**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

**MURILO NUNES ELIAS**

# INFORMATIZAÇÃO DE FARMÁCIAS PÚBLICAS UTILIZANDO SOFTWARE LIVRE

**FLORIANÓPOLIS**
**2007**

**MURILO NUNES ELIAS**

# INFORMATIZAÇÃO DE FARMÁCIAS PÚBLICAS UTILIZANDO SOFTWARE LIVRE

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção de grau de Bacharel em Sistemas de Informação.

Orientador: Prof. José Eduardo De Lucca

**FLORIANÓPOLIS**
**2007**

**MURILO NUNES ELIAS**

# INFORMATIZAÇÃO DE FARMÁCIAS PÚBLICAS UTILIZANDO SOFTWARE LIVRE

Trabalho de conclusão de curso apresentado ao Departamento de Informática e Estatística do Curso de Sistemas de Informação, da Universidade Federal de Santa Catarina como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. José Eduardo De Lucca

Banca examinadora:

_____
Prof. José Eduardo De Lucca

_____
Prof. João Bosco da Mota Alves

_____
Roberto Fasanaro Junior

## DEDICATÓRIA

Dedico este trabalho, primeiramente, a minha família, pelo grande apoio. Aos meus professores pelo grande conhecimento que me passaram durante estes anos e a todos os meus amigos que sempre me incentivaram ao estudo.

"Não espere por uma crise para descobrir o que é importante em sua vida”

Platão

**SUMÁRIO**

## LISTA DE FIGURAS

## LISTA DE SIGLAS

AJAX – Asynchronous Javascript And XML

CSS – Cascading Style Sheets

DAO – Data Access Objects

DTO – Data Transfer Objects

EJBs – Enterprise Java Beans

HTML – HyperText Markup Language

HTTP – Hiper Text Tranfer Protocol

J2EE – Java 2 Platform Enterprise Edition

JDBC – Java Data Base Connectivity

JS – JavaScript

JSP – Java Server Pages

JSP – Java Server Pages

MVC – Model-View-Controller

SQL – Structured Query Language

WWW – World Wide Web

XML – Extensible Markup Language

## RESUMO

Este trabalho visa ajudar no gerenciamento das farmácias públicas brasileiras integradas com o almoxarifado de suas respectivas prefeituras por intermédio de um sistema WEB. O desenvolvimento do sistema conta com a ajuda do projeto via digital, que a alguns anos vem incentivando a construção de soluções para a informatização do setor publico e que requisitou junto a empresa SWFactory em 2004 os levantamentos de requisitos para o desenvolvimento deste trabalho. Espera-se com este trabalho um controle completo das rotinas de distribuição de remédios pelas prefeituras, tais como controle de estoque, entrada, saída, perca, distribuição de remédios em seus respectivos postos de distribuição.



**Palavras-chave:** Governo eletrônico. Via digital. Prefeituras. Farmácias públicas. Postos de saúde. Software livre. Java.

# 1. INTRODUÇÃO

Os atuais municípios do Brasil, de pequeno e médio porte, encontram-se em situações extremamente difíceis para efetuar investimentos de grande porte em suas infra-estruturas. Principalmente no que diz respeito à informatização, é importante se ter o mínimo de controle na administração interna nas prefeituras.

O projeto via digital foi desenvolvido com o intuito de disponibilizar software livres para as prefeituras por meio de um repositório. Desse modo, qualquer pessoa, com um pouco de conhecimento em informática, pode desenvolver um componente para prefeituras, integrando-o a outros.

As farmácias públicas, muito importantes para o amparo social nos municípios, ajudam toda a população na distribuição de remédios indicados pelos médicos. A implementação de um componente para facilitar os processos de distribuição de uma farmácia pública, com base nos requisitos levantados pela SWFactory em 2004, e a validação do projeto é a proposta deste trabalho de conclusão de curso.

## 1.1. Justificativa

A atual distribuição de remédios em farmácias públicas e postos de saúde de municípios de médio e de pequeno porte necessita de um sistema integrado com os almoxarifados da prefeitura e dos hospitais públicos.

Esse sistema integrado com outros componentes teria a capacidade de aumentar o controle de estoque como, por exemplo, verificar quais postos e farmácias estão com estoques baixos e providenciar uma recarga.

O sistema controlará a entrada, saída, estorno e perda, de qualquer natureza, de todos os medicamentos distribuídos à população nas Farmácias Públicas e/ou Postos de Saúde, por meio do controle de lote e validade de cada medicamento movimentado.

Também evitaria corrupção, pois haveria possibilidade de verificar se o paciente, que adquiriu os remédios, realmente teve uma consulta marcada com o médico nos hospitais municipais ou postos de saúde. Esse procedimento reduziria os gastos públicos com a distribuição de remédio que, no ano de 2007, pode chegar a R$ 4,6 bilhões de reais.

## 1.2. Objetivos

### 1.2.1. Geral

Disponibilizar uma solução para farmácias públicas integrável a outros sistemas (da mesma família), dinamizando o atendimento e a gestão desses locais.

### 1.2.2. Específicos

- Fazer com que farmácias públicas tenham este componente como uma opção de controle administrativo.
- Colaborar com o projeto Via Digital.
- Ampliar os conhecimentos deste autor no desenvolvimento de aplicações WEB.

# 2. FUNDAMENTAÇÃO TEÓRICA

Foram estudados diversos projetos para o desenvolvimento deste, alguns deles foram descartados, mas os que demonstraram ser de uma boa utilidade, foram utilizados e são descritos a seguir.

## 2.1. Via Digital

Via Digital é um projeto cujo objetivo é servir de referência para viabilizar a informatização de prefeituras com software livre, permitindo o acesso e o desenvolvimento compartilhado de componentes para a gestão municipal.

Com o repositório Via Digital, é possível disponibilizar software livre para todos que desejam utilizar os componentes já desenvolvidos. Por ser livre a todos, é possível ter acesso ao código fonte para se efetuarem as alterações desejadas e adaptá-lo a algum sistema já existente. Além disso, o sistema permite contribuir com o componente, desenvolvendo novas funcionalidades.

Este sistema desenvolvido para gerenciamento de farmácias públicas fará parte do repositório da Via Digital. Desse modo, todas as prefeituras podem adquiri-lo gratuitamente por intermédio do portal. [via digital]

## 2.2. Software livre

Software livre é qualquer código de programação que pode ser usado, copiado, estudado, modificado e redistribuído. A principal vantagem de um sistema

ser software livre é que mais pessoas podem se dedicar ao projeto, aumentando, dessa forma, suas funcionalidades e realizando um verdadeiro “mutirão” por uma causa. [gnu]

## 2.3. Java

Java, atualmente, é uma linguagem muito difundida em todo o mundo. Sua principal vantagem é que seus sources não são compilados para um código nativo e sim para um bytecode que é executado por uma máquina virtual que pode ser rodada em diferentes plataformas.

A escolha desta linguagem para este projeto foi feita devido ao grande suporte que dispõe para sistemas web, com frameworks e ferramentas extremamente robustas, facilitando, assim, o desenvolvimento e a integração com outros sistemas. [sun]

## 2.4. Frameworks

Framework em desenvolvimento orientado a objetos, é uma biblioteca de classes que ajudam na flexibilidade e extensão no desenvolvimento de um sistema. A seguir são descritos os frameworks utilizados neste projeto.

### 2.4.1. Struts

Struts é um framework baseado para a WEB que implementa a arquitetura MVC (ver item 2.7). Com ele, pode-se controlar, de forma mais simples, o fluxo da aplicação, facilitando na montagem de telas, captação de informações e caso seja necessário, a implementação de outra forma de visualização, além da HTML.

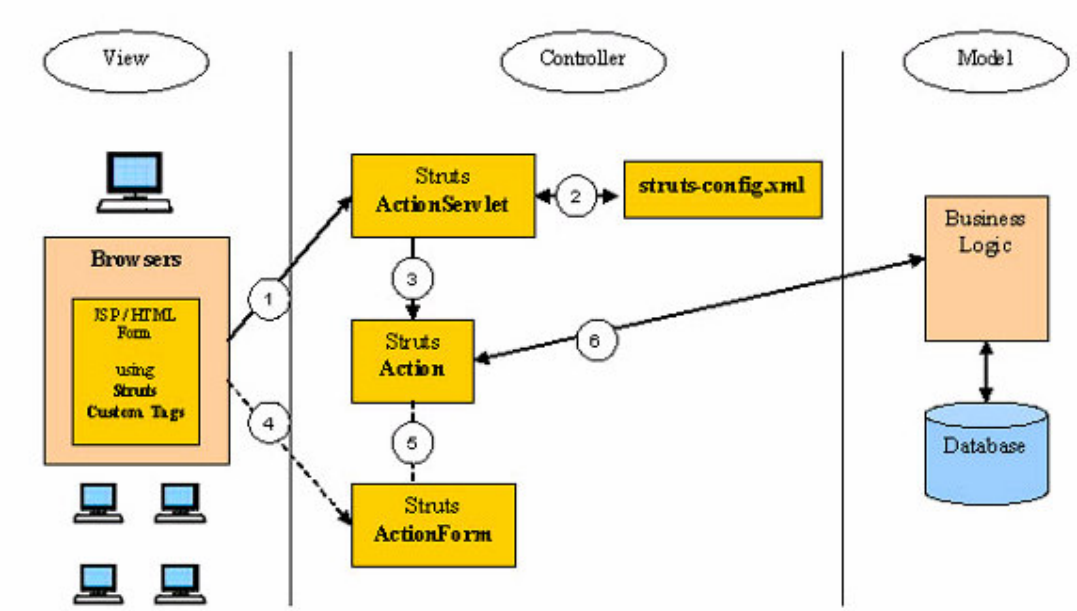


Figura 1 - Fluxo de uma aplicação usando Struts

Fonte:http://www.javafree.com.br/dependencias/tutoriais/struts2/struts_goetten_html_2bd23280.png

Como demonstra a figura 1, a visão será controlada por intermédio dos Actions que são previamente configurados no arquivo struts-config.xml. São essas funções e métodos que se comunicam com a camada de negócio que fará todo o controle do sistema. [struts]

### 2.4.2. Spring

Spring é um framework capaz de tornar uma aplicação totalmente plugável sem precisar parar a execução, tem suporte ao Ibatis (ver item 2.8.2) e Spring (ver item 2.4.1) permitindo com que trabalhe sincronizadamente com estes frameworks que também são usados neste projeto. Possibilita também controlar as instâncias dos objetos, de modo que são criados apenas uma vez, reutilizando assim os recursos já inicializados. [spring]

## 2.5. SGBD - PostgreSQL

O PostgreSQL é um SGBD de código aberto. Possibilita consultas complexas, chaves estrangeiras, integridade transacional, controle de concorrência multi-versão, suporte ao modelo híbrido objeto-relacional, além de apresentar outras funcionalidades que são padrões em outros SGBDS como triggers, view, etc.

Foi escolhido para este projeto também por seguir a definição técnica do Via Digital para que outras aplicações dependentes possam ter acesso a algum dado do sistema. [postgresql]

## 2.6. Interface

A seguir, são descritas as tecnologias usadas para o desenvolvimento da interface.

### 2.6.1. Servlet JSP

Tecnologia baseada na linguagem Java que permite o código se misturar à html para demonstração de conteúdo dinâmico, é similar a ASP e PHP. Com ela, podem-se imprimir na tela as informações solicitadas e retornadas pelo struts. [sun]

### 2.6.2. HTML

HTML é uma linguagem de marcação usada para a visualização de conteúdo das páginas WEB. É o limite entre o usuário e o sistema desenvolvido, é neste formato que é demonstrado o conteúdo dinâmico. [w3c]

### 2.6.3. CSS

CCS será a linguagem utilizada para definir os estilos da html. Com ela, pode-se criar um link em uma html, e setar para que o conteúdo siga o padrão de uma classe pré-definida no arquivo css. Facilitará muito na padronização e manutenção do layout do sistema.[css]

### 2.6.4. JavaScript

JavaScript é uma linguagem de programação interpretada pelo navegador que é usada em no sistema, basicamente, para validar formulários antes que o usuário poste, melhorando a usabilidade do sistema. [ecma]

## 2.7. Padrões MVC

MVC é uma forma de arquitetar um sistema, de modo que os dados (Model) fiquem separados da interface (View), sendo que as regras de negócio que precisam eventualmente ser realizadas fiquem sobre a responsabilidade do controlador. (Controller).

