <property name="emailSupport" value="${mail.to.support}" />
      </bean> -->

      <bean id="medicamentoAction"
class="br.org.viadigital.viafarmacos.web.action.MedicamentoAction">

            <constructor-arg ref="medicamentoBusiness" />
            <constructor-arg ref="grupoBusiness" />
      </bean>

      <bean id="grupoAction"
```

```
class="br.org.viadigital.viafarmacos.web.action.GrupoAction">
            <constructor-arg ref="grupoBusiness" />
      </bean>

      <bean id="categoriaEntradaSaidaAction"
class="br.org.viadigital.viafarmacos.web.action.CategoriaEntradaSaidaAction
">
            <constructor-arg ref="categoriaEntradaSaidaBusiness" />
      </bean>

      <bean id="localizacaoAction"
class="br.org.viadigital.viafarmacos.web.action.LocalizacaoAction">

            <constructor-arg ref="localizacaoBusiness" />
      </bean>

      <bean id="unidadeConsumidoraAction"
class="br.org.viadigital.viafarmacos.web.action.UnidadeConsumidoraAction">

            <constructor-arg ref="unidadeConsumidoraBusiness" />
            <constructor-arg ref="usuarioBusiness" />
      </bean>

      <bean id="fornecedorAction"
class="br.org.viadigital.viafarmacos.web.action.FornecedorAction">

            <constructor-arg ref="fornecedorBusiness" />
      </bean>

      <bean id="entradaSaidaMedicamentoAction"
class="br.org.viadigital.viafarmacos.web.action.EntradaSaidaMedicamentoActi
on">
            <constructor-arg ref="entradaSaidaMedicamentoBusiness" />
            <constructor-arg ref="categoriaEntradaSaidaBusiness" />
            <constructor-arg ref="fornecedorBusiness" />
            <constructor-arg ref="grupoBusiness" />
            <constructor-arg ref="medicamentoBusiness" />
            <constructor-arg ref="localizacaoBusiness" />
            <constructor-arg ref="unidadeConsumidoraBusiness" />
      </bean>

      <bean id="verificaEstoqueAction"
class="br.org.viadigital.viafarmacos.web.action.VerificaEstoqueAction">

            <constructor-arg ref="entradaSaidaMedicamentoBusiness" />
      </bean>

      <!-- === Timer Tasks === -->
      <!-- timer tasks (declaraÃ§Ã£o dos bean timers) -->
      <bean id="expirarTask"
class="br.org.viadigital.viafarmacos.common.ExpirarTask">
            <property name="sistema">
                  <ref local="medicamentoBusiness" />
            </property>
      </bean>

      <!-- scheduled Tasks (instanciadores dos timers) -->
      <bean id="scheduledExpirarTask"
```

```
      class="org.springframework.scheduling.timer.ScheduledTimerTask">
            <property name="period" value="${app.timer.expirar.period}" />
            <property name="timerTask">
                  <ref bean="expirarTask" />
            </property>
      </bean>