### 2.7.1. DAO

Data Access Object é um padrão para persistir um ou mais objetos ao SGBD. Neste projeto, usa-se o ibatis (ver item 2.8.2) para seguir este padrão.[sun]

### 2.7.2. DTO

Objeto de transferência de dados é um padrão utilizado normalmente para realizar transações distribuídas e é por intermédio dele que serão acessados os objetos remotos.

Neste projeto, serão necessários objetos de outros componentes como, por exemplo, o de autenticação de usuário, usando o DTO. [sun]

### 2.7.3. VO

Value Object é um padrão para se trabalhar com objetos geralmente persistidos do banco, as classes VO contêm os atributos dos objetos. Dessa forma,

é possível passar os objetos como parâmetro de métodos e funções, guardando a referência do VO. [sun]

## 2.8. Ferramentas

Segue a descrição das ferramentas utilizadas para o desenvolvimento deste projeto.

### 2.8.1. Eclipse

Eclipse é uma IDE muito famosa de código livre desenvolvida em JAVA. Tem em seu contexto suporte a vários plugins para diferentes funcionalidades, como editores para java, javascript, css, html, integração com o SGBD e CVS. É a principal ferramenta usada para desenvolver este sistema. [eclipse]

### 2.8.2. Ibatis

Framework usado para o mapeamento objeto relacional de uma forma simples e flexível. Por meio de um xml, as querys são declaradas e, utilizando-as, o framework trabalha para mapear o objeto na base. Como o banco do sistema é relacional e a metodologia de desenvolvimento é orientada a objetos, é o ibatis que faz a conversão de um para outro ou vice-versa.[ibatis]

### 2.8.3. Apache tomcat

Servidor de aplicação Java para web, no qual é armazenada a aplicação para o usuário poder acessar com seu navegador. Este container é open source e possibilita diversas configurações através de arquivos xml para customizar uma aplicação. [apache]

### 2.8.4. CVS

É um sistema de controle de versão que permite o desenvolvimento simultâneo de uma classe por diferentes programadores. Mantém no servidor todas as versões dos arquivos e um log com as modificações ocorridas. [cvs]

# 3. ESTADO DA ARTE

## 3.1. Levantamento de softwares relacionados

A pesquisa de softwares relacionados ou com funcionalidades semelhantes ao sistema proposto foi realizado na Internet em sites de busca, fóruns e indicados por pessoas que atuam na área.

## 3.2. Avaliação de softwares relacionados

### 3.2.1. CARE2X

CARE2X é um software inteligente para hospitais e organizações de cuidados médicos. É distribuído com a licença GNU e composto por diversas funcionalidades que ajudam no gerenciamento de hospitais, tais como, cadastro de pacientes, marcação de consulta, lista de médicos, entre outros. [CARE2X]

O software foi desenvolvido na linguagem PHP e pode utilizar diversos tipos de bancos de dados que a linguagem suporta. O sistema roda em servidor apache, é de simples instalação e necessita apenas de um navegador instalado para que o usuário possa utilizá-lo.

Analisando o módulo “farmácia”, que tem uma maior relação com este projeto, foi possível levantar as seguintes funcionalidades:

- Criar, enviar e receber uma lista de requisição de medicamentos de diferentes departamentos.

- Criar, editar ou apagar catálogos de medicamentos.
- Arquivamento de requisições antigas.
- Criar, editar ou apagar produtos da farmácia.
- Inserção de notícias referentes à farmácia.

Apesar de ajudar no controle de requisições de medicamentos de outros departamentos do sistema care2x, o módulo farmácia deixa a desejar quanto ao gerenciamento de estoque dos produtos e distribuição de remédio para pacientes.

### 3.2.2. HOSPUB

Hospub é o sistema hospitalar oficial do Ministério da Saúde para gerenciamento, gestão e controle social do Sistema Único de Saúde (SUS). O sistema é composto por vários módulos que podem ser integrados, compartilhando, desse modo, informações necessárias para um processo do dia-a-dia.

Entre estes módulos, existem dois que se relacionam com este projeto, chamados almoxarifado e farmácia. Eles apresentam vários requisitos como:

- Entrada e saída de medicamentos.
- Determinação de parâmetros de ressuprimento.
- Registro de nota fiscal por laboratório de fornecedor.
- Cadastro de laboratórios fornecedores.
- Codificação de medicamento pela tabela oficial do governo (SIAFI).
- Registro de receita médica por profissional requisitante com código SIAPE.
- Consulta, edição e exclusão de catálogos de medicamentos.
- Consulta de consumo por centro de custo/setor.

- Consulta de estoques.
- Consulta a tabela de SIAFI.
- Diversos relatórios.

Apesar de o módulo ter muitas funcionalidades e ser bem completo, foi desenvolvido com tecnologias que atualmente se encontram obsoletas, dificultando sua manutenção. Sua interface é via console, dificultando a usabilidade dos usuários e elevando o tempo para realizar tarefa simples. [datasus]

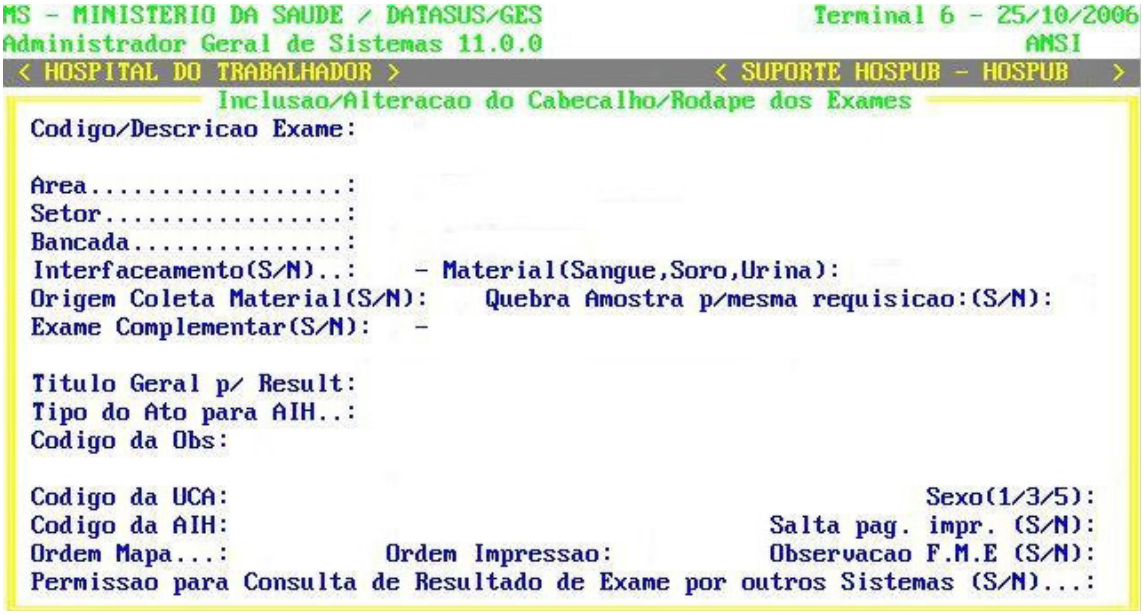


Figura 2 - Exemplo da interface HOSPUB

Fonte: http://w3.datasus.gov.br/hospub/arquivos/Interfaceamento.pdf

# 4. SISTEMA PROPOSTO

Os requisitos do sistema proposto foram documentandos pela SWFactory, empresa contratada pelo projeto Via Digital para descrever os requisitos e casos de uso.

## 4.1. Sistema de controle da farmácia

O módulo farmácia tem como objetivo o controle da entrada, saída, estorno e perda, de qualquer natureza, de todos os medicamentos distribuídos à população nas Farmácias Públicas e/ou Postos de Saúde, por meio do controle de lote e validade de cada medicamento movimentado.

A entrada de medicamentos será proveniente da CAF (módulo almoxarifado ver item 4.5) e será realizada automaticamente por intermédio da rede de computadores.

Depois de cadastrada a entrada, os medicamentos passam a ser distribuídos para a população, e todos os registros de saída de medicamentos serão feitos no computador. Cada registro conterá as informações do cidadão, do profissional de saúde solicitante e dos medicamentos distribuídos. Informará também a dosagem/quantidade desses medicamentos, com o objetivo de controlá-los para evitar, assim, que o usuário retorne para pegar medicamentos sem necessidade.

Abaixo pode-se visualizar o fluxo do sistema:

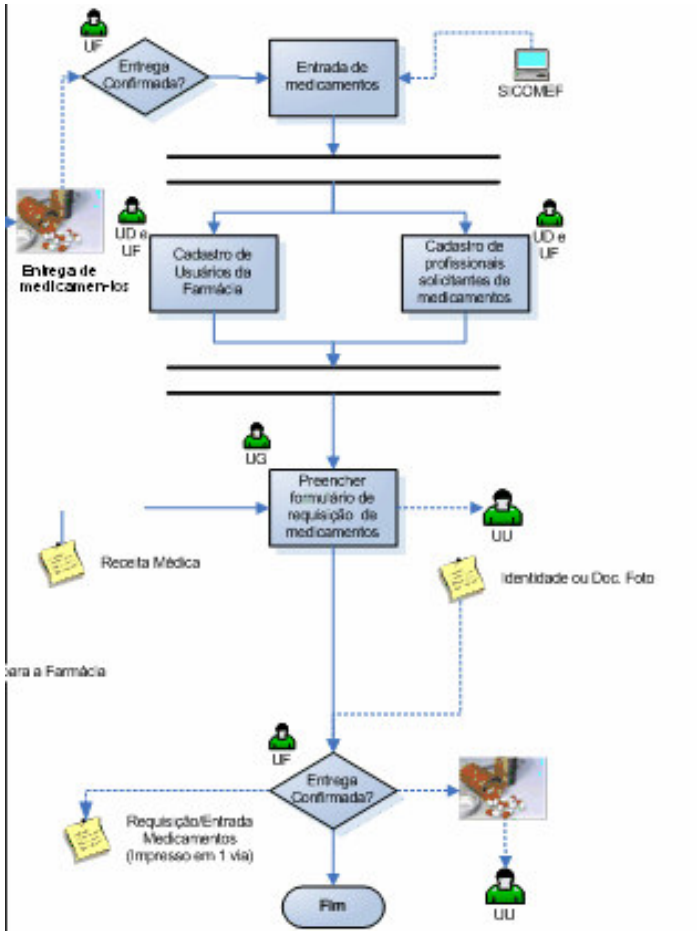


Figura 3 - Representações dos fluxos da farmácia pública elaborado pela SWFactory

Fonte: https://repositorio.viadigital.org.br/frs/download.php/81/SantaClara.zip

## 4.2. Funcionalidades gerais

As funcionalidades gerais são usadas em todos os módulos que fazem parte do projeto via digital. O desenvolvimento dessas funcionalidades não faz parte do escopo deste trabalho. Entretanto, serão listadas abaixo para facilitar na visão geral do sistema:

- Inserir Usuário do Sistema
- Alterar Usuário do Sistema
- Remover Usuário do Sistema
- Visualizar Usuário do Sistema
- Inserir Perfil de Usuário do Sistema
- Alterar Perfil de Usuário do Sistema
- Remover Perfil de Usuário do Sistema
- Visualizar Perfil de Usuário do Sistema
- Entrar no Sistema
- Sair do Sistema
- Alterar Senha do Usuário do Sistema
- Relatório/Listagem de Todos os Usuários do Sistema
- Relatório/Listagem dos Usuários do Sistema por Perfil
- Relatório/Listagem de Todos os Perfis
- Relatório/Listagem dos Medicamentos por Grupos de Medicamentos
- Relatório/Listagem dos Grupos de Medicamentos

## 4.3. Casos de usos relacionados ao sistema da farmácia

Os casos de usos foram documentados pela SWFactory a pedido do Via Digital, abaixo você pode visualizar o diagrama.