      <!-- timer task factory (executador dos timers) -->
      <bean
            class="org.springframework.scheduling.timer.TimerFactoryBean">
            <property name="scheduledTimerTasks">
                  <list>
                        <ref bean="scheduledExpirarTask" />
                  </list>
            </property>
      </bean>
</beans>
```

Anexo II - struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
      <constant name="struts.objectFactory" value="spring" />
      <constant name="struts.devMode" value="true" />

      <package name="medicamento" namespace="/medicamento"
            extends="struts-default">

            <action name="remove" method="remove"
                  class="medicamentoAction">
                  <result>/jsp/listarMedicamento.jsp</result>
                  <result name="input">/jsp/listarMedicamento.jsp</result>
            </action>

            <action name="save" method="save" class="medicamentoAction">
                  <result>/jsp/listarMedicamento.jsp</result>
                  <result name="input">/jsp/inserirMedicamento.jsp</result>
                  <result
name="sussess">/jsp/listarMedicamento.jsp</result>
            </action>

            <action name="list" class="medicamentoAction">
                  <result>/jsp/listarMedicamento.jsp</result>
                  <result name="input">/jsp/listarMedicamento.jsp</result>
            </action>

            <action name="saveMedicamento" method="execute"
                  class="medicamentoAction">
                  <result>/jsp/inserirMedicamento.jsp</result>
                  <result name="input">/jsp/inserirMedicamento.jsp</result>
            </action>

            <action name="edit" method="prepare"
                  class="medicamentoAction">
                  <result>/jsp/inserirMedicamento.jsp</result>
                  <result name="input">/jsp/inserirMedicamento.jsp</result>
                  <result
name="sussess">/jsp/listarMedicamento.jsp</result>
            </action>

            <action name="view" method="prepare"
                  class="medicamentoAction">
                  <result>/jsp/visualizarMedicamento.jsp</result>
                  <result
name="input">/jsp/visualizarMedicamento.jsp</result>
            </action>

      </package>
      <package name="grupo" namespace="/grupo" extends="struts-default">

            <action name="remove" method="remove"
                  class="grupoAction">
                  <result>/jsp/listarGrupo.jsp</result>
```

```
            <result name="input">/jsp/listarGrupo.jsp</result>
        </action>

        <action name="save" method="save" class="grupoAction">
            <result>/jsp/listarGrupo.jsp</result>
            <result name="input">/jsp/inserirGrupo.jsp</result>
            <result name="sussess">/jsp/listarGrupo.jsp</result>
        </action>

        <action name="list" class="grupoAction">
            <result>/jsp/listarGrupo.jsp</result>
            <result name="input">/jsp/listarGrupo.jsp</result>
        </action>

        <action name="saveGrupo" method="execute" class="grupoAction">
            <result>/jsp/inserirGrupo.jsp</result>
            <result name="input">/jsp/inserirGrupo.jsp</result>
        </action>

        <action name="edit" method="prepare" class="grupoAction">
            <result>/jsp/inserirGrupo.jsp</result>
            <result name="input">/jsp/inserirGrupo.jsp</result>
            <result name="sussess">/jsp/listarGrupo.jsp</result>
        </action>


        <action name="view" method="prepare" class="grupoAction">
            <result>/jsp/visualizarGrupo.jsp</result>
            <result name="input">/jsp/visualizarGrupo.jsp</result>
        </action>

    </package>

    <package name="categoriaEntradaSaida"
        namespace="/categoriaEntradaSaida" extends="struts-default">

        <action name="save" method="save"
            class="categoriaEntradaSaidaAction">
            <result>/jsp/inserirCategoriaEntradaSaida.jsp</result>
            <result name="input">
                /jsp/inserirCategoriaEntradaSaida.jsp
            </result>
            <result name="sussess">
                /jsp/inserirCategoriaEntradaSaida.jsp
            </result>
        </action>

        <action name="saveCategoria" method="execute"
            class="categoriaEntradaSaidaAction">
            <result>/jsp/inserirCategoriaEntradaSaida.jsp</result>
            <result name="input">
                /jsp/inserirCategoriaEntradaSaida.jsp
            </result>
        </action>

        <action name="view" method="prepare"
            class="categoriaEntradaSaidaAction">
            <result>/jsp/inserirCategoriaEntradaSaida.jsp</result>
            <result name="input">
```

```
                        /jsp/inserirCategoriaEntradaSaida.jsp
                  </result>
            </action>

      </package>


      <package name="localizacao" namespace="/localizacao"
            extends="struts-default">

            <action name="removeLocExt" method="removeLocExt"
                  class="localizacaoAction">
                  <result>/jsp/listarLocExt.jsp</result>
                  <result name="input">/jsp/listarLocExt.jsp</result>
            </action>

            <action name="saveLocExt" method="saveLocExt"
                  class="localizacaoAction">
                  <result>/jsp/listarLocExt.jsp</result>
                  <result name="input">/jsp/inserirLocExt.jsp</result>
                  <result name="sussess">/jsp/listarLocExt.jsp</result>
            </action>

            <action name="listLocExt" method="execute"
                  class="localizacaoAction">
                  <result>/jsp/listarLocExt.jsp</result>
                  <result name="input">/jsp/listarLocExt.jsp</result>
            </action>

            <action name="saveLocExterna" method="execute"
                  class="localizacaoAction">
                  <result>/jsp/inserirLocExt.jsp</result>
                  <result name="input">/jsp/inserirLocExt.jsp</result>
            </action>