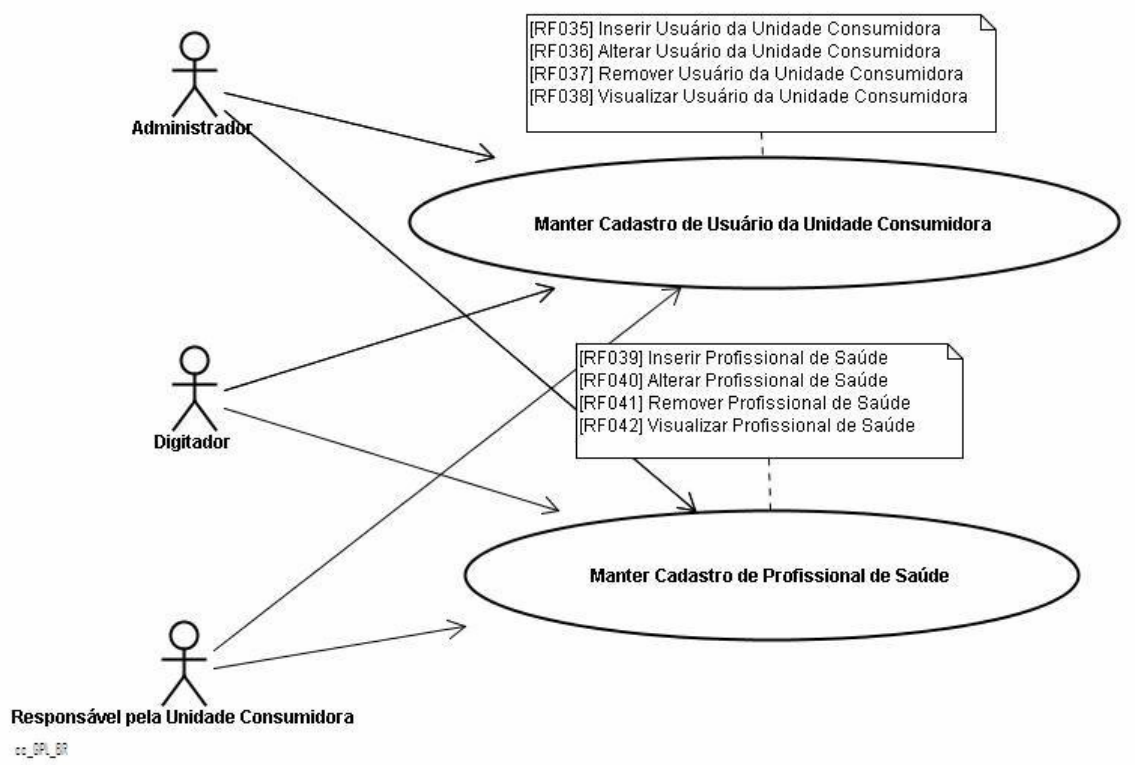


Figura 4 - Diagrama de caso de uso – cadastros relacionados ao ambiente das farmácias públicas elaborado pela SWFactory

Fonte: https://repositorio.viadigital.org.br/frs/download.php/81/SantaClara.zip

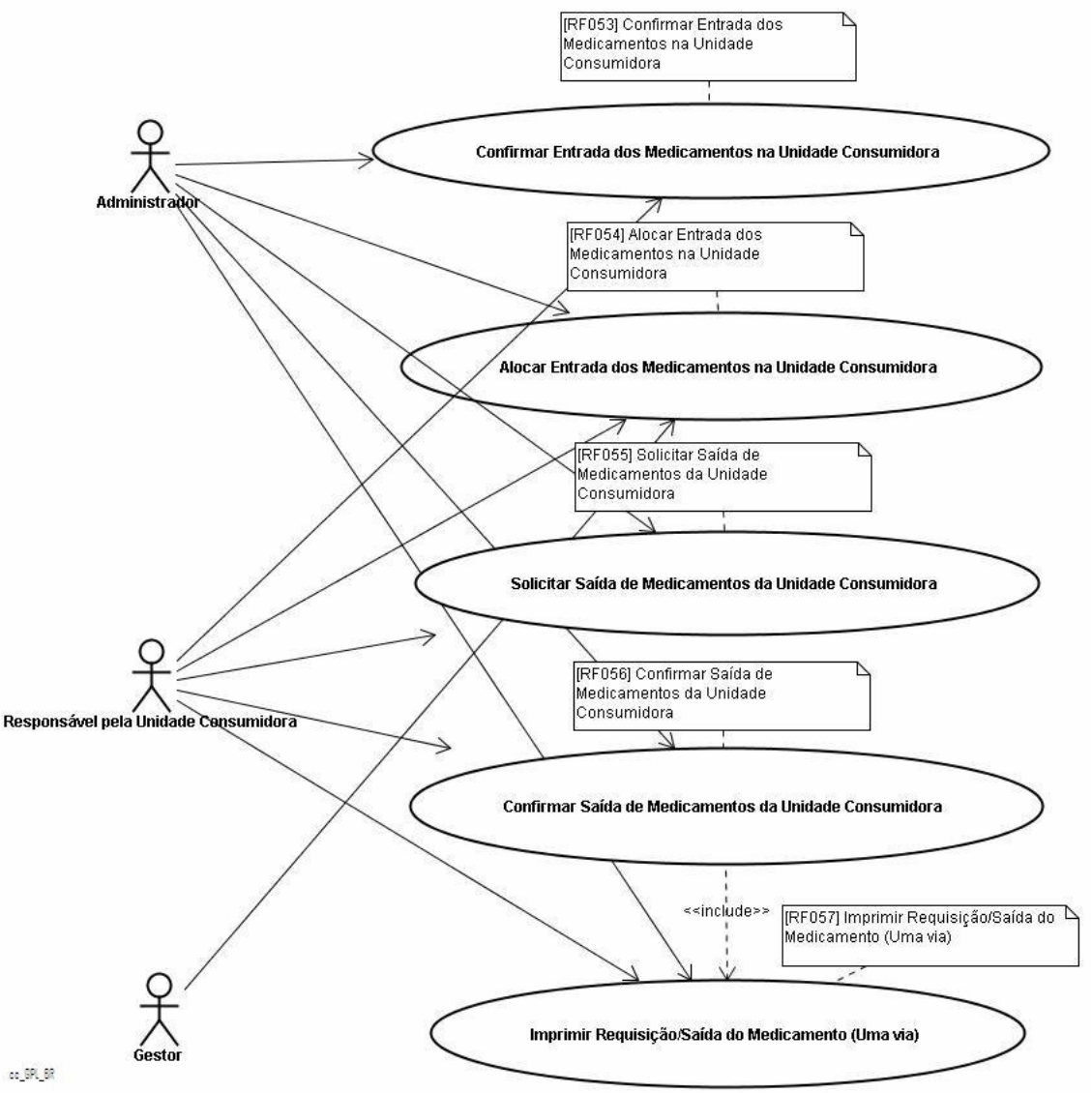


Figura 5 - Diagrama de caso de uso – funcionalidades relacionadas ao ambiente das farmácias elaborado pela SWFactory.

Fonte: https://repositorio.viadigital.org.br/frs/download.php/81/SantaClara.zip

Todos os requisitos mostrados acima são descritos detalhadamente no anexo deste trabalho, algumas alterações na documentação original da SWFactory foram realizadas para que o desenvolvimento fosse possível.

## 4.4. Sistema almoxarifado ou CAF

O sistema almoxarifado é responsável pela distribuição dos remédios para as unidades consumidoras (farmácias públicas). Portanto, o cadastro de medicamentos, unidade consumidora, fornecedores, entrada de medicamentos no CAF e saída para as unidades consumidoras fica sob a responsabilidade do sistema de controle da farmácia.

O sistema de farmácia terá apenas alguns relacionamentos com o sistema de almoxarifado. Como exemplo, pode-se citar a saída de medicamentos do CAF para as unidades consumidoras.

## 4.5. ARQUITETURA DO SISTEMA

### 4.5.1. Camada view

A camada view do sistema proposto foi elaborada em conjunto com as tecnologias JSP, CSS, Javascript(JS) e AJAX. As tecnologias JSP, CSS e JS interagirão e, com a ajuda do tomcat, formarão uma página html que pode ser acessada pelos usuários do sistema. O usuário, ao realizar alguma ação na página html, invocará o ajax que, por sua vez, passa a ação ao controller.

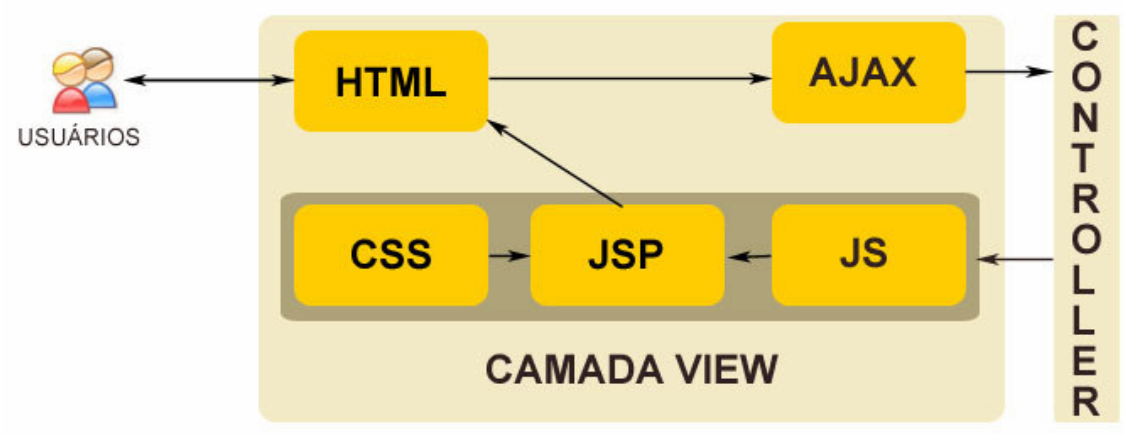


Figura 6 - Arquitetura da camada view do sistema.

No anexo deste trabalho existe um um exemplo de um JSP e como ele será apresentado ao se transformar em html.

### 4.5.2. Camada controller

Para facilitar o desenvolvimento, a camada controller foi desenvolvida com o framework struts. Por meio do struts, controlam-se as ações do sistema chamadas pela página html.

Por exemplo, quando se quer listar os profissionais de saúde, a página principal do sistema chama a ação 'profissionais/list.action'. Ao receber essa ação, o sistema executa a função 'execute' da classe ProfissionalAction.java que, por sua vez, vai contatar o model para receber os objetos dos profissionais cadastrados no sistema.

Após coletados os dados dos profissionais, as informações são visualizadas no JSP 'listarProfissionalSaude.jsp'. Abaixo, pode-se visualizar o código do controlador desta funcionalidade no sistema:

No arquivo de configuração struts.xml:

```
<package name="profissionais" namespace="/profissionais"   extends="struts-
default">
   <action name="list" class="profissionaisAction" method="executeInterna">
      <result>/jsp/listarProfissionalSaude.jsp</result>
      <result name="input">/jsp/ listarProfissionalSaude.jsp</result>
   </action>
</package>
```

Na classe ProfissionalAction.java:

```
public String execute() {
   this.profissionais = business.selectProfissionais();
   return ActionSupport.INPUT;
}
public List<Profissionais> getProfissionais() {
   return profissionais;
}
```

Como é possível ver na função execute(), é chamada a função selectProfissionais da classe business; é nesse momento que a aplicação passa a execução para o model. Após retornar com os objetos do model, a função List é executada, retornando todos os profissionais de saúde e exibidos no JSP listrarProfissionalSaude.jsp.

### 4.5.3. Camada model

Esta camada foi desenvolvida com os frameworks Spring e Ibatis. Estas tecnologias facilitam no reuso de códigos, gerenciamento de transações, dependências de objetos e no mapeamento dos objetos relacionais e vice-versa.

### 4.5.3.1. Camada de negócio

Na camada de negócio, trata-se os objetos e suas dependências caso seja necessário para inserir, editar, listar, excluir no banco de dados.

No exemplo do controller anterior, foi citado que a execução do sistema continuava na camada model, que era chamada na função 'bussness.selectProfissionais()':

```
public List selectProfissionais() {
   return profissionalDAO.selectProfissionais();

}
```

Como nesta ação quer-se listar os profissionais de saúde que neste caso não excluí qualquer dependência, a função apenas chama a função selectProfissionais da classe profissionalDAO.

```
public List selectProfissionais () {
   return
super.getSqlMapClientTemplate().queryForList("selectProfission
ais");
}
```

Esta classe interage com o framework spring e Ibatis, neste exemplo a função irá montar a lista de objetos dos profissionais de saúde com a ajuda da classe dto Profissional.java, que contem os getters e setters doa atributos do objeto, exemplo:

```
public String getNome() {
   return nome;
}
```

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

### 4.5.3.2. Camada de persistência Ibatis

Esta camada cuida do mapeamento dos objetos relacionais; ela transforma as informações do sistema para o banco de dados e vice-versa. No exemplo da camada de negócio, foi citado o método 'queryForList("selectProfissionais")'; o ibatis para transformar as informações dos profissionais que estão no banco de dados em objetos precisa de um mapeamento feito em xml:

```
<select id="selectProfissionais" resultMap="result-profissional"
parameterClass="profissional" resultClass="profissional">
   select
      codigo, nome, identificacao, cpf , cargo
   from
      profissionalsaude_medicamento
   order by nome
</select>
```

O XML contém um SQL que é passado ao banco de dados a fim de executar a query e retorne as informações. Essas informações são mapeadas no dto profissional. Para isso, é necessário descreverem-se os atributos no XML também:

```
<typeAlias alias="profissional"
type="br.org.viadigital.viafarmacos.model.dto.Profissional"/>

<resultMap id="result-profissional" class="profissional">
   <result property="id" column="codigo"/>
   <result property="nome" column="nome"/>
   <result property="identificacao" column="identificacao"/>
   <result property="cpf" column="cpf"/>
   <result property="cargo" column="cargo"/>
```

</resultMap>

## 4.6. Desenvolvimento

### 4.6.1. Configurações

Os arquivos de configurações encontram-se na pasta /WebContent/WEB-INF do sistema, são eles:

- aplication.properties: eventuais parâmetros usados no sistema e requiridos pelo spring.
- aplicationContext.xml: descrição dos beans utilizados pelo spring.
- Jdbc.properties: configuração dos parâmetros de configuração do banco.
- Sqlmap-config.xml: mapeamento dos xmls do Ibatis.
- Struts.xml: descrição das ações do struts.
- Viafarmacos-defaut.xml: arquivo que descreve as verificações para uma determinada ação.
- web.xml: configuração da aplicação no servidor, como página de erro e tempo de seção.

### 4.6.2. Página JSP

Páginas responsáveis pela montagem do html ao usuário:

- inserirUsuarioUnidadeConsumidora.jsp: página com formulário de inserção/edição de um usuário da unidade consumidora.

- inserirProfissionalSaude.jsp: página com formulário de inserção/edição de um profissional de saúde.
- visualizarUsuarioUnidadeConsumitora.jsp: visualização de um usuário da unidade consumidora.
- visualizarProfissionalSaude.jsp: visualização de um profissional de saúde.
- confirmarEntradaMedimentoUnidade.jsp: confirmar Entrada dos Medicamentos na Unidade Consumidora.
- alocarEntradaMedicamentoUnidade.jsp: alocar Entrada dos Medicamentos na Unidade Consumidora.
- solicitarSaidaMedicamentoUnidade.jsp: solicitar Saída de Medicamentos da Unidade Consumidora.
- confirmarSaidaMedicamentoUnidade.jsp: confirmar Saída de Medicamentos da Unidade Consumidora.
- imprimirRequisicaoMedicamento.jsp: imprimir Requisição/Saída do Medicamento.
- listarUsuarioUnidadeConsumidora.jsp: Relatório/Listagem dos Usuários de uma Unidade Consumidora.
- listarProfissionaisSaude.jsp: Relatório/Listagem dos Profissionais de Saúde.
- listarEntradaPedentesMedicamentosUnidade.jsp: Relatório/Listagem de Entrada Pendentes de Medicamentos na Unidade Consumidora (por Grupo de Medicamentos).

- listarEntradaConfirmadaMedicamentosUnidadeGM.jsp: Relatório/Listagem de Entrada Confirmada de Medicamentos na Unidade Consumidora (por Grupos de Medicamentos).
- listarEntradaAlocadasMedicamentosUnidade.jsp: Relatório/Listagem das Entradas Alocadas de Medicamentos na Unidade Consumidora (por Grupo de Medicamentos entre um Intervalo de datas).
- listarMedicamentoEstoqueUnidade.jsp: Relatório/Listagem dos Medicamentos do Estoque de uma Unidade Consumidora (por Grupo de Medicamentos).
- listarMedicamentoSaidaConfirmada.jsp: relatório/Listagem dos Medicamentos com Saída Confirmada no Total (por unidades consumidoras e entre um Intervalo de Datas).
- listarMedicamentoDistribuidoUsuario.jsp: Relatório/Listagem dos Medicamentos Distribuídos (por Usuário da Unidade Consumidora entre um Intervalo de Datas).
- listarMedicamentoDistribuidoProfissional.jsp: Relatório/Listagem dos Medicamentos Distribuídos (por Profissionais de Saúde entre um Intervalo de Datas).
- listarMedicamentoDistribuidoGestor.jsp: Relatório/Listagem dos Medicamentos Distribuídos (por Usuários Gestores do Sistema entre um Intervalo de Datas).
- listarMedicamentoUnidadeExpirar.jsp: Relatório/Listagem de Medicamentos da Unidade Consumidora que estão para Expirar a Validade (Especificando o Número de Dias).

- listarMedicamentoUnidadeVencido.jsp: Relatório/Listagem de Medicamentos da Unidade Consumidora que Ultrapassaram o Prazo de Validade

### 4.6.3. Actions

Ações que podem ser realizadas no componente e estão especificadas no struts.xml.

Ações relacionadas ao usuário da unidade consumidora:

- inserir: Mostra o formulário para inserção de um novo usuário consumidor;
- list: Lista todos os usuários da unidade consumidora;
- edit: Edita um usuário da unidade consumidora;
- save: Salva um usuário preenchido em um formulário;
- remove: Remove um usuário da unidade consumidora;
- view: Visualiza um usuário da unidade consumidora;

Ações relacionadas aos profissionais de saúde:

inserir: Mostra o formulário para inserção de um novo profissional de saúde;

- list: Lista todos os profissionais de saúde;
- edit: Edita um profissional de saúde;
- save: Salva um profissional de saúde;
- remove: Remove um profissional de saúde;
- view: Visualiza um profissional de saúde.

Ações relacionadas à entrada de medicamento:

- confirmarEntradaMedicamento: Ação que confirma a entrada de um grupo de medicamento na unidade consumidora;
- alocarEntradaMedicamento: Esta ação define uma localização dos lotes do medicamento no almoxarifado da unidade consumidora;
- listEntradaPedenteMedicamento: Ação que lista todas as entradas de medicamentos pendentes;
- listEntradaConfirmadaMedicamento: Lista todas as entradas de medicamentos confirmada;
- listEntradaAlocadaMedicamento: Lista todas as entradas de medicamentos alocadas.

Ações relacionadas à saída de medicamento:

- solicitarSaidaMedicamento: Ação que solicita a saída de um medicamento a um usuário da unidade consumidora pelo usuário gestor;
- confirmarSaidaMedicamento: Confirma a saída de medicamento ao usuário da unidade consumidora;
- imprimirSaidaMedicamento: Imprime a saída de medicamento;
- listMedicamentoSaidaConfirmada: Lista todas as saídas confirmadas dos medicamentos;
- listMedicamentoDistribuidosUsuario: Lista todos os medicamentos distribuídos por usuário da unidade consumidora;
- listMedicamentoDistribuidosProfissional: Lista todos os medicamentos distribuídos por profissionais da saúde.

- listMedicamentoDistribuidoGestor: Lista todos os medicamentos distribuídos por usuário gestor.

Ações relacionadas ao estoque:

- listMedicamentoEstoqueUnidade: Lista todo o estoque dos medicamentos da unidade consumidora;
- listMedicamentoExpirar: Lista todos os medicamentos do estoque cuja validade está para expirar;
- listMedicamentoVencido: Lista todos os medicamentos vencidos do estoque.

### 4.6.4. Classes de negócio

As classes de negócio desenvolvidas para o módulo da farmácia são as seguintes:

- UsuarioUnidadeConsumidoraBusiness: Responsável pela execução de todas as regras de negócio relacionadas com o usuário da unidade consumidora.
- ProfissionalSaudeBusiness: Responsável pela execução de todas as regras de negócio relacionadas com o profissional de saúde.
- EntradaSaidaMedicamentoBusiness: Responsável pela execução de todas as regras de negócio relacionadas à entrada e saída de medicamentos, consulta de estoque, relatórios.

### 4.6.5. Classes de persistência

Classes do módulo farmácia que definem a persistência dos dados no banco de dados:

- UsuarioUnidadeConsumidoraDAO: Especifica às querys disponíveis que persistem no banco de dados os objetos relacionados aos usuários da unidade consumidora.
- ProfissionalSaudeDAO: Especifica às querys disponíveis que persistem no banco de dados os objetos relacionados aos profissionais de saúde.
- EntradaSaidaMedicamentoDAO: Especifica às querys disponíveis que persistem no banco de dados os objetos relacionados aos medicamentos.

### 4.6.6. Arquivos XML de persistência

Estes XMLs mapeam como deve ser a SQL para uma determinada função da classe de persistência:

- UsuarioUnidadeConsumidora.xml: Definição das SQLs como insert, update, select e delete para os usuários da unidade consumidora.
- ProfissionalSaude.xml: Definição das SQLs como insert, update, select e delete para os profissionais de saúde da unidade consumidora.
- EntradaSaidaMedicamentoDAO.xml: Definição das SQLs como insert, update, select e delete para as funcionalidades relacionadas aos medicamentos.

# 5. CONCLUSÃO

O desenvolvimento do sistema para gerenciamento de farmácias públicas com integração com os almoxarifados de prefeituras foi muito ágil com a ajuda dos frameworks utilizados.

Com o auxílio do servidor de aplicação CVS, disponibilizado pelo projeto Via Digital, o desenvolvimento deste sistema em conjunto com o do almoxarifado de prefeituras foi possível e facilitou a integração entre os dois, fundamental para que muitas funcionalidades de ambos os sistemas funcionem perfeitamente.

O sistema estará pronto para a utilização em todas as prefeituras do país que desejam controlar melhor a distribuição de medicamento em seus postos de saúde. O controle sistemático evita o desperdício, a perda de medicamentos e, principalmente, permite economia do dinheiro público.

Em todos os postos de saúde visitados neste ano (2007), com o objetivo de verificar o processo de distribuições de medicamentos em Florianópolis, observou-se a falta de informatização.

Com base neste conhecimento, sugere-se, como trabalho futuro, mostrar às prefeituras as vantagens da implantação deste sistema e convencê-las a adotá-lo. Desse modo, implantando este sistema em todos os postos de saúde e dando capacitação para os usuários que irão geri-lo terão/terá um ótimo retorno pelo baixo investimento.

# 6. REFERÊNCIAS

CARE2X. Disponível em: <http://www.care2x.org/>. Acesso em: 23 jul. 2007.

CVS. Disponível em: <http://savannah.nongnu.org/projects/cvs/>. Acesso em: 23 jul. de 2007.

DATASUS. Disponível em: <http://www.datasus.gov.br/>. Acesso em: 23 jul. 2007.

ECLIPSE. Disponível em: <http://www.eclipse.org>. Acesso em: 19 jun. 2007.

ECMA. Disponível em:
<http://www.ecma-international.org/publications/standards/Ecma-262.htm>. Acesso em: 23 jul. 2007.

GNU. Disponível em: <http://www.gnu.org/home.pt.html>. Acesso em: 23 jul. 2007.

IBATIS. Disponível em: <http://ibatis.apache.org/>. Acesso em: 01 out. 2007.

JAVA. Disponível em: <http://sun.com>. Acesso em: 30 jun. 2007.

POSTGRESQL. Disponível em: <http://www.postgresql.org>. Acesso em: 19 jun. 2007.

SPRING. Disponível em : <http://www.springframework.org>. Acesso em: 29 jun. 2007.

STRUTS. Disponível em: <http://struts.apache.org>. Acesso em: 19 jun. 2007.

TOMCAT Project. Disponível em: <http://jakarta.apache.org/tomcat/index.html>. Acesso em: 19 jun. 2007.

VIA DIGITAL. Disponível em: <http://viadigital.org.br>. Acesso em: 01 jul. 2007.

# 7. ANEXO

## 7.1. Requisitos dos cadastros do sistema

### 7.1.1. Inserir usuário da unidade consumidora

Este cadastro é utilizado quando o usuário administrador, o usuário digitador ou o usuário responsável pela unidade consumidora desejam adicionar um usuário a unidade consumidora. Este usuário será o beneficiado com os medicamentos disponíveis na unidade consumidora e receitados pelo profissional de saúde.

Para realizar o registro, será necessário preencher obrigatoriamente os campos Nome e Endereço, e os campos descrição, identidade, cpf e foto são opcionais. Um código é gerado automaticamente quando o usuário é cadastrado.