            <action name="editLocExt" method="prepareLocExt"
                  class="localizacaoAction">
                  <result>/jsp/inserirLocExt.jsp</result>
                  <result name="input">/jsp/inserirLocExt.jsp</result>
                  <result name="sussess">/jsp/listarLocExt.jsp</result>
            </action>

            <action name="removeLocIntermed" method="removeLocIntermed"
                  class="localizacaoAction">
                  <result>/jsp/listarLocIntermed.jsp</result>
                  <result name="input">/jsp/listarLocIntermed.jsp</result>
            </action>

            <action name="saveLocIntermed" method="saveLocIntermed"
                  class="localizacaoAction">
                  <result>/jsp/listarLocIntermed.jsp</result>
                  <result name="input">/jsp/inserirLocIntermed.jsp</result>
                  <result
name="sussess">/jsp/listarLocIntermed.jsp</result>
            </action>

            <action name="listLocIntermed" method="executeIntermed"
                  class="localizacaoAction">
                  <result>/jsp/listarLocIntermed.jsp</result>
```

```
                  <result name="input">/jsp/listarLocIntermed.jsp</result>
            </action>

            <action name="saveLocIntermediaria" method="executeIntermed"
                  class="localizacaoAction">
                  <result>/jsp/inserirLocIntermed.jsp</result>
                  <result name="input">/jsp/inserirLocIntermed.jsp</result>
            </action>

            <action name="editLocIntermed" method="prepareLocIntermed"
                  class="localizacaoAction">
                  <result>/jsp/inserirLocIntermed.jsp</result>
                  <result name="input">/jsp/inserirLocIntermed.jsp</result>
                  <result
name="sussess">/jsp/listarLocIntermed.jsp</result>
            </action>

            <action name="removeLocInterna" method="removeLocInterna"
                  class="localizacaoAction">
                  <result>/jsp/listarLocInterna.jsp</result>
                  <result name="input">/jsp/listarLocInterna.jsp</result>
            </action>

            <action name="saveLocIntern" method="saveLocInterna"
                  class="localizacaoAction">
                  <result>/jsp/listarLocInterna.jsp</result>
                  <result name="input">/jsp/inserirLocInterna.jsp</result>
                  <result name="sussess">/jsp/listarLocInterna.jsp</result>
            </action>

            <action name="listLocInterna" method="executeInterna"
                  class="localizacaoAction">
                  <result>/jsp/listarLocInterna.jsp</result>
                  <result name="input">/jsp/listarLocInterna.jsp</result>
            </action>

            <action name="saveLocInterna" method="executeInterna"
                  class="localizacaoAction">
                  <result>/jsp/inserirLocInterna.jsp</result>
                  <result name="input">/jsp/inserirLocInterna.jsp</result>
            </action>

            <action name="editLocInterna" method="prepareLocInterna"
                  class="localizacaoAction">
                  <result>/jsp/inserirLocInterna.jsp</result>
                  <result name="input">/jsp/inserirLocInterna.jsp</result>
                  <result name="sussess">/jsp/listarLocInterna.jsp</result>
            </action>

      </package>

      <package name="unidadeConsumidora" namespace="/unidadeConsumidora"
extends="struts-default">

            <action name="remove" method="remove"
                  class="unidadeConsumidoraAction">
                  <result>/jsp/listarUnidadeConsumidora.jsp</result>
                  <result
name="input">/jsp/listarUnidadeConsumidora.jsp</result>
```

```
            </action>

            <action name="save" method="save"
class="unidadeConsumidoraAction">
                  <result>/jsp/listarUnidadeConsumidora.jsp</result>
                  <result
name="input">/jsp/inserirUnidadeConsumidora.jsp</result>
                  <result
name="sussess">/jsp/listarUnidadeConsumidora.jsp</result>
            </action>

            <action name="list" class="unidadeConsumidoraAction">
                  <result>/jsp/listarUnidadeConsumidora.jsp</result>
                  <result
name="input">/jsp/listarUnidadeConsumidora.jsp</result>
            </action>

            <action name="saveUnidade" method="execute"
class="unidadeConsumidoraAction">
                  <result>/jsp/inserirUnidadeConsumidora.jsp</result>
                  <result
name="input">/jsp/inserirUnidadeConsumidora.jsp</result>
            </action>

            <action name="edit" method="prepare"
class="unidadeConsumidoraAction">
                  <result>/jsp/inserirUnidadeConsumidora.jsp</result>
                  <result
name="input">/jsp/inserirUnidadeConsumidora.jsp</result>
                  <result
name="sussess">/jsp/listarUnidadeConsumidora.jsp</result>
            </action>


            <action name="view" method="prepare"
class="unidadeConsumidoraAction">
                  <result>/jsp/visualizarUnidadeConsumidora.jsp</result>
                  <result
name="input">/jsp/visualizarUnidadeConsumidora.jsp</result>
            </action>

      </package>

            <package name="fornecedor" namespace="/fornecedor"
extends="struts-default">

            <action name="remove" method="remove"
                  class="fornecedorAction">
                  <result>/jsp/listarFornecedor.jsp</result>
                  <result name="input">/jsp/listarFornecedor.jsp</result>
            </action>

            <action name="save" method="save" class="fornecedorAction">
                  <result>/jsp/listarFornecedor.jsp</result>
                  <result name="input">/jsp/inserirFornecedor.jsp</result>
                  <result name="sussess">/jsp/listarFornecedor.jsp</result>
            </action>

            <action name="list" class="fornecedorAction">
```

```
                <result>/jsp/listarFornecedor.jsp</result>
                <result name="input">/jsp/listarFornecedor.jsp</result>
            </action>

            <action name="saveFornecedor" method="execute"
class="fornecedorAction">
                <result>/jsp/inserirFornecedor.jsp</result>
                <result name="input">/jsp/inserirFornecedor.jsp</result>
                <result
name="sussess">/jsp/inserirFornecedor.jsp</result>
            </action>

            <action name="edit" method="prepare" class="fornecedorAction">
                <result>/jsp/inserirFornecedor.jsp</result>
                <result name="input">/jsp/inserirFornecedor.jsp</result>
                <result name="sussess">/jsp/listarFornecedor.jsp</result>
            </action>


            <action name="view" method="prepare" class="fornecedorAction">
                <result>/jsp/visualizarFornecedor.jsp</result>
                <result
name="input">/jsp/visualizarFornecedor.jsp</result>
            </action>

            </package>

            <package name="entradaSaidaMedicamento"
namespace="/entradaSaidaMedicamento" extends="struts-default">

            <action name="saveEntrada" method="saveEntradaMedicamento"
class="entradaSaidaMedicamentoAction">
                <result>/jsp/listarEntradaMedicamento.jsp</result>
                <result
name="input">/jsp/inserirEntradaMedicamento.jsp</result>
                <result
name="sussess">/jsp/listarEntradaMedicamento.jsp</result>
            </action>

            <action name="saveSaida" method="saveSaidaMedicamento"
class="entradaSaidaMedicamentoAction">
                <result>/jsp/listarSaidaMedicamento.jsp</result>
                <result
name="input">/jsp/inserirSaidaMedicamento.jsp</result>
                <result
name="sussess">/jsp/listarSaidaMedicamento.jsp</result>
            </action>

            <action name="listEntrada" method="executeEntrada"
class="entradaSaidaMedicamentoAction">
                <result>/jsp/listarEntradaMedicamento.jsp</result>
                <result
name="input">/jsp/listarEntradaMedicamento.jsp</result>
            </action>

            <action name="listEntradaPendente"
method="executeEntradaPendente" class="entradaSaidaMedicamentoAction">

      <result>/jsp/listarEntradasPendentesMedicamento.jsp</result>
```

```
                <result
name="input">/jsp/listarEntradasPendentesMedicamento.jsp</result>
            </action>

            <action name="listSaida" method="executeSaida"
class="entradaSaidaMedicamentoAction">
                <result>/jsp/listarSaidaMedicamento.jsp</result>
                <result
name="input">/jsp/listarSaidaMedicamento.jsp</result>
            </action>

            <action name="autorizarSaida" method="prepareSaida"
class="entradaSaidaMedicamentoAction">
                <result>/jsp/visualizarSaida.jsp</result>
                <result name="input">/jsp/visualizarSaida.jsp</result>
            </action>

            <action name="autorizarSaidaMedicamento"
method="autorizarSaida" class="entradaSaidaMedicamentoAction">

      <result>/jsp/listarSaidaMedicamentoAutorizada.jsp</result>
                <result
name="input">/jsp/listarSaidaMedicamentoAutorizada.jsp</result>
                <result
name="success">/jsp/listarSaidaMedicamentoAutorizada.jsp</result>
            </action>

            <action name="naoAutorizarSaidaMedicamento"
method="naoAutorizarSaida" class="entradaSaidaMedicamentoAction">

      <result>/jsp/listarSaidaMedicamentoAutorizada.jsp</result>
                <result
name="input">/jsp/listarSaidaMedicamentoAutorizada.jsp</result>
                <result
name="success">/jsp/listarSaidaMedicamentoAutorizada.jsp</result>
            </action>

            <action name="listSaidaAutorizada"
method="executeSaidaAutorizada" class="entradaSaidaMedicamentoAction">

      <result>/jsp/listarSaidaMedicamentoAutorizada.jsp</result>
                <result
name="input">/jsp/listarSaidaMedicamentoAutorizada.jsp</result>
            </action>

            <action name="saveEntradaMedicamento" method="executeEntrada"
class="entradaSaidaMedicamentoAction">
                <result>/jsp/inserirEntradaMedicamento.jsp</result>
                <result
name="input">/jsp/inserirEntradaMedicamento.jsp</result>
            </action>

            <action name="saveSaidaMedicamento" method="executeSaida"
class="entradaSaidaMedicamentoAction">
                <result>/jsp/inserirSaidaMedicamento.jsp</result>
                <result
name="input">/jsp/inserirSaidaMedicamento.jsp</result>
            </action>
```

```
            <action name="viewEntrada" method="prepareEntrada"
class="entradaSaidaMedicamentoAction">
                  <result>/jsp/visualizarEntradaMedicamento.jsp</result>
                  <result
name="input">/jsp/visualizarEntradaMedicamento.jsp</result>
            </action>

            <action name="viewEntradaPendente" method="prepareEntrada"
class="entradaSaidaMedicamentoAction">

      <result>/jsp/visualizarEntradaPendenteMedicamento.jsp</result>
                  <result
name="input">/jsp/visualizarEntradaPendenteMedicamento.jsp</result>
            </action>

            <action name="confirmar" method="confirmaEntrada"
class="entradaSaidaMedicamentoAction">

      <result>/jsp/listarEntradasPendentesMedicamento.jsp</result>
                  <result
name="input">/jsp/listarEntradasPendentesMedicamento.jsp</result>
            </action>

            <action name="rejeitar" method="rejeitaEntrada"
class="entradaSaidaMedicamentoAction">

      <result>/jsp/listarEntradasPendentesMedicamento.jsp</result>
                  <result
name="input">/jsp/listarEntradasPendentesMedicamento.jsp</result>
            </action>

      </package>

      <package name="verificaEstoque" namespace="/verificaEstoque"
extends="struts-default">

            <action name="verifica" method="execute"
class="verificaEstoqueAction">
                  <result>/jsp/listarEstoqueBaixo.jsp</result>
                  <result name="input">/jsp/listarEstoqueBaixo.jsp</result>
                  <result
name="sussess">/jsp/listarEstoqueBaixo.jsp</result>
            </action>

      </package>


</struts>
```