Descrições dos campos:

- Código: Um identificador numérico único auto-incrementado em cada registro na tabela.
- Nome: Nome do usuário da unidade consumidora (obrigatório).
- Endereço: Endereço do usuário da unidade consumidora (obrigatório).
- Descrição: Descrição ou observação do usuário da unidade consumidora (opcional).
- Identidade: RG do usuário da unidade consumidora (opcional).
- CPF: documento do usuário da unidade consumidora.
- Foto: Arquivo com a foto do usuário da unidade consumidora.

### 7.1.2. Alterar usuário da unidade consumidora

Para alterar o usuário da unidade consumidora, primeiramente se deve listar todos os usuários pré-cadastrados de acordo com o item 7.3.1. Assim o usuário administrador, usuário responsável pela unidade ou o usuário gestor podem selecionar o usuário que desejam alterar.

Após selecionado o usuário, a vizualização será de acordo com o item 7.1.4. Com o formulário exibido, o único campo que não poderá ser alterado é o campo “código”.

### 7.1.3. Remover usuário da unidade consumidora

Para remover um usuário da unidade consumidora, o usuário administrador, o gestor ou o responsável pela unidade consumidora, devem listar todos os usuários de acordo com o item 7.3.1 para poder ser selecionado.

Após selecionado o usuário, a visualização será de acordo com o item 7.1.4 e o usuário, então, poderá concluir a exclusão.

### 7.1.4. Visualizar usuário da unidade consumidora

O usuário administrador, o gestor ou o responsável pela unidade consumidora, para visualizar as informações dos usuários, devem primeiro listar todos os usuários de acordo com o item 7.3.1 para poder ser selecionado.

Após a seleção, serão exibidos os dados dos usuário com as informações: nome, descrição, endereço, identidade, CPF e foto.

### 7.1.5. Inserir profissional de saúde

Este cadastro é realizado quando o usuário administrador, o digitador ou o responsável pela unidade consumidora desejam adicionar um novo profissional de saúde no sistema. O profissional de saúde é quem solicita os remédios ao paciente.

Na inscrição do profissional da saúde, existem os seguintes campos:

- Código: Um identificador numérico único auto-incrementado em cada registro na tabela.
- Nome: Nome do profissional de saúde (obrigatório).
- Descrição: Descrição ou observação do profissional de saúde (opcional).
- Endereço: do profissional de saúde (opcional).
- Identificação Profissional: Identificação do profissional de saúde (obrigatório).
- CPF: documento do profissional de saúde (obrigatório).
- Foto: do profissional de saúde.
- Cargo: do profissional de saúde.

### 7.1.6. Alterar profissional de saúde

Para alterar o profissional de saúde, em primeiro lugar, deve-se enumerar todos os usuários pré-cadastrados de acordo com o item 7.3.2. Assim o usuário

administrador, o usuário responsável pela unidade ou o usuário gestor podem selecionar o usuário que desejam alterar.

Após selecionado o usuário, a visualização será de acordo com o item 7.1.8. Com o formulário exibido, o único campo que não poderá ser alterado é o campo “código”.

### 7.1.7. Remover profissional de saúde

Para remover um profissional de saúde, o usuário administrador, o gestor ou o responsável pela unidade consumidora devem listar todos os usuários de acordo com o item 7.3.2 para poder selecioná-los.

Após selecionado o usuário, a visualização será de acordo com o item 7.1.8 e o usuário, então, poderá concluir a exclusão.

### 7.1.8. Visualizar profissional de saúde

O usuário administrador, o gestor ou o responsável pela unidade consumidora, para visualizar as informações dos usuários, devem primeiramente listar todos os usuários de acordo com o item 7.3.2 para poder ser selecionado.

Após selecionado o usuário, serão exibidas as informações: nome, descrição, endereço, identidade profissional, CPF, foto e cargo.

## 7.2. Funcionalidades dos cadastros do sistema

### 7.2.1. Confirmar entrada dos medicamentos na unidade consumidora

Quando a unidade consumidora solicita medicamentos ao almoxarifado, e este, por sua vez, autoriza o envio, a solicitação fica pendente enquanto o medicamento não chegar à farmácia. Logo, quando o transporte é realizado, há necessidade de confirmação da entrada dos medicamentos na unidade consumidora.

Dessa forma, o usuário administrador ou o usuário responsável pela unidade consumidora devem verificar se os medicamentos estão de acordo com os solicitados. Para isso, devem primeiro listar todas as entradas pendentes de medicamento na unidade consumidora de acordo com o item 7.3.3. Posteriormente, o usuário deve localizar a solicitação correspondente.

Ao se visualizar a solicitação selecionada, aparecerá um formulário com os seguintes campos:

- Código de entrada: Identificador único gerado pelo almoxarifado (visualização).
- Categoria de entrada: do medicamento (visualização).
- Grupo: pertencente de medicamento gerado pelo almoxarifado (visualização).
- Medicamento: Nome do medicamento (visualização).
- Quantidade de medicamento: Quantidade enviada pelo almoxarifado (visualização).
- Usuário responsável pela unidade consumidora: Nome do usuário que está confirmando ou rejeitando a entrada de medicamento (visualização).

- Data de confirmação de entrada: Data atual preenchida automaticamente pelo sistema (visualização).
- Justificativa: Caso o usuário venha a rejeitar a entrada, deve descrever a justificativa (obrigatório).

Confirmada a entrada de medicamento, o estoque da unidade consumidora deve ser automaticamente atualizado.

### 7.2.2. Alocar entrada dos medicamentos na unidade consumidora

Após confirmar a entrada de medicamento, o usuário administrador ou o usuário responsável pela unidade consumidora devem determinar uma localização para esses medicamentos. Esse procedimento inicia com a listagem de todas as entradas de medicamentos confirmada, de acordo com o item 7.3.4, para que se possa selecionar a desejada.

Selecionado o medicamento, o usuário deverá preencher um formulário para distribuir a quantidade desses medicamentos em lotes. Para cada lote, deverá ser informada a quantidade, data de validade e localização do medicamento. Os campos do formulário são os seguintes:

- Nome: Nome do medicamento (visualização).
- Descrição: Descrição do medicamento (visualização).
- Similar: Se é similar ou não (visualização).
- Genérico: Se é genérico ou não (visualização).
- Classificação: Se é em gotas, injetável, pílulas (visualização).
- Forma de armazenamento: A forma como é armazenado (visualização).
- Laboratório: Laboratório do fabricante do remédio (visualização).

- Lote: Lote do medicamento (obrigatório).
- >> Quantidade: Unidades que irão pertencer a este lote (obrigatório).
- >> Data de validade: Data de validade do lote que irá ser armazenado (obrigatório).
- >> Localização no almoxarifado: Localização do medicamento no almoxarifado (Opcional). Ex.: Corredor 2, Prateleira 5.

Após confirmada a alocação, os medicamentos são automaticamentes disponíveis para a distribuição, e o estoque da unidade consumidora, atualizados.

### 7.2.3. Solicitar saída de medicamentos da unidade consumidora

Esta funcionalidade provavelmente será a mais utilizada nas unidades consumidoras. É utilizada quando o usuário administrador,o gestor ou o responsável pela unidade consumidora desejam dar baixa em uma quantidade de medicamento, seja por deterioração, perda ou solicitação do profissional de saúde.

Para cada tipo de solicitação, existe uma categoria. No caso de ela ser a de 'Distribuição ao Usuário da Unidade Consumidora', deve ser apresentada ao usuário responsável a receita médica com as descrições dos medicamentos. Em vista disso, o responsável pode preencher a solicitação com os seguintes campos:

- Código de Saída: Código identificador da Saída do Medicamento (Gerado automaticamente).
- Categoria da Saída: Categoria da Saída – neste caso, 'Distribuição ao Usuário da Unidade Consumidora' (visualização).
- Grupo: Grupo do medicamento (visualização).
- Medicamento: Nome do medicamento (visualização).

- Usuário da unidade consumidora: Usuário que solicitou o medicamento com a receita (visualização).
- Quantidade: Quantidade solicitada na receita (obrigatório).
- Usuário Gestor: Usuário logado no sistema que preencherá o formulário (visualização).
- Data da solicitação: Dia em que a solicitação é realizada, gerada automaticamente (visualização).
- Observação: Alguma observação com relação a solicitação (opcional).

Caso a categoria de saída for outra, se não 'Distribuição ao Usuário da Unidade Consumidora', o formulário será exibido com os seguintes campos:

- Código de Saída: Código identificador da Saída do Medicamento (Gerado automaticamente).
- Categoria de Saída: Categoria da Saída – neste caso, 'Distribuição ao Usuário da Unidade Consumidora' (visualização).
- Grupo: Grupo do medicamento (visualização).
- Medicamento: Nome do medicamento (visualização).
- Quantidade: Quantidade desejada para a solicitação de saída (obrigatório).
- Usuário responsável pela unidade consumidora: Usuário logado no sistema que preencherá o formulário (visualização).
- Data da solicitação: Dia em que a solicitação é realizada, gerada automaticamente (visualização).
- Observação: Alguma observação com relação a solicitação (opcional).

Após realizada a solicitação de saída, são vinculadas informações para facilitar ao usuário responsável a localização do medicamento e a quantidade a ser retirada em cada lote. Essas informações serão apresentadas com os seguintes campos:

- Grupo: Nome do grupo do medicamento.
- Medicamento: Nome do medicamento.
- Lote: Nome do lote.
- Quantidade: Quantidade solicitada.
- Localizações: Localização do medicamento cadastrada na sua alocação.

O sistema, ao vincular essas informações à solicitação, seleciona automaticamente o lote com o menor prazo de validade. Caso o usuário responsável queira escolher outro lote, há possibilidade, se a quantidade do medicamento solicitado não existe, o sistema avisa automaticamente.

Esses medicamentos solicitados mudam seu status para ‘medicamentos de saída autorizada’.

### 7.2.4. Confirmar saída de medicamentos da unidade consumidora

Após solicitar a saída de medicamento, o usuário administrador ou o responsável pela unidade consumidora devem confirmar a saída ao usuário da unidade consumidora (requisitante do remédio) ou de uma categoria diferente.

No caso de a saída ser ao requisitante do medicamento, constarão no formulário os seguintes campos:

- Usuário Gestor: Usuário que solicitou a saída do medicamento descrito no item anterior.
- Usuário da unidade consumidora: Usuário que solicitou o medicamento com a receita (visualização).
- Usuário responsável pela unidade unidade consumidora: Usuário logado no sistema que preencherá o formulário (visualização).
- Data: Dia em que a confirmação é realizada (obrigatório).
- Observação: Alguma observação que o usuário responsável queira fazer (opcional).

No caso de uma categoria diferente da 'Distribuição ao Usuário da Unidade Consumidora', o formulário apresentará os seguintes campos:

- Usuário Gestor: Usuário que solicitou a saída do medicamento descrito no item anterior (visualização).
- Usuário responsável pela unidade consumidora: Usuário logado no sistema que preencherá o formulário (visualização).
- Data: Dia em que a confirmação é realizada (obrigatório).
- Observação: Alguma observação que o usuário responsável queira fazer (opcional).

Confirmada a saída do medicamento, é realizada baixa no estoque atual da unidade consumidora que o forneceu.

### 7.2.5. Imprimir requisição/saída do medicamento

Nesta funcionalidade, o usuário administrador ou o responsável pela unidade consumidora imprimirão uma via com as informações da saída do medicamento, para o usuário da unidade consumidora (requisitante) assiná-la. Ao confirmar uma entrega de medicamento, a impressão é feita de forma automática e servirá como comprovante de entrega do medicamento.

Esse comprovante terá as seguintes informações:

- Usuário Gestor: Usuário que solicitou a saída do medicamento descrito no item 7.2.3 (visualização).
- Usuário Responsável pela Unidade Consumidora: Usuário que confirmou a entrega ao requisitante.
- Usuário da Unidade Consumidora: Usuário requisitante do medicamento.
- Grupo: Grupo do medicamento.
- Medicamento: Nome do medicamento.
- Quantidade: Quantidade requisitada.
- Data de entrega: Data em que o medicamento foi entregue.

## 7.3. Requisitos dos Relatórios/Listagem do sistema

Na seqüência, serão apresentados os relatórios que os usuários administradores, responsáveis pela unidade ou gestores podem realizar. Esses relatórios podem ser gerados em PDF e impressos em folha A4.

### 7.3.1. Relatório/Listagem dos usuários de uma unidade consumidora

Lista em ordem alfabética dos usuários das unidades consumidoras cadastrados no sistema. Esses usuários são os que solicitam remédios com receitas médicas. A lista terá o nome e CPF dos usuários. Somente o usuário administrador pode escolher em listar por unidade consumidora distinta. Os usuários responsáveis por uma unidade consumidora poderão listar apenas usuários que pertencem a sua respectiva unidade.