ANEXO III - MedicamentoDAO.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap PUBLIC "-//iBATIS.com//DTD SQL Map 2.0//EN"
"http://www.ibatis.com/dtd/sql-map-2.dtd">
<sqlMap namespace="MedicamentoDAO">
	<typeAlias alias="medicamento"
type="br.org.viadigital.viafarmacos.model.dto.Medicamento"/>
	<typeAlias alias="grupo"
type="br.org.viadigital.viafarmacos.model.dto.Grupo"/>
	<typeAlias alias="int" type="java.lang.Integer"/>

	<resultMap id="result-medicamento" class="medicamento">
		<result property="id" column="codigo"/>
		<result property="nome" column="nome"/>
		<result property="grupo.id" column="grupo"/>
		<result property="grupo.nome" column="nome_grupo"/>
		<result property="grupo.descricao" column="desc_grupo"/>
		<result property="descricao" column="descricao"/>
		<result property="similar" column="similar"/>
		<result property="generico" column="generico"/>
		<result property="classificacao" column="classificacao"/>
		<result property="armazenamento" column="forma_armazenamento"/>
		<result property="fabricante" column="fabricante"/>
		<result property="estoqueMin" column="estoque_minimo"/>
	</resultMap>

	<select id="selectMedicamento" resultMap="result-medicamento"
parameterClass="int" resultClass="medicamento">
		select
			m.*, g.nome as nome_grupo, g.descricao as desc_grupo
		from
			medicamento m join grupo_medicamento g on m.grupo =
g.codigo
		where
			m.codigo=#id#
	</select>