### 7.3.2. Relatório/Listagem dos profissionais de saúde

Listagem em ordem alfabética de todos os profissionais de saúde cadastrados no sistema. Essa lista terá as seguintes informações: Nome, identificação Profissional, CPF e Cargo.

### 7.3.3. Relatório/Listagem de entrada pendentes de medicamentos na unidade consumidora (por grupo de medicamentos)

Nesta listagem, os usuários responsáveis pela gestão do sistema podem ver os medicamentos que estão com a entrada pendente nas unidades consumidoras por grupo de medicamentos.

A listagem pode ser feita com os seguintes filtros:

- Categoria: Caso o usuário queira a lista de uma categoria.

- Unidade Consumidora: Caso o usuário for administrador, pode selecionar a unidade consumidora que deseja listar as entradas pendentes.
- Grupo de Medicamento: Caso o usuário não informe um grupo de medicamento, a Listagem não terá o detalhamento do grupo.

O Relatório/listagem conterá as seguintes informações para visualização:

- Unidade consumidora: Nome da unidade consumidora (farmácia pública).
- Grupo: Grupo do medicamento.
- >> Código de entrada: Identificador único do medicamento.
- >> Nome: Nome do medicamento.
- >>> Lote: Lote do medicamento.
- >>>> Quantidade: Quantidade do medicamento pertencente ao lote.
- >>>> Data de Validade: Data de validade do medicamento pertencente ao lote.

### 7.3.4. Relatório/Listagem de entrada confirmada de medicamentos na unidade consumidora (por grupos de medicamentos)

Nesta listagem, os usuários responsáveis pela gestão do sistema podem ver os medicamentos que estão com a entrada confirmada nas unidades consumidoras por grupo de medicamentos.

A listagem pode ser feita com os seguintes filtros:

- Categoria: Caso o usuário queira a lista de uma categoria.

- Unidade Consumidora: Caso o usuário for administrador, pode selecionar a unidade consumidora da qual deseja listar as entradas confirmadas.
- Grupo de Medicamento: Caso o usuário não informe um grupo de medicamento, a Listagem não terá o detalhamento do grupo.

O Relatório/listagem conterá as seguintes informações para visualização:

Unidade consumidora: Nome da unidade consumidora (farmácia pública).

- Grupo: Grupo do medicamento.
- >> Código de entrada: Identificador único do medicamento.
- >> Nome: Nome do medicamento.
- >>> Lote: Lote do medicamento.
- >>>> Quantidade: Quantidade do medicamento pertencente ao lote.
- >>>> Data de Validade: Data de validade do medicamento pertencente ao lote.

### 7.3.5. Relatório/Listagem das entradas alocadas de medicamentos na unidade consumidora (por grupo de medicamentos entre um intervalo de datas)

Este relatório/listagem fornece informações sobre as entradas alocadas de medicamentos na unidade consumidora em um intervalo de tempo. O relatório pode ter os seguintes filtros:

- Categoria: Caso o usuário queira a lista de uma categoria.
- Unidade Consumidora: Caso o usuário for administrador, pode selecionar a unidade consumidora que deseja listar as entradas confirmadas.

- Grupo de Medicamento: Caso o usuário não informe um grupo de medicamento, a listagem não terá o detalhamento do grupo.
- De: Data inicial do intervalo em que o lote dos grupos de medicamentos foram alocados (obrigatório).
- A: Data final do intervalo em que o lote dos grupos de medicamentos foram alocados (obrigatório).

A listagem conterá as seguintes informações:

- Unidade consumidora: Nome da unidade consumidora (farmácia pública).
- Grupo: Grupo do medicamento.
- >> Código de entrada: Identificador único do medicamento.
- >> Nome: Nome do medicamento.
- >>> Lote: Lote do medicamento.
- >>>> Quantidade: Quantidade do medicamento pertencente ao lote.
- >>>> Data de Validade: Data de validade do medicamento pertencente ao lote.

### 7.3.6. Relatório/Listagem dos medicamentos do estoque de uma unidade consumidora (por grupo de medicamentos)

Relatório/Listagem que informa todos as entradas por grupos de medicamentos, alocados ou confirmados na unidade consumidora. A listagem pode ser filtrada pelo grupo do medicamento e disponibilizará as seguintes informações:

- Unidade consumidora: Nome da unidade consumidora (farmácia pública).
- Grupo: Grupo do medicamento.

- >> Código de entrada: Identificador único do medicamento.
- >> Nome: Nome do medicamento.
- >>> Lote: Lote do medicamento.
- >>>> Quantidade: Quantidade do medicamento pertencente ao lote.
- >>>> Data de Validade: Data de validade do medicamento pertencente ao lote.
- >>>> Localização: Localização do lote do medicamento no almoxarifado da unidade consumidora.

### 7.3.7. Relatório/Listagem dos medicamentos com saída confirmada no total (por unidades consumidoras e entre um intervalo de datas)

Relatório/Listagem de todos as saídas confirmadas de medicamentos separados por grupos nas unidades consumidoras entre um intervalo de datas. No caso de o usuário ser o responsável pela unidade consumidora, poderá apenas listar os medicamentos da unidade da qual é responsável; caso o usuário for administrador, pode escolher a unidade consumidora como um filtro, caso o administrado não escolha, o relatório informará as saídas confirmadas de todas as unidades consumidoras.

Filtros deste relatório:

- Unidade consumidora: Nome da unidade consumidora (farmácia pública) que pode ser escolhida pelo administrador.
- De: Data inicial do intervalo em que a saída dos lote dos grupos de medicamentos foram confirmados (obrigatório).

- A: Data final do intervalo em que a saída dos lote dos grupos de medicamentos foram confirmados (obrigatório).

### 7.3.8. Relatório/Listagem dos medicamentos distribuídos (por usuário da unidade consumidora entre um intervalo de datas)

Relatório/Listagem dos medicamentos distribuídos, separados por usuário da unidade consumidora (favorecido) e intervalo de datas.

A listagem terá os seguintes filtros:

- Nome: Nome do usuário da unidade consumidora. Se não for fornecido, listará os medicamentos distribuídos de todos os usuários consumidores (opcional).
- De: Data inicial do intervalo em que a distribuição do medicamento foi realizada (obrigatório).
- A: Data final do intervalo em que a distribuição do medicamento foi realizada (obrigatório).

As informações conterão as seguintes informações:

- Nome: Nome do usuário da unidade consumidora.
- >Grupo: Nome do grupo do medicamento.
- >> Medicamento: Nome do medicamento.
- >> Quantidade: Quantidade distribuída.

### 7.3.9. Relatório/Listagem dos medicamentos distribuídos (por profissionais de saúde entre um intervalo de datas)

Relatório/listagem dos medicamentos distribuídos separados por profissionais de saúde entre um intervalo de datas.

A listagem terá os seguintes filtros:

- Nome: Nome do profissional de saúde. Se não fornecido, listará os medicamentos distribuídos por todos os profissionais de saúde (opcional).
- De: Data inicial do intervalo em que a distribuição do medicamento foi realizada (obrigatório).
- A: Data final do intervalo em que a distribuição do medicamento foi realizada (obrigatório).

As informações conterão os seguintes dados:

- Nome: Nome do profissional de saúde.
- >Grupo: Nome do grupo do medicamento.
- >> Medicamento: Nome do medicamento.
- >> Quantidade: Quantidade distribuída.

### 7.3.10. Relatório/Listagem dos medicamentos distribuídos (por usuários gestores do sistema entre um intervalo de datas)

Relatório/listagem dos medicamentos distribuídos, separados por usuários gestores entre um intervalo de datas.

A listagem terá os seguintes filtros:

- Nome: Nome do usuário gestor. Se não for fornecido, listará os medicamentos distribuídos por todos os usuários gestores (opcional).
- De: Data inicial do intervalo em que a distribuição do medicamento foi realizada (obrigatório).
- A: Data final do intervalo em que a distribuição do medicamento foi realizada (obrigatório).

As informações conterão os seguintes dados:

- Nome: Nome do usuário gestor.
- >Grupo: Nome do grupo do medicamento.
- >> Medicamento: Nome do medicamento.
- >> Quantidade: Quantidade distribuída.

### 7.3.11. Relatório/Listagem de medicamentos da unidade consumidora que estão para expirar a validade (especificando o número de dias)

Relatório/listagem dos medicamentos que irão vencer em um número de dias fornecido pelo usuário. Caso o usuário for administrador ou gestor, pode selecionar uma unidade consumidora ou listar os medicamentos de todas elas. Se for usuário responsável pela unidade, só poderá visualizar os medicamentos de sua respectiva unidade.

Campos para filtro:

- Unidade Consumidora: Seleção da unidade consumidora caso o usuário for administrador (opcional).

- Quantidade de dias: Número de dias em que os medicamentos irão vencer a partir da data em que o relatório for gerado (obrigatório).

Informações para visualização:

- Grupo: Grupo do medicamento.
- Nome: Nome do medicamento.
- >> Lote: Lote do medicamento.
- >>> Quantidade: Quantidade do medicamento pertencente ao lote.
- >>> Data de Validade: Data de validade do medicamento pertencente ao lote.
- >>> Localização: Localização do lote do medicamento no almoxarifado da unidade consumidora.

### 7.3.12. Relatório/Listagem de medicamentos da unidade consumidora que ultrapassaram o prazo de validade

Relatório/Listagem dos medicamentos que estão vencidos. Se o usuário for administrador ou gestor, pode selecionar uma unidade consumidora ou listar os medicamentos de todas elas. Se for usuário responsável pela unidade, somente poderá visualizar os medicamentos de sua respectiva unidade.

Campos para filtro:

- Unidade Consumidora: Seleção da unidade consumidora caso o usuário for administrador (opcional).

Informações para visualização:

- Grupo: Grupo do medicamento.
- Nome: Nome do medicamento.
- >> Lote: Lote do medicamento.
- >>> Quantidade: Quantidade do medicamento pertencente ao lote.
- >>> Data de Validade: Data de validade do medicamento pertencente ao lote.
- >>> Localização: Localização do lote do medicamento no almoxarifado da unidade consumidora.

## 7.4. Exemplo de JSP para html

JSP:

```
<%@taglib prefix="s" uri="/struts-tags"%>
<p class="titulo2">Profissionais de Saúde</p>
   <s:if test="profissionais.size > 0">
      <table border="1">
       <tr class="errorMessage">
          <td>Nome</td>
          <td>Identifica&ccedil;&atilde;o</td>
          <td>CPF</td>
          <td>Cargo</td>
          <td>A&ccedil;&atilde;o</td>
      </tr>
      <s:iterator value="profissionais">
         <tr id="row_<s:property value="id"/>">
            <s:url id="viewUrl" action="view">
              <s:param name="id" value="id" />
            </s:url>
            <td>
               <s:a href="%{viewUrl}" targets="profissional" property
value="profissional.nome" />
            </td>
            <td>
```

```
                <s: targets="profissional" property
value="profissional.identidicacao" />
            </td>
            <td>
                <s: targets="profissional" property value="profissional.cpf"
/>
            </td>
            <td>
                <s: targets="profissional" property
value="profissional.cargo" />
            </td>
            <td>
                <s:url id="removeUrl" action="remove">
                    <s:param name="id" value="id" />
                </s:url>
                <s:a href="%{removeUrl}" targets="fornecedor">Remover</s:a>
                <s:url id="editUrl" action="edit">
                    <s:param name="id" value="id" />
                </s:url>
                <s:a href="%{editUrl}" targets="fornecedor">Editar</s:a>
            </td>
         </tr>
      </s:iterator>
   </table>
</s:if>
```

HTML:

```
<table border="1">
   <tr class="errorMessage">
      <td> Nome</td>
      <td>Identifica&ccedil;&atilde;o</td>
      <td>CPF</td>
      <td>Cargo</td>
      <td>A&ccedil;&atilde;o</td>
   </tr>
   <tr id="row_6" class="texto5">
      <td><a href="profissionais/view.action.5">Murilo Nunes Elias
</a></td>
      <td>CRM: 121354/SC </td>
      <td>04164516917</td>
```

```
      <td>M&eacute;dico</td>
      <td><a
href="profissionais/removeProfissionais.action.6">Remover</a><a
href="profissionais/editProfissionais.action.6">Editar</a></td>
   </tr>
   <tr id="row_5" class="texto5">
      <td><a href="profissionais/view.action.6">Rafael Bessan</a> </td>
      <td>CRM: 121354/SC </td>
      <td>04164516917</td>
      <td>Urologista</td>
      <td><a
href="profissionais/removeProfissionais.action.5">Remover</a><a
href="profissionais/editProfissionais.action.5">Editar</a></td>
  </tr>
</table>
```

Print Screen:


Figura 7 - Print Screen do navegador ao listar profissionais de saúde.

## 7.5. 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>

	<!-- === Definição de arquivos/recursos de mensagens e constantes ===
-->
	<bean id="messageSource"

	class="org.springframework.context.support.ResourceBundleMessageSourc
e">
		<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.ResourceBundleMessageSourc
e">
		<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 configuração. Carrega e propriedades (constantes)
da aplicação -->
	<bean id="propertyConfigurer"

	class="org.springframework.beans.factory.config.PropertyPlaceholderCo
nfigurer">
		<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.OsCacheManagerFactory
Bean">
      </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.DataSourceTransactionManag
er">
            <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> é o nome do atributo na classe <ref> é o nome da
referencia nesse arquivo
            <value> é um valor que pode ser definido(vai ser pego do
aplication.properties) -->


      <bean id="profissionalSaudeDAO"
```

```
	class="br.org.viadigital.viafarmacos.model.dao.ProfissionalSaudeDAO">
		<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.EntradaSaidaMedicament
oDAO">
		<property name="dataSource">
			<ref local="dataSource" />
		</property>
		<property name="sqlMapClient">
			<ref local="sqlMapClient" />
		</property>
	</bean>

	<bean id="usuarioUnidadeConsumidoraDAO"

	class="br.org.viadigital.viafarmacos.model.dao.UsuarioUnidadeConsumid
oraDAO">
		<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 negócio - 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="entradaSaidaMedicamentoBusiness"

      class="org.springframework.transaction.interceptor.TransactionProxyFa
ctoryBean">
            <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.EntradaSaidaMedic
amentoBusiness">
            <property name="entradaSaidaMedicamentoDAO">
                  <ref local="entradaSaidaMedicamentoDAO" />
            </property>
      </bean>


      <bean id="usuarioUnidadeConsumidoraBusiness"

      class="org.springframework.transaction.interceptor.TransactionProxyFa
ctoryBean">
            <property name="transactionManager" ref="transactionManager" />
            <property name="target" ref="usuarioUnidadeConsumidoraTarget"
/>
            <property name="proxyTargetClass" value="true" />
            <property name="transactionAttributes">
                  <props>
                        <prop key="insertUsuarioUnidadeConsumidora">
                              PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
                              -Exception
                        </prop>
                        <prop key="updateUsuarioUnidadeConsumidora">
                              PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
                              -Exception
                        </prop>
                  </props>
            </property>
      </bean>
```

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

	class="br.org.viadigital.viafarmacos.model.business.UsuarioUnidadeCon
sumidoraBusiness">
		<property name="usuarioUnidadeConsumidoraDAO">
			<ref local="usuarioUnidadeConsumidoraDAO" />
		</property>

	</bean>


	<bean id="profissionalSaudeBusiness"

	class="org.springframework.transaction.interceptor.TransactionProxyFa
ctoryBean">
		<property name="transactionManager" ref="transactionManager" />
		<property name="target" ref="profissionalSaudeTarget" />
		<property name="proxyTargetClass" value="true" />
		<property name="transactionAttributes">
			<props>
				<prop key="insertProfissionalSaude">
					PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
					-Exception
				</prop>
				<prop key="updateProfissionalSaude">
					PROPAGATION_REQUIRED,
ISOLATION_READ_UNCOMMITTED,
					-Exception
				</prop>
			</props>
		</property>
	</bean>

	<bean id="profissionalSaudeTarget" parent="abstractBusiness"

	class="br.org.viadigital.viafarmacos.model.business.ProfissionalSaude
Business">
		<property name="profissionalSaudeDAO">
			<ref local="profissionalSaudeDAO" />
		</property>

	</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="entradaSaidaMedicamentoAction"

      class="br.org.viadigital.viafarmacos.web.action.EntradaSaidaMedicamen
toAction">
            <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" />
            <constructor-arg ref="profissionalSaudeBusiness" />
            <constructor-arg ref="usuarioUnidadeConsumidoraBusiness" />
      </bean>

      <bean id="usuarioUnidadeConsumidoraAction"

      class="br.org.viadigital.viafarmacos.web.action.UsuarioUnidadeConsumi
doraAction">
            <constructor-arg ref="usuarioUnidadeConsumidoraBusiness" />
            <constructor-arg ref="unidadeConsumidoraBusiness" />
      </bean>

      <bean id="profissionalSaudeAction"

      class="br.org.viadigital.viafarmacos.web.action.ProfissionalSaudeActi
on">
            <constructor-arg ref="profissionalSaudeBusiness" />
            <constructor-arg ref="unidadeConsumidoraBusiness" />
      </bean>


      <!-- === Timer Tasks === -->
      <!-- timer tasks (declaração 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>
```

## 7.6. 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="false" />

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

            <action name="listEntradaPendenteUnidadeConsumidora"
                  method="executeEntradaPendenteUnidadeConsumidora"
                  class="entradaSaidaMedicamentoAction">
                  <result>
      /jsp/listarEntradasPendentesMedicamentoUnidadeConsumidora.jsp
                  </result>
                  <result name="input">
      /jsp/listarEntradasPendentesMedicamentoUnidadeConsumidora.jsp
                  </result>
            </action>

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

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

            <action name="viewEntradaPendenteUnidadeConsumidora"
                  method="prepareEntradaPendenteUnidadeConsumidora"
                  class="entradaSaidaMedicamentoAction">
                  <result>
      /jsp/visualizarEntradaPendenteMedicamentoUnidadeConsumidora.jsp
                  </result>
                  <result name="input">
      /jsp/visualizarEntradaPendenteMedicamentoUnidadeConsumidora.jsp
                  </result>
            </action>

            <action name="listSaidaPendenteUnidadeConsumidora"
                  method="executeSaidaPendenteUnidadeConsumidora"
                  class="entradaSaidaMedicamentoAction">
                  <result>
      /jsp/listarSaidasPendentesMedicamentoUnidadeConsumidora.jsp
                  </result>
```

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

      <action
name="rejeitarEntradaPendenteMedicamentoUnidadeConsumidora"
method="rejeitarEntradaPendenteMedicamentoUnidadeConsumidora"
            class="entradaSaidaMedicamentoAction">
            <result>
/jsp/listarEntradasPendentesMedicamentoUnidadeConsumidora.jsp
            </result>
            <result name="input">
/jsp/listarEntradasPendentesMedicamentoUnidadeConsumidora.jsp
            </result>
            <result name="sussess">
/jsp/listarEntradasPendentesMedicamentoUnidadeConsumidora.jsp
            </result>
      </action>

      <action
name="confirmarEntradaPendenteMedicamentoUnidadeConsumidora"
method="confirmarEntradaPendenteMedicamentoUnidadeConsumidora"
            class="entradaSaidaMedicamentoAction">
            <result>
/jsp/listarEntradasPendentesMedicamentoUnidadeConsumidora.jsp
            </result>
            <result name="input">
/jsp/listarEntradasPendentesMedicamentoUnidadeConsumidora.jsp
            </result>
      </action>

      <action name="confirmarSaidaMedicamentoUnidadeConsumidora"
            method="confirmarSaidaMedicamentoUnidadeConsumidora"
            class="entradaSaidaMedicamentoAction">
            <result>
/jsp/listarSaidasPendentesMedicamentoUnidadeConsumidora.jsp
            </result>
            <result name="input">
/jsp/listarSaidasPendentesMedicamentoUnidadeConsumidora.jsp
            </result>
      </action>

      <action name="listEntradaConfirmadaUnidadeConsumidora"
            method="executeEntradaConfirmadaUnidadeConsumidora"
            class="entradaSaidaMedicamentoAction">
            <result>
/jsp/listarEntradasConfirmadasMedicamentoUnidadeConsumidora.jsp
            </result>
            <result name="input">
/jsp/listarEntradasConfirmadasMedicamentoUnidadeConsumidora.jsp
            </result>
      </action>

      <action name="viewEntradaConfirmadaUnidadeConsumidora"
            method="prepareEntradaConfirmadaUnidadeConsumidora"
            class="entradaSaidaMedicamentoAction">
            <result>
/jsp/visualizarEntradaConfirmadaMedicamentoUnidadeConsumidora.jsp
            </result>
```

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

            <action name="viewSaidaPendenteUnidadeConsumidora"
                  method="prepareSaidaPendenteUnidadeConsumidora"
                  class="entradaSaidaMedicamentoAction">
                  <result>
      /jsp/visualizarSaidaPendenteMedicamentoUnidadeConsumidora.jsp
                  </result>
                  <result name="input">
      /jsp/visualizarSaidaPendenteMedicamentoUnidadeConsumidora.jsp
                  </result>
            </action>

            <action name="alocarEntradaMedicamentoUnidadeConsumidora"
                  method="alocarEntradaMedicamentoUnidadeConsumidora"
                  class="entradaSaidaMedicamentoAction">
                  <result>
      /jsp/listarEntradasConfirmadasMedicamentoUnidadeConsumidora.jsp
                  </result>
                  <result name="input">
      /jsp/listarEntradasConfirmadasMedicamentoUnidadeConsumidora.jsp
                  </result>
            </action>

            <action name="saveSaidaMedicamentoUnidadeConsumidora"
                  method="executeSaidaMedicamentoUnidadeConsumidora"
                  class="entradaSaidaMedicamentoAction">
                  <result>
                        /jsp/inserirSaidaMedicamentoUnidadeConsumidora.jsp
                  </result>
                  <result name="input">
                        /jsp/inserirSaidaMedicamentoUnidadeConsumidora.jsp
                  </result>
            </action>

            <action name="saveSaidaMedicamentoUnidade"
                  method="saveSaidaMedicamentoUnidadeConsumidora"
                  class="entradaSaidaMedicamentoAction">
                  <result>
      /jsp/listarSaidasPendentesMedicamentoUnidadeConsumidora.jsp
                  </result>
                  <result name="input">
      /jsp/listarSaidasPendentesMedicamentoUnidadeConsumidora.jsp
                  </result>
                  <result name="sussess">
      /jsp/listarSaidasPendentesMedicamentoUnidadeConsumidora.jsp
                  </result>
            </action>

      </package>

      <package name="usuarioUnidadeConsumidora"
            namespace="/usuarioUnidadeConsumidora" extends="struts-
default">
            <action name="remove" method="remove"
                  class="usuarioUnidadeConsumidoraAction">
                  <result>/jsp/listarUsuarioUnidadeConsumidora.jsp</result>
                  <result name="input">
```

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

            <action name="save" method="save"
                  class="usuarioUnidadeConsumidoraAction">
                  <result>/jsp/listarUsuarioUnidadeConsumidora.jsp</result>
                  <result name="input">
                        /jsp/inserirUsuarioUnidadeConsumidora.jsp
                  </result>
                  <result name="sussess">
                        /jsp/listarUsuarioUnidadeConsumidora.jsp
                  </result>
            </action>

            <action name="list" class="usuarioUnidadeConsumidoraAction">
                  <result>/jsp/listarUsuarioUnidadeConsumidora.jsp</result>
                  <result name="input">
                        /jsp/listarUsuarioUnidadeConsumidora.jsp
                  </result>
            </action>

            <action name="saveUsuarioUnidadeConsumidora" method="execute"
                  class="usuarioUnidadeConsumidoraAction">

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

            <action name="edit" method="prepare"
                  class="usuarioUnidadeConsumidoraAction">

      <result>/jsp/inserirUsuarioUnidadeConsumidora.jsp</result>
                  <result name="input">
                        /jsp/inserirUsuarioUnidadeConsumidora.jsp
                  </result>
                  <result name="sussess">
                        /jsp/listarUsuarioUnidadeConsumidora.jsp
                  </result>
            </action>

            <action name="view" method="prepare"
                  class="usuarioUnidadeConsumidoraAction">
                  <result>
                        /jsp/visualizarUsuarioUnidadeConsumidora.jsp
                  </result>
                  <result name="input">
                        /jsp/visualizarUsuarioUnidadeConsumidora.jsp
                  </result>
            </action>

      </package>

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

            <action name="remove" method="remove"
                  class="profissionalSaudeAction">
                  <result>/jsp/listarProfissionalSaude.jsp</result>
```