	<select id="selectMedicamentos" resultMap="result-medicamento"
parameterClass="medicamento" resultClass="medicamento">
		select
			m.*, g.nome as nome_grupo, g.descricao as desc_grupo
		from
			medicamento m join grupo_medicamento g on m.grupo =
g.codigo
	</select>

	<select id="selectMedicamentosByGrupo" resultMap="result-medicamento"
parameterClass="grupo" resultClass="medicamento">
		select
			m.*, g.nome as nome_grupo, g.descricao as desc_grupo
		from
			medicamento m join grupo_medicamento g on m.grupo =
g.codigo
		where
			m.grupo = #id#
	</select>
```

```
	<select id="selectMaxMedicamento" resultClass="int">
		select
			max(codigo)
		from
			medicamento
	</select>

	<update id="updateMedicamento" parameterClass="medicamento">
		update
			medicamento
		set
			nome = #nome#,
			grupo = #grupo.id#,
			descricao = #descricao#,
			"similar" = #similar#,
			generico = #generico#,
			classificacao = #classificacao#,
			forma_armazenamento = #armazenamento#,
			fabricante = #fabricante#,
			estoque_minimo = #estoqueMin#
		where
			codigo = #id#
	</update>

	<insert id="insertMedicamento" parameterClass="medicamento">

		insert into medicamento (
			codigo,
			nome,
			grupo,
			descricao,
			"similar",
			generico,
			classificacao,
			forma_armazenamento,
			fabricante,
			estoque_minimo
		) values (
			(select CASE when max(codigo) notnull then max(codigo)+1
else 1 end from medicamento),
			#nome#,
			#grupo.id#,
			#descricao#,
			#similar#,
			#generico#,
			#classificacao#,
			#armazenamento#,
			#fabricante#,
			#estoqueMin#
		)
	</insert>

	<delete id="removeMedicamento" parameterClass="int">
		delete from medicamento where codigo = #id#
	</delete>

</sqlMap>
```