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

      <action name="save" method="save"
            class="profissionalSaudeAction">
            <result>/jsp/listarProfissionalSaude.jsp</result>
            <result name="input">
                  /jsp/inserirProfissionalSaude.jsp
            </result>
            <result name="sussess">
                  /jsp/listarProfissionalSaude.jsp
            </result>
      </action>

      <action name="list" class="profissionalSaudeAction">
            <result>/jsp/listarProfissionalSaude.jsp</result>
            <result name="input">
                  /jsp/listarProfissionalSaude.jsp
            </result>
      </action>

      <action name="saveProfissionalSaude" method="execute"
            class="profissionalSaudeAction">
            <result>/jsp/inserirProfissionalSaude.jsp</result>
            <result name="input">
                  /jsp/inserirProfissionalSaude.jsp
            </result>
      </action>

      <action name="edit" method="prepare"
            class="profissionalSaudeAction">
            <result>/jsp/inserirProfissionalSaude.jsp</result>
            <result name="input">
                  /jsp/inserirProfissionalSaude.jsp
            </result>
            <result name="sussess">
                  /jsp/listarProfissionalSaude.jsp
            </result>
      </action>

      <action name="view" method="prepare"
            class="profissionalSaudeAction">
            <result>/jsp/visualizarProfissionalSaude.jsp</result>
            <result name="input">
                  /jsp/visualizarProfissionalSaude.jsp
            </result>
      </action>

</package>

 </struts>
```

## 7.7. UsuarioUnidade.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="UsuarioUnidadeConsumidoraDAO">
      <typeAlias alias="usuarioUnidadeConsumidora"
type="br.org.viadigital.viafarmacos.model.dto.UsuarioUnidadeConsumidora"/>
      <typeAlias alias="unidadeConsumidora"
type="br.org.viadigital.viafarmacos.model.dto.UnidadeConsumidora"/>
      <typeAlias alias="endereco"
type="br.org.viadigital.viafarmacos.model.dto.Endereco"/>
      <typeAlias alias="int" type="java.lang.Integer"/>

      <resultMap id="result-usuarioUnidadeConsumidora"
class="usuarioUnidadeConsumidora">
            <result property="id" column="codigo"/>
            <result property="nome" column="nome"/>
      <result property="unidadeConsumidora.id" column="id_unidade"/>
      <result property="unidadeConsumidora.nome" column="nome_unidade"/>
            <result property="descricao" column="descricao"/>
            <result property="identidade" column="identidade"/>
            <result property="cpf" column="cpf"/>
            <result property="foto" column="foto"/>
         <result property="endereco.id" column="endereco"/>
      <result property="endereco.logradouro" column="logradouro"/>
      <result property="endereco.numero" column="numero"/>
      <result property="endereco.bairro" column="bairro"/>
      <result property="endereco.cidade" column="cidade"/>
      <result property="endereco.estado" column="estado"/>
      <result property="endereco.complemento" column="complemento"/>
      <result property="endereco.cep" column="cep"/>
      </resultMap>

      <select id="selectUsuarioUnidadeConsumidora" resultMap="result-
usuarioUnidadeConsumidora" parameterClass="int"
resultClass="usuarioUnidadeConsumidora">
      select
uuc.codigo,uuc.nome,uuc.descricao,uuc.identidade,uuc.cpf,uuc.foto,uc.codigo
as id_unidade,
   uc.nome as nome_unidade, e.codigo as endereco, e.logradouro, numero,
bairro, cidade, estado,
   complemento,cep
   from usuario_da_unidade_consumidora uuc
   inner join endereco e on e.codigo=uuc.id_endereco
   inner join unidade_consumidora uc on
uc.codigo=uuc.id_unidade_consumidora
            where
                  uuc.codigo=#id#
      </select>

      <select id="selectUsuarioUnidadeConsumidoras" resultMap="result-
usuarioUnidadeConsumidora" parameterClass="usuarioUnidadeConsumidora"
resultClass="usuarioUnidadeConsumidora">
            select
uuc.codigo,uuc.nome,uuc.descricao,uuc.identidade,uuc.cpf,uuc.foto,uc.codigo
as id_unidade,
   uc.nome as nome_unidade, e.codigo as endereco, e.logradouro, numero,
bairro, cidade, estado,
   complemento,cep
   from usuario_da_unidade_consumidora uuc
```

```
   inner join endereco e on e.codigo=uuc.id_endereco
   inner join unidade_consumidora uc on
uc.codigo=uuc.id_unidade_consumidora order by nome
      </select>

      <select id="selectUsuarioUnidadeConsumidorasByGrupo"
resultMap="result-usuarioUnidadeConsumidora" parameterClass="grupo"
resultClass="usuarioUnidadeConsumidora">
            select
uuc.codigo,uuc.nome,uuc.descricao,uuc.identidade,uuc.cpf,uuc.foto,uc.codigo
as id_unidade,
   uc.nome as nome_unidade, e.codigo as endereco, e.logradouro, numero,
bairro, cidade, estado,
   complemento,cep
   from usuario_da_unidade_consumidora uuc
   inner join endereco e on e.codigo=uuc.id_endereco
   inner join unidade_consumidora uc on
uc.codigo=uuc.id_unidade_consumidora
            where
                  uc.codigo = #id#
      </select>

      <select id="selectMaxUsuarioUnidadeConsumidora" resultClass="int">
            select
                  max(codigo)
            from
                  usuario_da_unidade_consumidora
      </select>

      <update id="updateUsuarioUnidadeConsumidora"
parameterClass="usuarioUnidadeConsumidora">
            update
                  usuario_da_Unidade_Consumidora
            set
                  nome = #nome#,
                  id_unidade_consumidora = #unidadeConsumidora.id#,
                  descricao = #descricao#,
                  identidade = #identidade#,
                  cpf = #cpf#,
                  foto = #foto#,
                  id_endereco = #endereco.id#
            where
                  codigo = #id#
      </update>

      <insert id="insertUsuarioUnidadeConsumidora"
parameterClass="usuarioUnidadeConsumidora">

            insert into usuario_da_Unidade_Consumidora (
                  codigo,
                  nome,
                  descricao,
                  identidade,
                  cpf,
                  foto,
                  id_endereco,
                  id_unidade_consumidora
            ) values (
                  (select CASE when max(codigo) notnull then max(codigo)+1
else 1 end from usuario_da_Unidade_Consumidora),
                  #nome#,
```

```
                #descricao#,
                #identidade#,
                #cpf#,
                #foto#,
                #endereco.id#,
                #unidadeConsumidora.id#
            )
        </insert>

        <delete id="removeUsuarioUnidadeConsumidora" parameterClass="int">
            delete from usuario_da_Unidade_Consumidora where codigo = #id#
        </delete>

</sqlMap>
```

## 7.8. Artigo

# INFORMATIZAÇÃO DE FARMÁCIAS PÚBLICAS UTILIZANDO SOFTWARE

**Murilo Nunes Elias**[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina(UFSC)
Caixa Postal 475 – 88.040-900 – Florianópolis – SC – Brazil

muriloe@inf.ufsc.br

***Resumo.*** *Este trabalho visa ajudar no gerenciamento das farmácias públicas brasileiras integradas com o almoxarifado de suas respectivas prefeituras por intermédio de um sistema WEB. O desenvolvimento do sistema conta com a ajuda do projeto via digital, que a alguns anos vem incentivando a construção de soluções para a informatização do setor publico e que requisitou junto a empresa SWFactory em 2004 os levantamentos de requisitos para o desenvolvimento deste trabalho. Espera-se com este trabalho um controle completo das rotinas de distribuição de remédios pelas prefeituras, tais como controle de estoque, entrada, saída, perca, distribuição de remédios em seus respectivos postos de distribuição.*

## 1. Informações Gerais

O sistema para gerenciamento de farmácias públicas foi desenvolvido para plataforma WEB seguindo os requisitos levantamentos pela empresa SWFactory. Todos os arquivos relacionados ao projeto, como requisitos, casos de uso e código fonte estão disponíveis no repositório do Via Digital [Via Digital].

## 2. O projeto

Após analisar a atual situação de informatização das farmácias publicas, muito importante para o amparo social nos municípios, ajudando toda a população na distribuição de remédios indicados pelos médicos , surgiu-se a idéia de desenvolver um projeto com software livre para facilitar sua gestão.

O protejo tem como funcionalidade principal facilitar a distribuição de remédios em farmácias públicas e postos de saúde de municípios de médio e de pequeno porte que necessitam de um sistema integrado com os almoxarifados da prefeitura e dos hospitais públicos.

Esse sistema integrado com outros componentes tem a capacidade de aumentar o controle de estoque como, por exemplo, verificar quais postos e farmácias estão com estoques baixos e providenciar uma recarga.

O sistema controla a entrada, saída, estorno e perda, de qualquer natureza, de todos os medicamentos distribuídos à população nas Farmácias Públicas e/ou Postos de Saúde, por meio do controle de lote e validade de cada medicamento movimentado.

Evita também a corrupção, pois há a possibilidade de verificar se o paciente, que adquiriu os remédios, realmente teve uma consulta marcada com o médico nos hospitais municipais ou postos de saúde. Esse procedimento reduziria os gastos públicos com a distribuição de remédio que, no ano de 2007, pode chegar a R$ 4,6 bilhões de reais.

## 3. Arquitetura

A sistema foi desenvolvido com a arquitetura MVC(Model-View-Controller), dividindo assim o desenvolvimento em camadas. A camada view corresponde a interface com o usuário, para que o sistema tenha uma boa usabilidade ela foi desenvolvida com as tecnologias JSP, CSS, Javascript e AJAX.

A camada Controller trata as ações do sistema, utiliza o framework Struts integrado ao framework Spring, de modo a facilitar a instanciação das actions.

A camada Model cuida das regras de negócio e da persistência dos dados, foi desenvolvida com as tecnologias Spring e Ibatis, que facilitam na elaboração das regras e a integração com o banco de dados PostgreSQL.

## 4. Sistema

Pode-se dividir o sistema em cadastros, funcionalidades e listagem, estas partes são descritas abaixo.

### 4.1. Cadastros

Para que as funcionalidades do sistema tenham parâmetros dinâmicos é necessário realizar cadastros de informações relacionadas as farmácias públicas. Os cadastros disponíveis no sistema são os de usuário da unidade consumidora e profissionais de saúde, é possível também editar um cadastro existente.

Abaixo pode-se observar um exemplo de um formulário de cadastro disponível no sistema:


**Figura 1. Cadastro de usuário da unidade consumidora**

## 4.2. Funcionalidades

As funcionalidades do sistema são responsáveis pela manipulação dos dados pré-cadastrados no sistema, dentre as funcionalidades do sistema podemos destacar a solicitação de saída de medicamentos da unidade consumidora aos usuários solicitantes e a confirmação de entrada de medicamentos no estoque da unidade consumidora. É através destas duas funcionalidades que o controle de estoque é realizada.

## 4.3. Listagem

A listagem corresponde na exibição de dados cadastrados no sistema de acordo com alguma característica ou não. Alguns exemplos são as listagem de medicamentos com entrada confirmada, dos medicamentos distribuídos por usuários e de medicamentos distribuídos por profissional de saúde. É através destas listas que se é possível selecionar um registro para visualizar, editar ou excluir seus dados.

Abaixo pode-se observar um exemplo de uma lista disponível no sistema:


**Figura 2. Listagem de entradas pendentes de medicamentos na unidade consumidora**

## 5. Conclusão

O sistema está pronto e apto para a utilização em todas as prefeituras do país que desejam controlar melhor a distribuição de medicamento em seus postos de saúde. O controle sistemático evita o desperdício, a perda de medicamentos e, principalmente, permite economia do dinheiro público.

Em quase todos os postos de saúde visitados neste ano (2007), com o objetivo de verificar o processo de distribuições de medicamentos em Florianópolis, observou-se a falta de um sistema que gerencie a distribuição de medicamento nas farmácias publicas.

Com base neste conhecimento, sugere-se, como trabalho futuro, mostrar às prefeituras as vantagens da implantação deste sistema e convencê-las a adotá-lo. Desse modo, implantando este sistema em todos os postos de saúde e dando capacitação para os usuários que irão geri-lo, as prefeituras terão um ótimo retorno pelo baixo investimento.

## 6. Referencias

Via Digital, projeto que apoiou o desenvolvimento deste e contém informações sobre outros projetos concluídos ou em andamento, disponível em: http://www.viadigital.ufsc.br/

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

SwFactory, empresa de consultoria e sistemas, http://www.swfactory.com.br/